ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ
Навчально-науковий інститут денної освіти
Форма навчання денна
Кафедра комп'ютерних наук та інформаційних технологій

Допускається до захисту
Завідувач кафедри
__Олена ОЛЬХОВСЬКА
(підпис)

«___»__202_ р.

КВАЛІФІКАЦІЙНА РОБОТА

на тему

«Розробка навчального програмного забезпечення з теми «Collections. LinkedList» дисципліни «Програмування»

зі спеціальності 122 Комп'ютерні науки освітня програма «Комп'ютерні науки» ступеня бакалавра

Виконавець роботи Кіліян Дмитро Володимирович

_____«___»_____202_ р.
(підпис)

Науковий керівник к. ф.-м. н., доцент, Олексійчук Юрій Федорович

_____«___»_____202_ р.
(підпис)

Рецензент _____

ПОЛТАВА 2024

РЕФЕРАТ

Записка: 70 с., 12 рис., 1 таблиця, 1 додаток, 17 джерел.

COLLECTIONS LINKEDLIST, НАВЧАЛЬНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ПРОГРАМУВАННЯ.

Об’єкт розробки – дистанційне навчання студентів спеціальності «Комп’ютерні науки».

Мета роботи – проєктування і програмна реалізація навчального програмного забезпечення з теми «Collections. LinkedList» дисципліни «Програмування».

Методи дослідження – інтерфейс класу LinkedList, інтерфейс програмування додатків Windows Forms у середовищі розробки програм Microsoft Visual Studio 2019 та об’єктно-орієнтована мова програмування C#.

Зроблено огляд програмного забезпечення для дистанційного навчання студентів спеціальності «Комп’ютерні науки», виявлено позитивні та негативні сторони. Розроблено алгоритм навчального програмного забезпечення з теми «Collections. LinkedList», побудовано його блок-схему.

Здійснено повну програмну реалізацію навчального програмного забезпечення з теми «Collections. LinkedList».

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Умовні позначення, символи, скорочення, терміни	Пояснення умовних позначень, символів, скорочень
тренажер	Комп’ютерна програма, що призначена для вивчення та засвоєння практичних навичок
C#	Об’єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET
.NET	Кероване програмне забезпечення з відкритим кодом для операційних систем Windows, Linux, macOS
Collections	Java collections framework це набір класів і інтерфейсів, котрі реалізують структури даних
Windows Forms	Інтерфейс програмування додатків, відповідальний за графічний інтерфейс і є частиною .NET Framework

Вступ

Сьогодні світ стрімко розвивається, і все більше сфер людської діяльності стають цифровими. Зокрема неупинно цей процес розвивається у навчанні. Вже довгий час онлайн-освіта є основним методом навчання, як для студентів чи школярів, так і для бажаючих власноруч освоїти або укріпити бажані навички. У зв'язку з цим зростає роль інформаційних технологій, зокрема, мов програмування.

Однією з найпопулярніших мов програмування у світі є Java. Вона використовується для розробки веб-додатків, мобільних додатків, десктопних додатків, ігор та інших програм. Вивчення Java дозволяє студентам отримати цінні навички, які будуть жаданими на ринку праці. Java-розробники є одними з найпопулярніших і високооплачуваних фахівців у світі інформаційних технологій.

Одним з важливих ступенів вивчення Java є детальний розбір бібліотеки стандартних колекцій Collections, котра містить широкий спектр класів для зберігання та управління даними. Її вивчення значно збільшує шанси на працевлаштування, адже освоєння цієї бібліотеки надає можливість створювати ефективні програми, що мають справу з великими обсягами даних.

Одним з найважливіших класів бібліотеки Collections в Java є клас LinkedList. Його динамічність, гнучкість та ефективність операцій вставки та видалення елементів роблять його більш вигідним для вивчення ніж інші класи бібліотеки Collections.

Метою роботи є реалізація навчальної програми з вивчення основних методів класу LinkedList.

Об’єктом розробки роботи є навчальне програмне забезпечення для LinkedList класу бібліотеки Collections.

Предмет розробки – безпосередньо програмний продукт, який реалізує навчальне забезпечення для вивчення класу LinkedList бібліотеки Collections для мови Java, реалізований на мові C# з застосуванням Windows Forms.

Головне завдання – розробити алгоритм роботи та виконати програмну реалізацію навчального програмного забезпечення.

Методи розробки – програмні засоби мови C#.

При реалізації навчальної програми використано мову програмування C# та середовище розробки Microsoft Visual Studio.

Робота має чотири розділи. У першому розглянуто постановку задачі та основних вимог до змісту. Другий розділ описує переваги використання програм для дистанційного навчання та складові дистанційної освіти. Третій розділ розглядає поняття теми “Collections.LinkedList”, також описано алгоритм роботи та блок-схему програмного забезпечення. У четвертому розділі описано обґрунтування вибору програмних засобів, процес програмної реалізації та інструкцію по використанню навчального програмного забезпечення.

Обсяг пояснювальної записки: 107 стор., основна частина – 70 стор., джерела – 17 назв.

1. Постановка задачі

1.1. Постановка основних вимог до змісту

Зміст роботи має відповідати завданню затвердженому керівником та завідувачем кафедри.

Виклад матеріалу:

- Чіткий, логічний, послідовний.
- Переконливий, з чіткою аргументацією.
- Стислий, точний, без неоднозначностей.
- Коректне викладення результатів дослідження.
- Обґрунтовані рекомендації та пропозиції.

Зміст роботи:

- Актуальна тематика, відповідна сучасному стану науки та техніки.
- Обґрунтування обраного напрямку досліджень, методів розв'язку задачі.
- Аналіз та узагальнення існуючих результатів.
- Характер та зміст теоретичних досліджень, розрахунків, методів досліджень.
- Принцип дії розроблених програм, їх характеристики.
- Теоретичний (практичний) аналіз розроблених алгоритмів (програм).
- Блок-схема програми (алгоритму).
- Приклади подібних робіт.
- Оцінка повноти вирішення поставленої задачі.
- Наукова або практична цінність виконаної роботи, виклад наукової новизни (якщо є).

1.2. Постановка задачі для реалізації навчального програмного забезпечення

Розглянемо основні завдання роботи:

- підготувати матеріал, котрий буде використовуватися при реалізації навчальної програми;
- розробити алгоритм програмного забезпечення для вивчення основ класу “LinkedList”;
- скласти блок-схему алгоритму роботи програмного забезпечення;
- розробити навчальне програмне забезпечення з даної теми.

Оскільки, програмне навчальне забезпечення розробляється з метою вивчення класу “LinkedList” для мови програмування Java, то пропонується наступний перелік тем для ознайомлення:

- Що таке Collections. LinkedList?;
- Переваги та недоліки LinkedList;
- Конструктори, методи та робота з ними;
- Додатково про методи класу LinkedList;
- Застосування Collections. LinkedList.

Розроблене програмне навчальне забезпечення повинно мати зрозумілий для користувача інтерфейс, тому слід зберегти таку структуру:

- Стартова сторінка:
  - Тема тренажера;
  - Вітальне повідомлення.
- Теоретична частина:

- Назва теми;
- Теоретичні відомості по ній.

- Частина теоретичних завдань:

- Завдання;
- Варіанти відповідей.

- Кінцева сторінка:

- Прощальне повідомлення.

2. ІНФОРМАЦІЙНА ЧАСТИНА

2.1.Складові дистанційної освіти

Дистанційна освіта має потенціал стати революційною формою навчання, вона вже робить освіту більш доступною, ефективною та персоналізованою. Вона вбирає в себе всі плюси заочної освіти доповнюючи їх очним спілкуванням з викладачем, незалежно від положення студента на земній кулі, що дає можливість залучити таку кількість студентів котру не може собі дозволити традиційне навчання.

Проте як і будь-яка форма навчання дистанційне навчання потребує організованості для підтримки своєї ефективності, складовими такої організаційної структури є:

- **Адміністрація:** відповідає за загальне керівництво та управління дистанційним навчанням;
- **Викладачі:** розроблюють навчальний матеріал, проводять заняття, за потреби оцінюють результати студентів при очному спілкуванні;
- **Технологічна підтримка:** забезпечує доступ до технологій, які використовуються в дистанційному навчанні;
- **Інформаційні ресурси:** курси та методичні матеріали;
- **Студенти.**

Організаційна структура дистанційного навчання може бути різною в залежності від умов його організації, але як правило, вона включає в себе всі перелічені елементи. Однак, для підтримки ефективності вона зобов'язана забезпечувати кожну складову структури виконувати перелік завдань:

Адміністрація:

- Розробка стратегії та політики дистанційного навчання;
- Забезпечення ресурсів, що потребує реалізація дистанційного навчання;
- Контроль якості навчання.

Викладачі:

- Розробка навчального матеріалу;
- Проведення занять;
- Оцінювання результатів навчання.

Технологічна підтримка:

- Забезпечення коректної роботи та постійного доступу до технологій, що використовуються у навчанні;
- Консультація та допомога студентам у налаштуванні технологій;
- Вирішення технічних проблем, що можуть виконати під час навчання.

Студенти:

- Самостійне навчання;
- Виконання завдань та участь у заняттях;
- Відповідне оцінювання.

Також окрім організаційної структури повинна бути присутня технічна структура котру обслуговує технічна підтримка:

- Сайт дистанційного курсу чи довідкових матеріалів;
- Налаштування систем зв’язку та інших засобів спілкування;
- Система адміністрування;
- Повноцінна система тестування та оцінювання абітурієнтів;
- Доступ студентів до навчального матеріалу.

Структура дистанційного навчання повинна підтримуватись базовими принципами до яких можна віднести:

- **Централізацію управління:** управління навчанням має бути централізованим задля забезпечення ефективної координації всіх його аспектів;
- **Делегування розпоряджень:** розпорядження повинні бути делеговані на нижчі рівні, щоб забезпечити оперативність системи навчання;
- **Гнучкість:** організаційна система має бути гнучкою, щоб адаптуватися до різних умов;

Використання ефективних принципів та структур організації дистанційного навчання дозволяє забезпечити його вражаючу ефективність, що і робить дистанційну форму навчання настільки революційною і доступною для сучасних студентів.

2.2. Використання навчального програмного забезпечення у дистанційному навчанні

Навчальне програмне забезпечення, зокрема тренажери є одним із найважливіших інструментів дистанційного навчання. Вони дозволяють отримувати доступ до навчального матеріалу, виконувати завдання та отримувати зворотний зв'язок у будь-який час і в будь-якому місці.

Тренажери можуть використовуватися для різних тем, включаючи математику, природничі науки, іноземні мови, програмування та інші. Вони є ефективним доповненням до традиційних методів навчання і можуть допомогти студентам краще засвоїти матеріал.

Легко виділити такі переваги тренажерів у дистанційному навчанні:

- **Гнучкість:** навчальні програми доступні в будь-який час, що дозволяє студентам навчатися в зручний для них час;

- **Фінансова доступність:** навчальне програмне забезпечення як і дистанційна освіта може бути більш економічно доступною, ніж традиційне навчання;
- **Індивідуальний підхід:** тренажери дозволяють студентам обирати цікави саме їм теми;
- **Зворотній зв'язок:** навчальні програми здатні надавати студентам інформацію про їх результати швидко, що допомагає зрозуміти помилки та покращити свої вміння;
- **Реалізація покрокового підходу:** можливість зручно подавати матеріал теми покроково чи таким чином контролювати вирішення задач з теми;
- **Автоматизація:** автоматизація процесу навчання за допомогою тренажера дозволяє викладачу підвищити ефективність своєї роботи;

Сукупність всіх цих факторів робить розробку та впровадження нових навчальних тренажерів все більш актуальною темою [10-17].

2.3. Аналіз існуючого навчального програмного забезпечення

Розвиток ефективного навчального тренажера є завданням, що вимагає розуміння підходів та практик у сфері навчання. Одним із ключових етапів цього процесу є аналіз існуючого навчального програмного забезпечення. Він дозволяє отримати унікальний досвід використання та розуміння інших педагогічних інструментів, що є ключовим для успішного створення тренажера.

Спостереження за функціональністю та взаємодією існуючих програм надає можливість визначити найбільш ефективні методи, засоби та технічні аспекти, які можна впровадити в новий тренажер. Виходячи із зібраної інформації, можливо створити тренажер, що відповідає всім вимогам подальших користувачів.

Першим буде розглядатися “Тренажер для вивчення основ мови програмування Python” [10] дисципліни “Програмування”, котрий розробив Мельницький Ян Віталійович в 2020 році.

Це навчальне програмне забезпечення створене для студентів, що прагнуть освоїти об’єктно-орієнтованого програмування, у цьому їм допоможе опанування мови Python, теоретичну та практичну інформацію до якої надає даний тренажер.

Тренажер зустрічає користувача інформацією про його назву та ім’ям розробника, при натисканні кнопки “Розпочати тестування” користувач переноситься на вікно яке представляє можливість обрати тему (Рисунок 2.1).

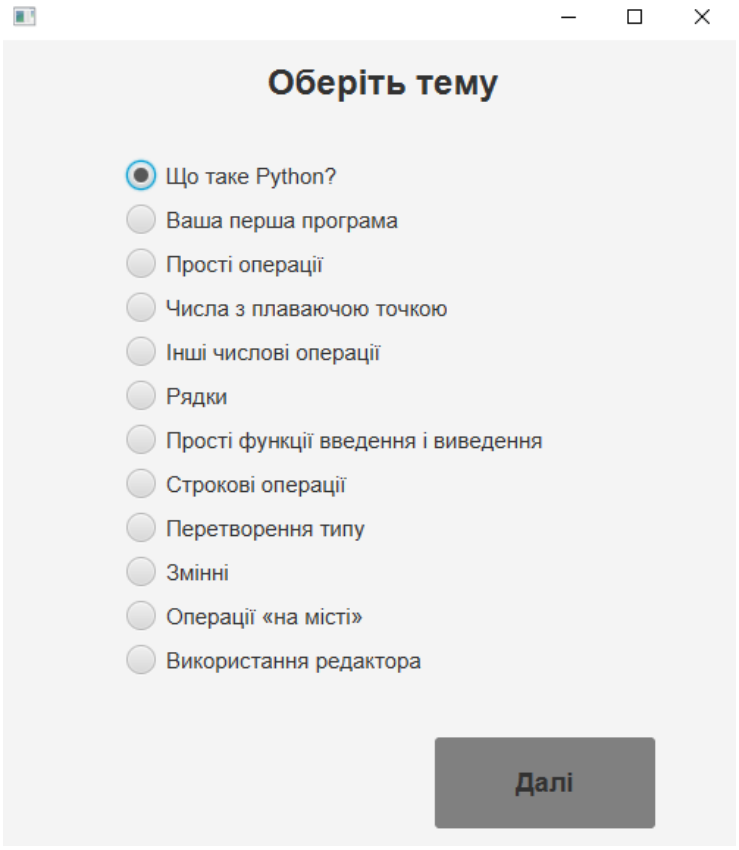

Рисунок 2.1 - Вікно програми-тренажера з теми “вивчення основ мови програмування Python” [10]

У вікні представлений конкретний набір тем котрі надають можливість користувачу обрати саме цікавий йому розділ, або продовжити з тієї теми на якій він зупинився минулого разу (Рисунок 2.1).

Також безпосередньо після ознайомлення з матеріалом слідують завдання для його закріплення, що покращує якість навчання за допомогою даного тренажера.

При виборі хибної відповіді користувач одразу дізнається про це, адже з’явиться повідомлення про похибку, завдяки якому одразу зможе зрозуміти чи слід йому повернутися до теоретичної частини.

Оцінювання реалізовано через систему жорсткого поділу відповідей, тобто відповідь, може бути або вірна або ні, це дозволяє користувачу чітко розуміти свої можливості та степінь розуміння матеріалу, що є однією з ключових вимог до навчального програмного забезпечення.

Після повного опрацювання теми, тренажер повертає вікно вибору однієї з представлених тем.

Підсумувавши всі ключові аспекти цього тренажера можна отримати основні плюси такої реалізації навчального програмного забезпечення:

1. Тема викладається розділено, що надає можливість мати більш індивідуальний підхід до користувача;
2. Послідовне представлення матеріалу, що позитивно відображається на якості вивчення матеріалу;
3. Зручний та зрозумілий функціонал програми;
4. Зрозуміла та прозора система оцінювання.

Загалом ці аспекти дають повне представлення про те, якою має бути навчальне програмне забезпечення, та цілком створюють загальне уявлення про тренажери.

Наступним на огляді буде тренажер з теми “Контекстовільні мови” [11] предмету “Теорія програмування”, що розроблений Лєбєвою Марією.

Програма зустрічає користувача вікном з інформацією про розробника та загальною темою тренажера та кнопкою “Почати роботу” (Рисунок 2.2).

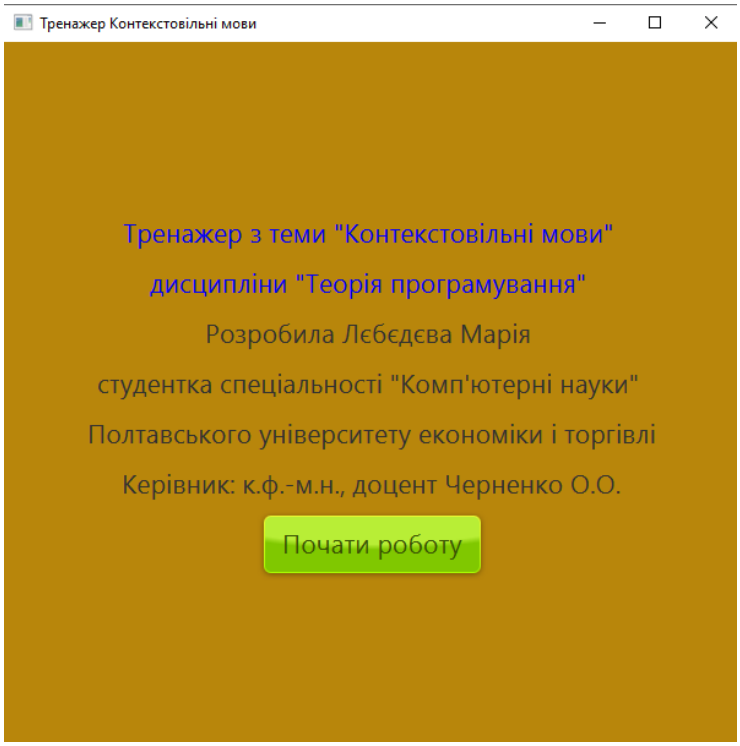

Рисунок 2.2 - Вікно програми-тренажера з теми “Контекстовільні мови” [11]

Після натискання на кнопку у вікні з’являється тестове завдання та варіанти відповідей з кнопкою “Відповісти”. У разі будь якої відповіді з’являється вікно, наповнення котрого відрізняється в залежності від відповіді.

Після проходження всіх тестових завдань тест закінчується та з’являється повідомлення запустити тест знову.

Порівнюючи цей Тренажер з попереднім можна одразу підкреслити декілька недоліків:

1. Програма виконує лише функцію тесту, потрібно подавати користувачеві інформацію в тренажері;

2. Не вистачає вінка вибору теми, задля того щоб користувач міг обрати зручний йому порядок вивчення матеріалу;
3. По закінченню проходження тесту зручно було б мати кнопку повернення до початку, щоб не закривати програму кожного разу задля повторного проходження.

Третій тренажер (Рисунок 2.3) теми “Синтаксичний аналіз” [12] для предмету “Теорія програмування” розроблений Корсун Ольгою та являє собою тест з 30 питань, після вірно обраного варіанту відповіді видаються рекомендація спробувати знову та надається можливість обрати нову відповідь.

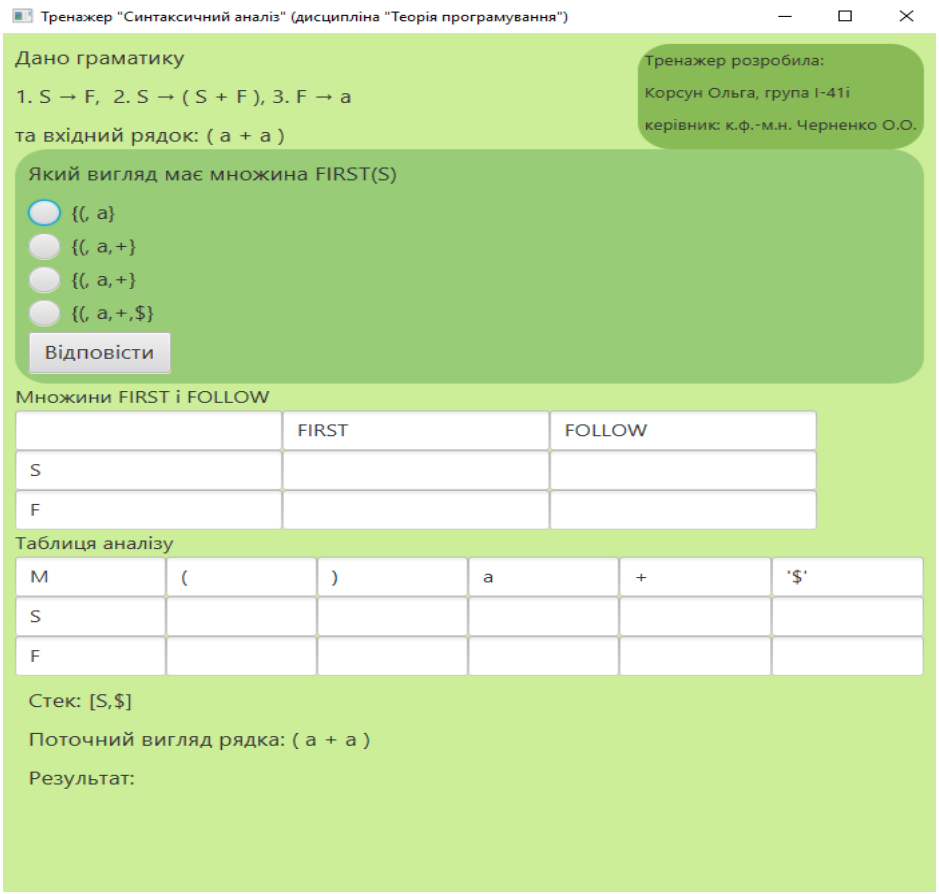

Рисунок 2.3 – Вікно програми-тренажера з теми “Синтаксичний аналіз” [12]

Протягом усього тесту вірні відповіді змінюють табличку заданих множин та таблицю аналізу і ілюструють роботу зі стеком.

Аналізуючи даний тренажер можна підкреслити велику відмінність у порівнянні з минулими програмами в подачі завдання. Та виділити такі недоліки:

1. Не вистачає теоретичної частини котра повинна саме навчати користувача;
2. Мала вибірка варіантів відповідей, вони завжди однакові, що дає змогу проходити тест лише інтуїтивно;
3. Знову не вистачає кнопки повернення до початку для повторного проходження;
4. Відсутність рекомендацій стосовно хибних відповідей.

Цікавим плюсом програми є те, що за допомогою таблиць в які вносяться зміни, користувач може відстежувати свій прогрес по завданням та наглядно впевнюватися в своїй кваліфікації, що позитивно впливає на зацікавленість під час проходження тесту.

В розробці програми я врахую зроблені підсумки, що надасть можливість врахувати помилки, котрі зробили минулі програми та взяти з них найкращі риси, які підходять обраному мною формату роботи.

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1. Основні поняття теми “Collections. LinkedList”

Що таке Collections.LinkedList?

Collections.LinkedList - це динамічна структура даних, що являє собою список елементів, пов'язаних один з одним за допомогою двоспрямованих посилань. **LinkedList** з'явився в **Java 1.2 (1998)** як альтернатива **ArrayList**. На відміну від **ArrayList**, який використовує фіксований масив для зберігання елементів, **LinkedList** не має фіксованого розміру та може динамічно розширюватися або скорочуватися під час додавання або видалення елементів. Він успадковує **AbstractSequentalList** і реалізує інтерфейси **List** та **Deque**(а через останній і **Queue**).

Структура (Рисунок 3.1)**:**

- **Елементи -** кожен елемент містить значення та два посилання: next, яке вказує на наступний елемент в списку, і previous, яке вказує на попередній елемент;
- **Голова -** це посилання на перший елемент в списку;
- **Хвіст -** це посилання на останній елемент в списку.

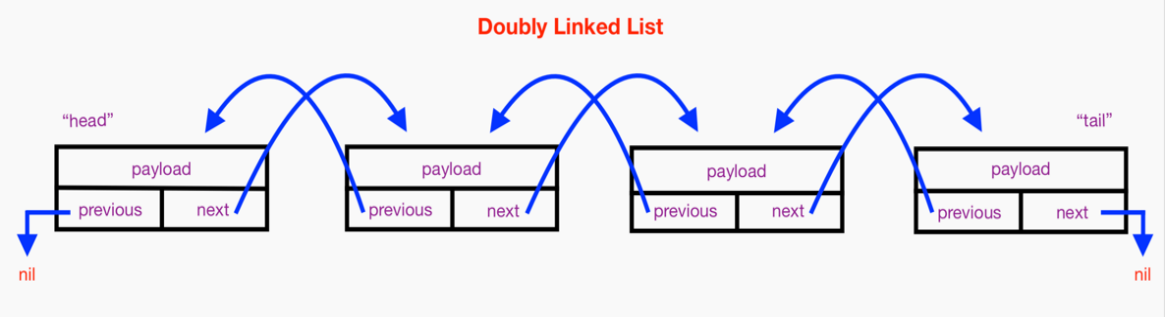

Рисунок 3.1 – Візуалізація подвійного зв'язаного списку LinkedList

Через динамічність та простоту вставок і видалень він краще масивів. Також він має кілька недоліків, такі як неможливість прямого доступу до елементів, натомість нам потрібно починати з початку та переходити за посиланням, щоб дістатися до елемента, який ми шукаємо, що значно збільшує час роботи операцій отримання елементів.

Порівняння часу роботи операції отримання елементу з індексом 8000000 для LinkedList та ArrayList:

```
int result;
startTime = System.nanoTime();
result = list1.get(8000000);
finishTime=System.nanoTime();
System.out.println(result);
System.out.println("ArrayList get: "+(double)(finishTime-startTime)/1000000
+"ms");

startTime = System.nanoTime();
result = list2.get(8000000);
finishTime=System.nanoTime();
System.out.println(result);
System.out.println("LinkedList get: "+(double)(finishTime-startTime)/1000000
+"ms");
```

Результат:

ArrayList get:0.00591ms
LinkedList get:11.678236ms

Оскільки **LinkedList** діє як динамічний масив, і нам не потрібно вказувати розмір під час його створення, розмір списку автоматично збільшується, коли ми додаємо та видаляємо елементи. **LinkedList** реалізований з використанням структури даних двозв'язкового списку

Основна відмінність між звичайним зв'язаним списком та подвійним зв'язаним списком полягає в тому, що двозв'язаковий список містить додатковий покажчик, який зазвичай називають попереднім покажчиком. Ці

додаткові посилання збільшують обсяг пам'яті, необхідний для зберігання **LinkedList**.

Переваги та недоліки

Надалі, порівняння **LinkedList** буде переважно проводитися саме з **ArrayList**, адже він створювався саме як альтернатива йому.

Переваги Collections.LinkedList:

- **Ефективне додавання та видалення елементів в середині списку -** є основною перевагою, завдяки двоспрямованим посиланням, додавання та видалення елементів в середині списку **LinkedList** не потребує переміщення інших елементів, як це відбувається в **ArrayList**. Це робить **LinkedList** більш ефективним для цих операцій;

- **Не потрібне попереднє виділення пам'яті - LinkedList** не потребує попереднього виділення пам'яті для зберігання елементів, оскільки розмір списку може динамічно змінюватися. Це може бути корисно для програм, де розмір списку невідомий заздалегідь;

- **Можливість реалізації черги (FIFO) або стека (LIFO) - LinkedList** може використовуватися для реалізації черги (FIFO), де елементи видаляються в тому ж порядку, в якому вони були додані, або стека (LIFO), де елементи видаляються в зворотному порядку до їх додавання.

Недоліки Collections.LinkedList:

- **Менш ефективний доступ до елементів за індексом -** доступ до елементів **LinkedList** за індексом може бути менш ефективним, ніж в **ArrayList**, оскільки **LinkedList** не використовує фіксований масив. Це

пов'язано з тим, що для доступу до елемента за індексом потрібно пройти через список, починаючи з першого елемента, доки не буде знайдений елемент з потрібним індексом;

- **Більше використання пам'яті - LinkedList** використовує більше пам'яті, ніж **ArrayList**, через додаткові посилання, необхідні для зв'язку елементів.

Конструктори, методи та робота з ними

Клас LinkedList має наступні конструктори:

- **LinkedList()** - котрий створює порожній список;
- **LinkedList(Collection<? extends E> col)** - створює список, в який додає всі елементи колекції col.

LinkedList містить всі методи, котрі містять в собі інтерфейси List, Queue, Deque, основні з них:

- **addFirst() / offerFirst()** - додає елемент в початок списку;
- **addLast() / offerLast()** - додає елемент в кінець списку;
- **removeFirst() / pollFirst()** - видаляє перший елемент з початку списку;
- **removeLast() / pollLast()** - видаляє останній елемент з кінця списку;
- **getFirst() / peekFirst()** - отримує перший елемент;
- **getLast() / peekLast()** - отримує останній елемент.

- **add(n, E)** - додає елемент E за індексом n;
- **remove(n)** - видаляє елемент за індексом n;

- **contains(E)** - перевірка на наявність елементу E;
- **size()** - повертає розмір списку;
- **isEmpty()** - Перевіряє, чи пустий список;
- **iterator()** - повертає ітератор для обходу елементів списку;
- **celar()** - очищує список.

Приклад:

```
LinkedList <String> list = new LinkedList<>();

list.addFirst("A"); // Додавання елемента "A" в початок списку
list.addFirst("B"); // Додавання елемента "B" в початок списку
list.addLast("C"); // Додавання елемента "C" в кінець списку
list.add("D"); // Додавання елемента "D" в кінець списку
list.add(1,"E"); // Додавання елемента "E" в список за індексом 1

for(String str:list){
   System.out.println(str);
}
```

Результат:

```
B
E
A
C
D
```

Додатково про методи класу LinkedList

contains(E) - *перевірка на наявність елементу E*

Цей метод використовується для перевірки, чи містить список **LinkedList** заданий елемент. Він повертає true, якщо елемент знайдено, і false, якщо ні.

```
LinkedList<String> colors = new LinkedList<>();
colors.add("червоний");
colors.add("зелений");
colors.add("синій");

if (colors.contains("червоний")) {
    System.out.println("Список містить червоний колір");
} else {
    System.out.println("Список не містить червоний колір");
}
```

Результат:

Список містить червоний колір

---

size() - *повертає розмір списку*

Цей метод повертає кількість елементів у списку **LinkedList**.

```
LinkedList<Integer> numbers = new LinkedList<>();
numbers.add(10);
numbers.add(20);
numbers.add(30);

int size = numbers.size();
System.out.println("Розмір списку: " + size);
```

Результат:

Розмір списку: 3

---

isEmpty() - *Перевіряє, чи пустий список*

Цей метод повертає true, якщо список **LinkedList** порожній, і false, якщо він містить хоча б один елемент.

```
LinkedList<String> names = new LinkedList<>();

if (names.isEmpty()) {
    System.out.println("Список порожній");
} else {
    System.out.println("Список не порожній");
}
```

Результат:

Список порожній

---

iterator() - *повертає ітератор для обходу елементів списку*

Цей метод повертає ітератор, який можна використовувати для обходу елементів списку **LinkedList**.

```
LinkedList<String> fruits = new LinkedList<>();
fruits.add("яблуко");
fruits.add("банан");
fruits.add("апельсин");

Iterator<String> iterator = fruits.iterator();
while (iterator.hasNext()) {
    String fruit = iterator.next();
    System.out.println(fruit);
}
```

Результат:

яблуко
банан
апельсин

Застосування Collections.LinkedList

Collections.LinkedList може використовуватися для реалізації різних *структур* даних та *алгоритмів*, наприклад:

• **Черги (FIFO)** - елементи додаються в кінець списку, видаляються з початку;
• **Стеки (LIFO)** - елементи додаються в кінець списку, видаляються з кінця;

• **Двоспрямовані списки** - **LinkedList** може використовуватися для представлення двоспрямованих списків, де можна обходити список у двох напрямках: вперед і назад. Це неможливо реалізувати ефективно за допомогою **ArrayList**;

• **Графи** - **LinkedList** може використовуватися для представлення вершин та ребер графа. Кожен вузол **LinkedList** може містити інформацію про вершину графа, а посилання можуть використовуватися для представлення зв'язків між вершинами.

Коли використовувати Collections.LinkedList?

Використовуйте **LinkedList**, коли вам потрібна *структур*а даних, що дозволяє:

• *Ефективно додавати та видаляти елементи в середині списку*;
• *Динамічно змінювати розмір списку*;
• *Реалізувати чергу (FIFO) або стек (LIFO)*;
• *Представляти двоспрямовані списки*.

Коли НЕ використовувати Collections.LinkedList?

Не використовуйте **LinkedList**, коли:

• *Вам потрібен швидкий доступ до елементів за індексом* (в цьому випадку **ArrayList** більш ефективний);
• *Пам'ять є обмеженим ресурсом* (**LinkedList** використовує більше пам'яті через додаткові посилання).

3.2. Алгоритм роботи навчального програмного забезпечення

Крок 0. У вікні програми виводиться вітальне повідомлення з текстом:

«**Вас вітає** навчальна програма-тренажер з теми **Collections.LinkedList**. Цей навчальний ресурс покликаний перевірити наявні та виробити нові знання та навички роботи з цією динамічною структурою даних.

Під час ознайомлення з програмою вам буде запропоновано:

- Теоретичні та практичні відомості - теоретична частина з розділами, що описують **LinkedList**, його операції, застосування та приклади коду;
- Тестові питання - оберіть правильні відповіді з декількох варіантів на питання, котрі стосуються основних концепцій та операцій **LinkedList**.

Як працювати з програмою:

- Прочитайте теоретичну частину, яка містить пояснення та приклади коду;
- Дайте відповідь на тестові запитання, щоб закріпити знання.

Важливо:

- Для кращого розуміння теми рекомендується мати базові знання з програмування на **Java**;
- Ви можете використовувати онлайн-ресурси та документацію для більш глибокого вивчення **LinkedList**.

Пам'ятайте:

- Ця програма є лише інструментом для навчання. Для більш глибокого розуміння теми рекомендується використовувати інші ресурси та практикуватися самостійно. »

Під повідомленням знаходиться кнопка з написом “Старт”, котра при натисканні переносить користувача на наступний крок.

Крок 1. Виводиться теорія: « **Collections.LinkedList** - це динамічна структура даних, що являє собою список елементів, пов'язаних один з одним за допомогою двоспрямованих посилань. **LinkedList** з'явився в **Java 1.2 (1998)** як альтернатива **ArrayList**. На відміну від **ArrayList**, який використовує фіксований масив для зберігання елементів, **LinkedList** не має фіксованого розміру та може динамічно розширюватися або скорочуватися під час додавання або видалення елементів. Він успадковує **AbstractSequentalList** і реалізує інтерфейси **List** та **Deque**(а через останній і **Queue**).

Структура:

- **Елементи -** кожен елемент містить значення та два посилання: next, яке вказує на наступний елемент в списку, і previous, яке вказує на попередній елемент;
- **Голова -** це посилання на перший елемент в списку;
- **Хвіст -** це посилання на останній елемент в списку.».

Перехід на наступний крок.

Крок 2. Виводиться завдання: «Що таке **Collections.LinkedList**?» і варіанти відповіді:

- Клас для роботи з фіксованими масивами даних;
- Динамічна структура даних, що являє собою список елементів, пов’язаних двоспрямованими посиланнями;
- Інтерфейс для роботи зі списками елементів;
- Абстрактний клас для роботи з послідовними списками даних.

Якщо обрано другий варіант, то перехід на наступний крок, в іншому випадку, з’являється одне з трьох повідомлень про помилку:

- Неправильна відповідь. Будь ласка, поверніться до теоретичної частини;
- Помилка. Рекомендую Вам краще ознайомитися з теоретичною частиною;
- Ви допустилися помилки. Рекомендую повернутися до теоретичного матеріалу;

Крок 3. Виводиться завдання: «Яка з наступних структур даних не має фіксованого розміру?» і варіанти відповіді:

- ArrayList;
- HashMap;
- LinkedList;
- Vector.

Якщо обрано третій варіант, то перехід на наступний крок, в іншому випадку, з’являється одне з трьох повідомлень про помилку:

- Неправильна відповідь. Будь ласка, поверніться до теоретичної частини;
- Помилка. Рекомендую Вам краще ознайомитися з теоретичною частиною;
- Ви допустилися помилки. Рекомендую повернутися до теоретичного матеріалу;

Крок 4. Виводиться теорія: «Через динамічність та простоту вставок і видалень він краще масивів. Також він має кілька недоліків, такі як неможливість прямого доступу до елементів, натомість нам потрібно починати з початку та переходити за посиланням, щоб дістатися до елемента, який ми шукаємо, що значно збільшує час роботи операцій отримання елементів.

Порівняння часу роботи операції отримання елементу з індексом 8000000 для LinkedList та ArrayList:

```
int result;
startTime = System.nanoTime();
result = list1.get(8000000);
finishTime=System.nanoTime();
System.out.println(result);
System.out.println("ArrayList get: "+(double)(finishTime-startTime)/1000000
+"ms");

startTime = System.nanoTime();
result = list2.get(8000000);
finishTime=System.nanoTime();
System.out.println(result);
System.out.println("LinkedList get: "+(double)(finishTime-startTime)/1000000
+"ms");
```

Результат:

ArrayList get:0.00591ms
LinkedList get:11.678236ms

Оскільки **LinkedList** діє як динамічний масив, і нам не потрібно вказувати розмір під час його створення, розмір списку автоматично збільшується, коли ми додаємо та видаляємо елементи. **LinkedList** реалізований з використанням структури даних двозв'язкового списку

Основна відмінність між звичайним зв'язаним списком та подвійним зв'язаним списком полягає в тому, що двозв'язаковий список містить додатковий покажчик, який зазвичай називають попереднім покажчиком. Ці додаткові посилання збільшують обсяг пам'яті, необхідний для зберігання **LinkedList**.».

Перехід на наступний крок.

Крок 5. Виводиться теорія: « Надалі, порівняння **LinkedList** буде переважно проводитися саме з **ArrayList**, адже він створювався саме як альтернатива йому.

Переваги Collections.LinkedList:

- **Ефективне додавання та видалення елементів в середині списку -** є основною перевагою, завдяки двоспрямованим посиланням, додавання та видалення елементів в середині списку **LinkedList** не потребує переміщення інших елементів, як це відбувається в **ArrayList**. Це робить **LinkedList** більш ефективним для цих операцій;

- **Не потрібне попереднє виділення пам'яті - LinkedList** не потребує попереднього виділення пам'яті для зберігання елементів, оскільки розмір списку може динамічно змінюватися. Це може бути корисно для програм, де розмір списку невідомий заздалегідь;

- **Можливість реалізації черги (FIFO) або стека (LIFO) - LinkedList** може використовуватися для реалізації черги (FIFO), де елементи видаляються в тому ж порядку, в якому вони були додані, або стека (LIFO), де елементи видаляються в зворотному порядку до їх додавання.

Недоліки Collections.LinkedList:

- **Менш ефективний доступ до елементів за індексом -** доступ до елементів **LinkedList** за індексом може бути менш ефективним, ніж в **ArrayList**, оскільки **LinkedList** не використовує фіксований масив. Це пов'язано з тим, що для доступу до елемента за індексом потрібно пройти через список, починаючи з першого елемента, доки не буде знайдений елемент з потрібним індексом;

- **Більше використання пам'яті - LinkedList** використовує більше пам'яті, ніж **ArrayList**, через додаткові посилання, необхідні для зв'язку елементів.».

Перехід на наступний крок.

Крок 6. Виводиться завдання: «Яку з наступних структур даних можна реалізувати за допомогою **LinkedList**?» і варіанти відповіді:

- Черга (FIFO);
- Стек (LIFO);
- Двоспрямований список;
- Все вищезазначене.

Якщо обрано четвертий варіант, то перехід на наступний крок, в іншому випадку, з’являється одне з трьох повідомлень про помилку:

- Неправильна відповідь. Будь ласка, поверніться до теоретичної частини;
- Помилка. Рекомендую Вам краще ознайомитися з теоретичною частиною;
- Ви допустилися помилки. Рекомендую повернутися до теоретичного матеріалу;

Крок 7. Виводиться завдання: «Яка з наступних характеристик є недоліком **LinkedList**?» і варіанти відповіді:

- Ефективне додавання та видалення елементів в середині списку;
- Не потребує попереднього виділення мап’яті;
- Менш ефективний доступ до елементів за індексом;
- Можливість реалізації черги (FIFO) або стека (LIFO).

Якщо обрано третій варіант, то перехід на наступний крок, в іншому випадку, з’являється одне з трьох повідомлень про помилку:

- Неправильна відповідь. Будь ласка, поверніться до теоретичної частини;
- Помилка. Рекомендую Вам краще ознайомитися з теоретичною частиною;
- Ви допустилися помилки. Рекомендую повернутися до теоретичного матеріалу;

Крок 8. Виводиться завдання: «Яка основна перевага **LinkedList** над **ArrayList**?» і варіанти відповіді:

- Швидший доступ до елементів за індексом;
- Більш ефективне додавання та видалення елементів в середині списку;
- Менше використання мап’яті;
- Фіксований розмір списку.

Якщо обрано другий варіант, то перехід на наступний крок, в іншому випадку, з’являється одне з трьох повідомлень про помилку:

- Неправильна відповідь. Будь ласка, поверніться до теоретичної частини;
- Помилка. Рекомендую Вам краще ознайомитися з теоретичною частиною;
- Ви допустилися помилки. Рекомендую повернутися до теоретичного матеріалу;

Крок 9. Виводиться теорія: « **Клас LinkedList має наступні конструктори:**

- **LinkedList()** - котрий створює порожній список;
- **LinkedList(Collection<? extends E> col)** - створює список, в який додає всі елементи колекції col.

LinkedList містить всі методи, котрі містять в собі інтерфейси List, Queue, Deque, основні з них:

- **addFirst() / offerFirst()** - додає елемент в початок списку;
- **addLast() / offerLast()** - додає елемент в кінець списку;
- **removeFirst() / pollFirst()** - видаляє перший елемент з початку списку;
- **removeLast() / pollLast()** - видаляє останній елемент з кінця списку;
- **getFirst() / peekFirst()** - отримує перший елемент;
- **getLast() / peekLast()** - отримує останній елемент.

- **add(n, E)** - додає елемент E за індексом n;

- **remove(n)** - видаляє елемент за індексом n;
- **contains(E)** - перевірка на наявність елементу E;
- **size()** - повертає розмір списку;
- **isEmpty()** - Перевіряє, чи пустий список;
- **iterator()** - повертає ітератор для обходу елементів списку;
- **celar()** - очищує список.

Приклад:

```
LinkedList <String> list = new LinkedList<>();

list.addFirst("A"); // Додавання елемента "A" в початок списку
list.addFirst("B"); // Додавання елемента "B" в початок списку
list.addLast("C"); // Додавання елемента "C" в кінець списку
list.add("D"); // Додавання елемента "D" в кінець списку
list.add(1,"E"); // Додавання елемента "E" в список за індексом 1

for(String str:list){
   System.out.println(str);
}
```

Результат:

```
B
E
A
C
D
```

».

Перехід на наступний крок.

Крок 10. Виводиться завдання: «Який конструктор **LinkedList** створює порожній список?» і варіанти відповіді:

- LinkedList();
- LinkedList<String>();

- LinkedList(Collection<? extends E> col);
- LinkedList.create().

Якщо обрано перший варіант, то перехід на наступний крок, в іншому випадку, з’являється одне з трьох повідомлень про помилку:

- Неправильна відповідь. Будь ласка, поверніться до теоретичної частини;
- Помилка. Рекомендую Вам краще ознайомитися з теоретичною частиною;
- Ви допустилися помилки. Рекомендую повернутися до теоретичного матеріалу;

Крок 11. Виводиться завдання: «Який метод **LinkedList** використовується для додавання елемента в початок списку?» і варіанти відповіді:

- addFirst();
- addLast();
- add();
- insert.

Якщо обрано перший варіант, то перехід на наступний крок, в іншому випадку, з’являється одне з трьох повідомлень про помилку:

- Неправильна відповідь. Будь ласка, поверніться до теоретичної частини;
- Помилка. Рекомендую Вам краще ознайомитися з теоретичною частиною;
- Ви допустилися помилки. Рекомендую повернутися до теоретичного матеріалу;

Крок 12. Виводиться завдання: «Який метод **LinkedList** використовується для отримання першого елемента списку?» і варіанти відповіді:

- getLast();
- get();
- getFirst();
- peek().

Якщо обрано третій варіант, то перехід на наступний крок, в іншому випадку, з’являється одне з трьох повідомлень про помилку:

- Неправильна відповідь. Будь ласка, поверніться до теоретичної частини;
- Помилка. Рекомендую Вам краще ознайомитися з теоретичною частиною;
- Ви допустилися помилки. Рекомендую повернутися до теоретичного матеріалу;

Крок 13. Виводиться завдання: «Який результат виведе наступний код?

```
LinkedList<Integer> list = new LinkedList<>();
list.add(5);
list.add(3);
list.add(0, 1);
list.addFirst(2);
list.addLast(4);

System.out.println(list.getFirst());
System.out.println(list.getLast());
System.out.println(list.get(2));
```

» і варіанти відповіді:

- 5 4 3;
- 2 4 1;
- 1 5 3;
- 2 5 1.

Якщо обрано другий варіант, то перехід на наступний крок, в іншому випадку, з’являється одне з трьох повідомлень про помилку:

- Неправильна відповідь. Будь ласка, поверніться до теоретичної частини;
- Помилка. Рекомендую Вам краще ознайомитися з теоретичною частиною;
- Ви допустилися помилки. Рекомендую повернутися до теоретичного матеріалу;

Крок 14. Виводиться теорія: «
contains(E) - *перевірка на наявність елементу E*
Цей метод використовується для перевірки, чи містить список **LinkedList** заданий елемент. Він повертає true, якщо елемент знайдено, і false, якщо ні.

```
LinkedList<String> colors = new LinkedList<>();
colors.add("червоний");
colors.add("зелений");
colors.add("синій");

if (colors.contains("червоний")) {
   System.out.println("Список містить червоний колір");
} else {
   System.out.println("Список не містить червоний колір");
}
```

Результат:
Список містить червоний колір

---

size() - *повертає розмір списку*
Цей метод повертає кількість елементів у списку **LinkedList**.

```
LinkedList<Integer> numbers = new LinkedList<>();
numbers.add(10);
numbers.add(20);
```

```
numbers.add(30);

int size = numbers.size();
System.out.println("Розмір списку: " + size);
```

Результат:

```
Розмір списку: 3
```

---

isEmpty() - *Перевіряє, чи пустий список*

Цей метод повертає true, якщо список **LinkedList** порожній, і false, якщо він містить хоча б один елемент.

```
LinkedList<String> names = new LinkedList<>();

if (names.isEmpty()) {
    System.out.println("Список порожній");
} else {
    System.out.println("Список не порожній");
}
```

Результат:

```
Список порожній
```

---

iterator() - *повертає ітератор для обходу елементів списку*

Цей метод повертає ітератор, який можна використовувати для обходу елементів списку **LinkedList**.

```
LinkedList<String> fruits = new LinkedList<>();
fruits.add("яблуко");
fruits.add("банан");
fruits.add("апельсин");

Iterator<String> iterator = fruits.iterator();
while (iterator.hasNext()) {
    String fruit = iterator.next();
    System.out.println(fruit);
}
```

Результат:
яблуко
банан
апельсин
».
Перехід на наступний крок.

Крок 15. Виводиться завдання: «Додайте елемент "манго" в кінець списку fruits.

```
LinkedList<String> fruits = new LinkedList<>();
fruits.add("яблуко");
fruits.add("банан");
fruits.add("апельсин");
```

» і варіанти відповіді:

- fruits.addLast(“манго”);
- fruits.removeFirst();
- fruits.add(1,”манго”);
- fruits.clear().

Якщо обрано перший варіант, то перехід на наступний крок, в іншому випадку, з’являється одне з трьох повідомлень про помилку:

- Неправильна відповідь. Будь ласка, поверніться до теоретичної частини;
- Помилка. Рекомендую Вам краще ознайомитися з теоретичною частиною;
- Ви допустилися помилки. Рекомендую повернутися до теоретичного матеріалу;

Крок 16. Виводиться завдання: «Перевірте, скільки елементів має список fruits.

```
LinkedList<String> fruits = new LinkedList<>();
```

```
fruits.add("яблуко");
fruits.add("банан");
fruits.add("апельсин");
```

» і варіанти відповіді:

- System.out.println(fruits);
- Int size = fruits.size();
  System.out.println(“Розмір списку: ” + size);
- fruits.add(1,”полуниця”);
- System.out.println(list.getLast()).

Якщо обрано другий варіант, то перехід на наступний крок, в іншому випадку, з’являється одне з трьох повідомлень про помилку:

- Неправильна відповідь. Будь ласка, поверніться до теоретичної частини;
- Помилка. Рекомендую Вам краще ознайомитися з теоретичною частиною;
- Ви допустилися помилки. Рекомендую повернутися до теоретичного матеріалу;

Крок 17. Виводиться завдання: «Очистіть список fruits

```
LinkedList<String> fruits = new LinkedList<>();
fruits.add("яблуко");
fruits.add("банан");
fruits.add("апельсин");
```

» і варіанти відповіді:

- System.out.println(fruits);
- fruits.removeFirst();
- fruits.clear();
- System.out.println(list.getLast()).

Якщо обрано третій варіант, то перехід на наступний крок, в іншому випадку, з’являється одне з трьох повідомлень про помилку:

- Неправильна відповідь. Будь ласка, поверніться до теоретичної частини;
- Помилка. Рекомендую Вам краще ознайомитися з теоретичною частиною;
- Ви допустилися помилки. Рекомендую повернутися до теоретичного матеріалу;

Крок 18. Виводиться теорія: «**Collections.LinkedList** може використовуватися для реалізації різних *структур* даних та *алгоритмів*, наприклад:

• Черги (FIFO) - елементи додаються в кінець списку, видаляються з початку;
• Стеки (LIFO) - елементи додаються в кінець списку, видаляються з кінця;

• Двоспрямовані списки - **LinkedList** може використовуватися для представлення двоспрямованих списків, де можна обходити список у двох напрямках: вперед і назад. Це неможливо реалізувати ефективно за допомогою **ArrayList**;

• Графи - **LinkedList** може використовуватися для представлення вершин та ребер графа. Кожен вузол **LinkedList** може містити інформацію про вершину графа, а посилання можуть використовуватися для представлення зв'язків між вершинами.

Коли використовувати Collections.LinkedList?
Використовуйте **LinkedList**, коли вам потрібна *структур*а даних, що дозволяє:

• *Ефективно додавати та видаляти елементи в середині списку*;

• *Динамічно змінювати розмір списку*;

• *Реалізувати чергу (FIFO) або стек (LIFO)*;

• *Представляти двоспрямовані списки*.

Коли НЕ використовувати Collections.LinkedList?

Не використовуйте **LinkedList**, коли:

• *Вам потрібен швидкий доступ до елементів за індексом* (в цьому випадку **ArrayList** більш ефективний);

• *Пам'ять є обмеженим ресурсом* (**LinkedList** використовує більше пам'яті через додаткові посилання).».

Перехід на наступний крок.

Крок 19. Виводиться завдання: «Коли слід використовувати **LinkedList**?» і варіанти відповіді:

- Коли потрібен швидкий доступ до елементів за індексом;
- Коли потрібна динамічна структура даних, що легко змінює розмір;
- Коли потрібні великі списки;
- Коли пам’ять є обмеженим ресурсом.

Якщо обрано другий варіант, то перехід на наступний крок, в іншому випадку, з’являється одне з трьох повідомлень про помилку:

- Неправильна відповідь. Будь ласка, поверніться до теоретичної частини;
- Помилка. Рекомендую Вам краще ознайомитися з теоретичною частиною;
- Ви допустилися помилки. Рекомендую повернутися до теоретичного матеріалу;

Крок 20. Виводиться завдання: «Коли НЕ слід використовувати **LinkedList**?» і варіанти відповіді:

- Коли потрібна динамічна структура даних, що легко змінює розмір;
- Коли потрібна реалізація черги (FIFO) або стека (LIFO);
- Коли потрібен швидкий доступ до елементів за індексом;
- Коли потрібен невідомий заздалегідь розмір списку.

Якщо обрано третій варіант, то перехід на наступний крок, в іншому випадку, з'являється одне з трьох повідомлень про помилку:

- Неправильна відповідь. Будь ласка, поверніться до теоретичної частини;
- Помилка. Рекомендую Вам краще ознайомитися з теоретичною частиною;
- Ви допустилися помилки. Рекомендую повернутися до теоретичного матеріалу;

Крок 21. У вікні програми виводиться прощальне повідомлення з текстом:

«**Вітаємо** з успішним проходженням тренажера з теми **Collections.LinkedList**. Під час його проходження Ви здобули такі знання та вміння як:

• Основні знання про **Collections.LinkedList**, його структуру та особливості;

• Слабкі та сильні сторони цієї структури даних, відносно її альтернатив;

• Дізналися, для реалізації яких структур потрібен **LinkedList**;

• Знання, як і коли краще використовувати **LinkedList**, а коли краще уникати його використовування;

• Освоїли методи та конструктори потрібні для роботи з **LinkedList**;

• Навчилися працювати з кодом на **Java**, котрий реалізує методи **LinkedList**

Сподіваюся цей тренажер був корисним для вас та допоміг освоїти та закріпити весь матеріал, котрий був поданий у рамках цієї програми.».

Під повідомленням знаходиться кнопка з написом “Кінець”, котра при натисканні переносить користувача на **Крок 0**.

3.3. Блок-схема алгоритму роботи навчального програмного забезпечення

На рисунку 3.2 зображена блок-схема алгоритму роботи тренажера.

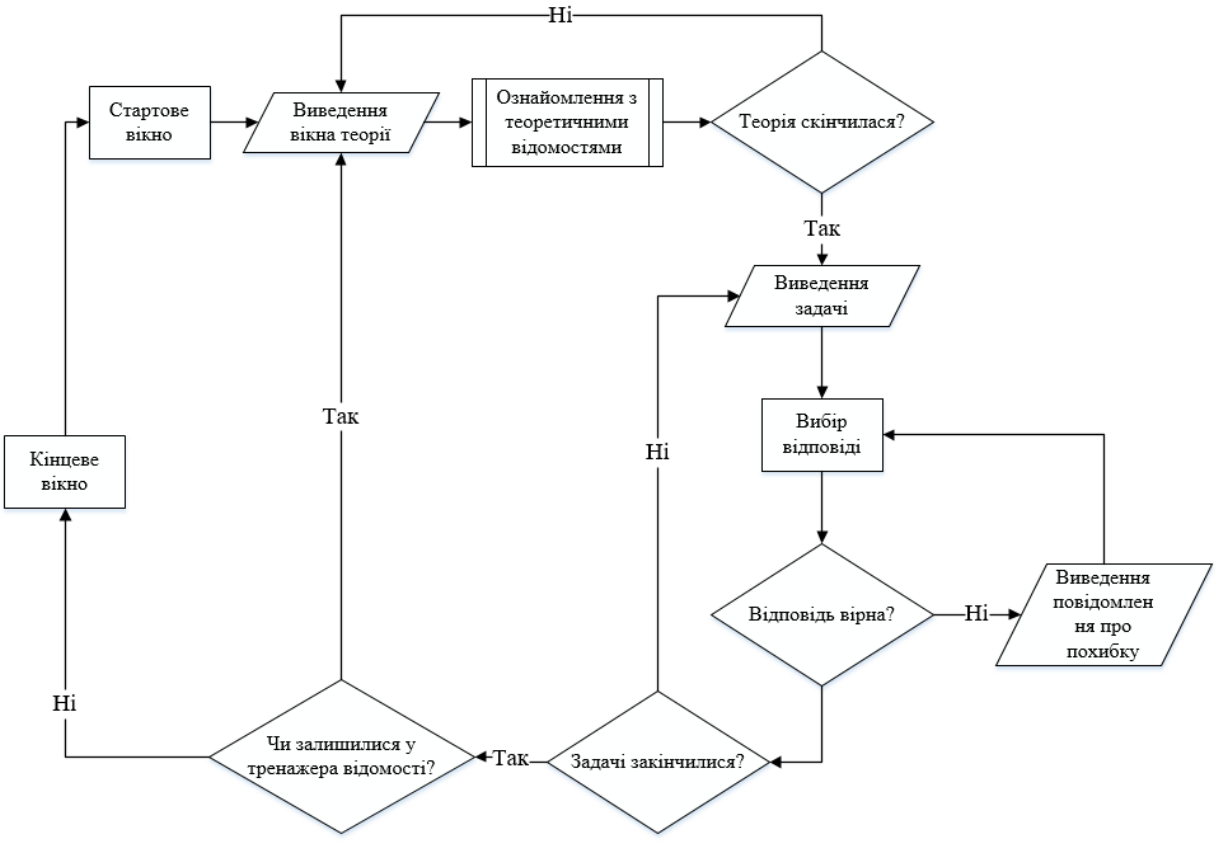

Рисунок 3.2 – Повна блок-схема роботи тренажера

4. ПРАКТИЧНА ЧАСТИНА

4.1. Обґрунтування вибору програмних засобів

C# - це об'єктно-орієнтована мова програмування, потужна і гнучка мова, яка може використовуватися для розробки широкого спектру програмного забезпечення, включаючи десктопні програми. Розроблена компанією Microsoft. Вона була випущена в 2002 році як наступник мови Visual Basic.NET. C# є мовою, яка компілюється в машинний код, що означає, що вона виконується швидко і ефективно. Вона також підтримує паралельне програмування, що дозволяє розробляти програми, які можуть використовувати кілька процесорів одночасно.

Однією з ключових особливостей C# є її об'єктно-орієнтована парадигма. Це означає, що програми C# складаються з об'єктів, які взаємодіють один з одним. Об'єкти в C# мають властивості, методи та події, які описують їх поведінку.

C# - це популярна мова програмування, яка широко використовується в розробці програмного забезпечення. Вона є потужною і гнучкою мовою, яка може використовуватися для розробки широкого спектру програм.

Щоб створювати інтерфейс програм на C#, можна використовувати Windows Forms

Windows Forms - це платформа, яка дозволяє розробляти інтерфейси користувача для програм, які працюють на Windows. Для використання Windows Forms потрібно додати до проекту посилання на бібліотеку System.Windows.Forms.

4.2.Опис процесу програмної реалізації

Підключаємо бібліотеки, оголошуємо змінні якими будемо оперувати надалі та ініціалізуємо запуск тренажера:

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace WindowsFormsApp1
{
    public partial class Form1 : Form
    {
        byte step;
        byte maxStep = 22;
        byte correctAnswer;
        byte RID;
        byte x = 1;
        byte x1;
        byte num;

        public Form1()
        {
            InitializeComponent();
            Start_window();
          }
```

Позиції та розміри елементів інтерфейсу тренажера прописано в методі CalculateStart:

```
private void CalculateStart()
        {
            button2.Size = new Size(button1.Size.Width, button1.Size.Height);
            button3.Size = new Size(button1.Size.Width, button1.Size.Height);

            int a = (button1.Size.Width / 2);
            int b = (ClientSize.Width / 3);
            int c = ((ClientSize.Width / 2) - a);

            int d = button1.Size.Height + 55;
            int f = radioButton1.Size.Height + d + 30;

            head2.Location = new Point((ClientSize.Width - head2.Size.Width)
/ 2, -2);

            radioButton1.Size = new Size(ClientSize.Width / 2, 80);
            radioButton2.Size = radioButton1.Size;
```

```
            radioButton3.Size = new Size(ClientSize.Width / 2 - 42, 80);
            radioButton4.Size = radioButton3.Size;
            radioButton1.Location = new Point(12, ClientSize.Height - (f +
110));
            radioButton2.Location = new Point(12, ClientSize.Height - f);
            radioButton3.Location = new Point(radioButton1.Size.Width + 30,
ClientSize.Height - (f + 110));
            radioButton4.Location = new Point(radioButton1.Size.Width + 30,
ClientSize.Height - f);

            button1.Location = new Point(b - a, ClientSize.Height - d);
            button2.Location = new Point(c, ClientSize.Height - d);
            button3.Location = new Point(b * 2 - a, ClientSize.Height - d);
            FALSE.Location = new Point((ClientSize.Width - FALSE.Size.Width)
/ 2, ClientSize.Height - (FALSE.Size.Height + 5));

            pictureBox1.Image = Properties.Resources.image1;
            pictureBox1.Location = new Point((ClientSize.Width / 2) -
(pictureBox1.Width / 2), ClientSize.Height - d - pictureBox1.Height - 30);

            if (x == 1)
            {
                RichTextBox1.Size = new Size(ClientSize.Width - 15,
ClientSize.Height - (d + button1.Size.Height + 15));
            }
            else
            {
                RichTextBox1.Size = new Size(ClientSize.Width - 15,
ClientSize.Height - (f + 170));
            }
              }
```

Стилізація тексту виконується методами BoldText, ColorText, UnderlineText, ItalicText, RegularText:

- Метод BoldText відповідає за написання напівтовстого тексту:

```
private void BoldText(RichTextBox box, string text)
        {
            int i = box.Text.IndexOf(text);
            while (i >= 0)
            {
                box.Select(i, text.Length);
                box.SelectionFont = new Font(box.Font, FontStyle.Bold);
                i = box.Text.IndexOf(text, i + text.Length);
            }
        }
```

- Метод ColorText відповідає за колір тексту:

```
        private void ColorText(RichTextBox box, string text, Color color)
        {
            int i = box.Text.IndexOf(text);
            while (i >= 0)
            {
                box.Select(i, text.Length);
```

```
            box.SelectionColor = color;
            i = box.Text.IndexOf(text, i + text.Length);
        }
    }
```

- Метод UnderlineText відповідає за підкреслений тексту:

```
private void UnderlineText(RichTextBox box, string text)
{
    int i = box.Text.IndexOf(text);
    while (i >= 0)
    {
        box.Select(i, text.Length);
        box.SelectionFont = new Font(box.Font, FontStyle.Underline);
        i = box.Text.IndexOf(text, i + text.Length);
    }
}
```

- Метод ItalicText відповідає за текст курсивом:

```
private void ItalicText(RichTextBox box, string text)
{
    int i = box.Text.IndexOf(text);
    while (i >= 0)
    {
        box.Select(i, text.Length);
        box.SelectionFont = new Font(box.Font, FontStyle.Italic);
        i = box.Text.IndexOf(text, i + text.Length);
    }
}
```

- Метод RegularText відповідає за звичайний текст:

```
private void RegularText(RichTextBox box, string text)
{
    int i = box.Text.IndexOf(text);
    while (i >= 0)
    {
        box.Select(i, text.Length);
        box.SelectionFont = new Font(box.Font, FontStyle.Regular);
        i = box.Text.IndexOf(text, i + text.Length);
    }
      }
```

Метод Start_window виводить початкову (вітальну) сторінку:

```
private void Start_window()
        {
            CalculateStart();
            button2.Text = "Розпочати";
            head2.Visible = true;
            head2.Text = "Тренажер з теми Collections.LinkedList";
            button1.Visible = false;
            button2.Visible = true;
            button3.Visible = false;
```

```
            FALSE.Visible = false;
            pictureBox1.Visible = false;
            RichTextBox1.Text = "\tВас вітає навчальна програма-тренажер з теми
Collections.LinkedList. Цей навчальний ресурс покликаний перевірити наявні та виробити
нові знання та навички роботи з цією динамічною структурою даних. \n\n Під час
ознайомлення з програмою вам буде запропоновано: \n\n• Теоретичні та практичні відомості
- теоретична частина з розділами, що описують LinkedList, його операції, застосування та
приклади коду;\n• Тестові питання - оберіть правильні відповіді з декількох варіантів на
питання, котрі стосуються основних концепцій та операцій LinkedList.\n\n Як працювати з
програмою:\n\n• Прочитайте теоретичну частину, яка містить пояснення та приклади коду;\n•
Дайте відповідь на тестові запитання, щоб закріпити знання.\n\n Важливо:\n\n• Для кращого
розуміння теми рекомендується мати базові знання з програмування на Java;\n• Ви можете
використовувати онлайн-ресурси та документацію для більш глибокого вивчення
LinkedList.\n\nПам'ятайте:\n\n• Ця програма є лише інструментом для навчання. Для більш
глибокого розуміння теми рекомендується використовувати інші ресурси та практикуватися
самостійно.";

            BoldText(RichTextBox1, "Вас вітає");
            BoldText(RichTextBox1, "Під час ознайомлення з програмою вам буде
запропоновано:");
            BoldText(RichTextBox1, "Як працювати з програмою:");
            BoldText(RichTextBox1, "Важливо:");
            BoldText(RichTextBox1, "Пам'ятайте:");
            UnderlineText(RichTextBox1, "програма є лише інструментом для навчання");

            Switch(0);
               }
```

Функції, що виводять вміст кожної сторінки тренажера:

```
private void Step(double Step)
        {
            RichTextBox1.Location = new Point(9, 41);

            if (Step == maxStep)
            {
                Start_window();
                step = 0;
            }
            if (Step == 0)
            {
                Start_window();
            }


            switch (Step)
            {

                case 1:
                    num = 1;
                    pictureBox1.Visible = true;
                    pictureBox1.Image = Properties.Resources.image1;
                    head2.Text = "Що таке Collections.LinkedList?";
```

Виведення тексту, що з’являється на першій, теоретичній, сторінці навчальної програми:

```
                    RichTextBox1.Text = "\tCollections.LinkedList - це
динамічна структура даних, що являє собою список елементів, пов'язаних один з
```

```
одним за допомогою двоспрямованих посилань. LinkedList з'явився в Java 1.2 (1998) як альтернатива ArrayList. На відміну від ArrayList, який використовує фіксований масив для зберігання елементів, LinkedList не має фіксованого розміру та може динамічно розширюватися або скорочуватися під час додавання або видалення елементів. Він успадковує AbstractSequentalList і реалізує інтерфейси List та Deque(а через останній і Queue).\n\nСтруктура:\n\n\t• Елементи - кожен елемент містить значення та два посилання: next, яке вказує на наступний елемент в списку, і \n\t\t\t  previous, яке вказує на попередній елемент;\n\t• Голова - це посилання на перший елемент в списку;\n\t• Хвіст - це посилання на останній елемент в списку.";
```

Параметри для стилізації тексту, що був виведений у вікно програми вище:

```
                    TForm();

                    BoldText(RichTextBox1, "Елементи -");
                    BoldText(RichTextBox1, "Структура:");
                    BoldText(RichTextBox1, "Голова -");
                    BoldText(RichTextBox1, "Хвіст -");
                    BoldText(RichTextBox1, "Java 1.2 (1998)");
                    break;

                case 2:
                    pictureBox1.Visible = false;
```

Виведення завдання, що з’являється на першій, тестовій, сторінці навчальної програми:

```
                    RichTextBox1.Text = "Що таке Collections.LinkedList?";
```

Варіанти відповідей до нього:

```
                    radioButton1.Text = "Клас для роботи з фіксованими масивами даних;";
                    radioButton2.Text = "Динамічна структура даних, що являє собою список елементів, пов'язаних двоспрямованими посиланнями;";
                    radioButton3.Text = "Інтерфейс для роботи зі списками елементів;";
                    radioButton4.Text = "Абстрактний клас для роботи з послідовними списками даних.";

                    PForm();

                    correctAnswer = 2;
                    break;

                case 3:
```

Виведення завдання, що з’являється на другій, тестовій, сторінці навчальної програми:

```
                    RichTextBox1.Text = "Яка з наступних структур даних не має фіксованого розміру?";
```

Варіанти відповідей до нього:

```
                radioButton1.Text = "ArrayList;";
                radioButton2.Text = "HashMap;";
                radioButton3.Text = "LinkedList;";
                radioButton4.Text = "Vector.";

                PForm();

                correctAnswer = 3;
                break;

            case 4:

                head2.Text = "Що таке Collections.LinkedList?";
```

Виведення тексту, що з’являється на другій, теоретичній, сторінці навчальної програми:

```
                RichTextBox1.Text = "\tЧерез динамічність та простоту
вставок і видалень він краще масивів. Також він має кілька недоліків, такі як
неможливість прямого доступу до елементів, натомість нам потрібно починати з
початку та переходити за посиланням, щоб дістатися до елемента, який ми
шукаємо, що значно збільшує час роботи операцій отримання
елементів.\n\n\nПорівняння часу роботи операції отримання елементу з індексом
8000000 для LinkedList та ArrayList:\n\nint result;\nstartTime =
System.nanoTime();\nresult =
list1.get(8000000);\nfinishTime=System.nanoTime();\nSystem.out.println(result
);\nSystem.out.println(\"ArrayList get: \"+(double)(finishTime-
startTime)/1000000 +\"ms\");\n\nstartTime = System.nanoTime();\nresult =
list2.get(8000000);\nfinishTime=System.nanoTime();\nSystem.out.println(result
);\nSystem.out.println(\"LinkedList get: \"+(double)(finishTime-
startTime)/1000000 +\"ms\");\n\n\tРезультат:\nArrayList
get:0.00591ms\nLinkedList get:11.678236ms\n\n\n\tОскільки LinkedList діє як
динамічний масив, і нам не потрібно вказувати розмір під час його створення,
розмір списку автоматично збільшується, коли ми додаємо та видаляємо
елементи. LinkedList реалізований з використанням структури даних
двозв'язкового списку\n\tОсновна відмінність між звичайним зв'язаним списком
та подвійним зв'язаним списком полягає в тому, що двозв'язаковий список
містить додатковий покажчик, який зазвичай називають попереднім покажчиком.
Ці додаткові посилання збільшують обсяг пам'яті, необхідний для зберігання
LinkedList.";
```

Параметри для стилізації тексту, що був виведений у вікно програми вище:

```
                TForm();

                BoldText(RichTextBox1, "Порівняння часу роботи операції
отримання елементу з індексом 8000000 для LinkedList та ArrayList:");

                ItalicText(RichTextBox1, "nanoTime");

                ColorText(RichTextBox1, "0", Color.Blue);
                ColorText(RichTextBox1, "1", Color.Blue);
```

```
                    ColorText(RichTextBox1, "8000000", Color.Blue);
                    ColorText(RichTextBox1, "println", Color.Black);
                    ColorText(RichTextBox1, "list1", Color.Black);
                    ColorText(RichTextBox1, "\"ArrayList get: \"",
Color.Green);
                    ColorText(RichTextBox1, "\"ms\"", Color.Green);
                    ColorText(RichTextBox1, "\"LinkedList get: \"",
Color.Green);
                    ColorText(RichTextBox1, "ArrayList get:0.00591ms",
Color.Black);
                    ColorText(RichTextBox1, "LinkedList get:11.678236ms",
Color.Black);

                    RegularText(RichTextBox1, "List<Integer>");
                    RegularText(RichTextBox1, "ArrayList<>();");
                    RegularText(RichTextBox1, "LinkedList<>();");
                    RegularText(RichTextBox1, "ArrayList get:");
                    RegularText(RichTextBox1, "LinkedList get:");
                    RegularText(RichTextBox1, "ArrayList get:0.00591ms");
                    RegularText(RichTextBox1, "LinkedList get:11.678236ms");

                    break;

                case 5:
                    head2.Text = "Переваги та недоліки";
```

Виведення тексту, що з’являється на третій, теоретичній, сторінці навчальної програми:

```
                    RichTextBox1.Text = "\tНадалі, порівняння LinkedList буде
переважно проводитися саме з ArrayList, адже він створювався саме як
альтернатива йому.\n\n\tПереваги Collections.LinkedList:\n\n• Ефективне
додавання та видалення елементів в середині списку - є основною перевагою,
завдяки двоспрямованим посиланням, додавання та видалення елементів в
середині списку LinkedList не потребує переміщення інших елементів, як це
відбувається в ArrayList. Це робить LinkedList більш ефективним для цих
операцій;\n\n• Не потрібне попереднє виділення пам'яті - LinkedList не
потребує попереднього виділення пам'яті для зберігання елементів, оскільки
розмір списку може динамічно змінюватися. Це може бути корисно для програм,
де розмір списку невідомий заздалегідь;\n\n• Можливість реалізації черги
(FIFO) або стека (LIFO) - LinkedList може використовуватися для реалізації
черги (FIFO), де елементи видаляються в тому ж порядку, в якому вони були
додані, або стека (LIFO), де елементи видаляються в зворотному порядку до їх
додавання.\n\n\tНедоліки Collections.LinkedList:\n\n• Менш ефективний доступ
до елементів за індексом - доступ до елементів LinkedList за індексом може
бути менш ефективним, ніж в ArrayList, оскільки LinkedList не використовує
фіксований масив. Це пов'язано з тим, що для доступу до елемента за індексом
потрібно пройти через список, починаючи з першого елемента, доки не буде
знайдений елемент з потрібним індексом;\n\n• Більше використання пам'яті -
LinkedList використовує більше пам'яті, ніж ArrayList, через додаткові
посилання, необхідні для зв'язку елементів.";
```

Параметри для стилізації тексту, що був виведений у вікно програми вище:

```
                    BoldText(RichTextBox1, "Переваги
Collections.LinkedList:");
                    BoldText(RichTextBox1, "Ефективне додавання та видалення
елементів в середині списку -");
                    BoldText(RichTextBox1, "Не потрібне попереднє виділення
пам'яті -");
                    BoldText(RichTextBox1, "Можливість реалізації черги
(FIFO) або стека (LIFO) -");
                    BoldText(RichTextBox1, "Недоліки
Collections.LinkedList:");
                    BoldText(RichTextBox1, "Менш ефективний доступ до
елементів за індексом -");
                    BoldText(RichTextBox1, "Більше використання пам'яті -");

                    TForm();
                    break;


                case 6:
```

Виведення завдання, що з’являється на третій, тестовій, сторінці навчальної програми:

```
                    RichTextBox1.Text = "Яку з наступних структур даних можна
реалізувати за допомогою LinkedList?";
```

Варіанти відповідей до нього:

```
                    radioButton1.Text = "Черга (FIFO);";
                    radioButton2.Text = "Стек (LIFO);";
                    radioButton3.Text = "Двоспрямований список;";
                    radioButton4.Text = "Все вищезазначене.";

                    PForm();

                    correctAnswer = 4;

                    break;

                case 7:
```

Виведення завдання, що з’являється на четвертій, тестовій, сторінці навчальної програми:

```
                    RichTextBox1.Text = "Яка з наступних характеристик є
недоліком LinkedList?";
```

Варіанти відповідей до нього:

```
                    radioButton1.Text = "Ефективне додавання та видалення
елементів в середині списку;";
                    radioButton2.Text = "Не потребує попереднього виділення
пам'яті;";
```

```
                    radioButton3.Text = "Менш ефективний доступ до елементів
за індексом;";
                    radioButton4.Text = "Можливість реалізації черги (FIFO)
або стека (LIFO).";

                    PForm();

                    correctAnswer = 3;

                    break;

                case 8:
```

Виведення завдання, що з’являється на п’ятій, тестовій, сторінці навчальної програми:

```
                    RichTextBox1.Text = "Яка основна перевага LinkedList над
ArrayList?";
```

Варіанти відповідей до нього:

```
                    radioButton1.Text = "Швидший доступ до елементів за
індексом;";
                    radioButton2.Text = "Більш ефективне додавання та
видалення елементів в середині списку;";
                    radioButton3.Text = "Менше використання пам'яті;";
                    radioButton4.Text = "Фіксований розмір списку.";

                    PForm();

                    correctAnswer = 2;

                    break;

                case 9:
```

Виведення тексту, що з’являється на четвертій, теоретичній, сторінці навчальної програми:

```
                    head2.Text = "Конструктори, методи та робота з ними";
                    RichTextBox1.Text = "\tКлас LinkedList має наступні
конструктори:\n\n• LinkedList() - котрий створює порожній список;\n•
LinkedList(Collection<? extends E> col) - створює список, в який додає всі
елементи колекції col.\n\n\tLinkedList містить всі методи, котрі містять в
собі інтерфейси List, Queue, Deque, основні з них:\n\n• addFirst() /
offerFirst() - додає елемент в початок списку;\n• addLast() / offerLast() -
додає елемент в кінець списку;\n• removeFirst() / pollFirst() - видаляє
перший елемент з початку списку;\n• removeLast() / pollLast() - видаляє
останній елемент з кінця списку;\n• getFirst() / peekFirst() - отримує перший
елемент;\n• getLast() / peekLast() - отримує останній елемент.\n\n• add(n, E)
- додає елемент E за індексом n;\n• remove(n) - видаляє елемент за індексом
n;\n• contains(E) - перевірка на наявність елементу E;\n• size() - повертає
розмір списку;\n• isEmpty() - Перевіряє, чи пустий список;\n• iterator() -
повертає ітератор для обходу елементів списку;\n• celar() - очищує
список.\n\n\tПриклад:\n\n\nLinkedList <String> list = new
```

```
LinkedList<>();\n\nlist.addFirst(\"A\"); // Додавання елемента \"A\" в
початок списку\nlist.addFirst(\"B\"); // Додавання елемента \"B\" в початок
списку\nlist.addLast(\"C\"); // Додавання елемента \"C\" в кінець
списку\nlist.add(\"D\"); // Додавання елемента \"D\" в кінець
списку\nlist.add(1,\"E\"); // Додавання елемента \"E\" в список за індексом
1\n\nfor(String
str:list){\n\tSystem.out.println(str);\n}\n\n\tРезультат:\n\nB\nE\nA\nC\nD";
```

Параметри для стилізації тексту, що був виведений у вікно програми вище:

```
                    TForm();
                    BoldText(RichTextBox1, "Клас LinkedList має наступні
конструктори:");
                    BoldText(RichTextBox1, "LinkedList()");
                    BoldText(RichTextBox1, "LinkedList(Collection<? extends
E> col)");
                    BoldText(RichTextBox1, "LinkedList містить всі методи,
котрі містять в собі інтерфейси List, Queue, Deque, основні з них:");
                    BoldText(RichTextBox1, "addFirst() / offerFirst()");
                    BoldText(RichTextBox1, "addLast() / offerLast()");
                    BoldText(RichTextBox1, "removeFirst() / pollFirst()");
                    BoldText(RichTextBox1, "removeLast() / pollLast()");
                    BoldText(RichTextBox1, "getFirst() / peekFirst()");
                    BoldText(RichTextBox1, "getLast() / peekLast()");
                    BoldText(RichTextBox1, "add(n, E)");
                    BoldText(RichTextBox1, "contains(E)");
                    BoldText(RichTextBox1, "remove(n)");
                    BoldText(RichTextBox1, "size()");
                    BoldText(RichTextBox1, "isEmpty()");
                    BoldText(RichTextBox1, "iterator()");
                    BoldText(RichTextBox1, "celar()");
                    BoldText(RichTextBox1, "Приклад:");

                    RegularText(RichTextBox1, ".isEmpty()");
                    RegularText(RichTextBox1, ".iterator()");
                    RegularText(RichTextBox1, ".size();");
                    RegularText(RichTextBox1, "LinkedList <String> ");
                    RegularText(RichTextBox1, "LinkedList<>()");

                    ColorText(RichTextBox1, "1", Color.Blue);
                    ColorText(RichTextBox1,"// Додавання елемента \"A\" в
початок списку",Color.DarkGreen);
                    ColorText(RichTextBox1, "// Додавання елемента \"B\" в
початок списку", Color.DarkGreen);
                    ColorText(RichTextBox1, "// Додавання елемента \"C\" в
кінець списку", Color.DarkGreen);
                    ColorText(RichTextBox1, "// Додавання елемента \"D\" в
кінець списку", Color.DarkGreen);
                    ColorText(RichTextBox1, "// Додавання елемента \"E\" в
список за індексом 1", Color.DarkGreen);
                    ColorText(RichTextBox1, "\"A\"", Color.Green);
                    ColorText(RichTextBox1, "\"B\"", Color.Green);
                    ColorText(RichTextBox1, "\"C\"", Color.Green);
                    ColorText(RichTextBox1, "\"D\"", Color.Green);
                    ColorText(RichTextBox1, "\"E\"", Color.Green);
                    break;
```

```
            case 10:
```

Виведення завдання, що з’являється на шостій, тестовій, сторінці навчальної програми:

```
                RichTextBox1.Text = "Який конструктор LinkedList створює порожній список?";
```

Варіанти відповідей до нього:

```
                radioButton1.Text = "LinkedList();";
                radioButton2.Text = "LinkedList<String>();";
                radioButton3.Text = "LinkedList(Collection<? extends E> col);";
                radioButton4.Text = "LinkedList.create().";

                PForm();

                correctAnswer = 1;

                break;

            case 11:
```

Виведення завдання, що з’являється на сьомій, тестовій, сторінці навчальної програми:

```
                RichTextBox1.Text = "Який метод LinkedList використовується для додавання елемента в початок списку?";
```

Варіанти відповідей до нього:

```
                radioButton1.Text = "addFirst();";
                radioButton2.Text = "addLast();";
                radioButton3.Text = "add();";
                radioButton4.Text = "insert.";

                PForm();

                correctAnswer = 1;

                break;

            case 12:
```

Виведення завдання, що з’являється на восьмій, тестовій, сторінці навчальної програми:

```
                RichTextBox1.Text = "Який метод LinkedList використовується для отримання першого елемента списку?";
```

Варіанти відповідей до нього:

```
                radioButton1.Text = "getLast();";
```

```
                radioButton2.Text = "get();";
                radioButton3.Text = "getFirst();";
                radioButton4.Text = "peek().";

                PForm();

                correctAnswer = 3;

                break;

            case 13:
```

Виведення завдання, що з’являється на дев’ятій, тестовій, сторінці навчальної програми:

```
                RichTextBox1.Text = "Який результат виведе наступний
код?\n\nLinkedList<Integer> list = new
LinkedList<>();\nlist.add(5);\nlist.add(3);\nlist.add(0,
1);\nlist.addFirst(2);\nlist.addLast(4);\n\nSystem.out.println(list.getFirst(
));\nSystem.out.println(list.getLast());\nSystem.out.println(list.get(2));";
```

Стилізація тексту та варіанти відповідей до нього:

```
                radioButton1.Text = "5 4 3;";
                radioButton2.Text = "2 4 1;";
                radioButton3.Text = "1 5 3;";
                radioButton4.Text = "2 5 1.";

                ColorText(RichTextBox1, "0", Color.Blue);
                ColorText(RichTextBox1, "1", Color.Blue);
                ColorText(RichTextBox1, "2", Color.Blue);
                ColorText(RichTextBox1, "3", Color.Blue);
                ColorText(RichTextBox1, "4", Color.Blue);
                ColorText(RichTextBox1, "5", Color.Blue);

                PForm();
                RegularText(RichTextBox1, "LinkedList");
                correctAnswer = 2;

                break;

            case 14:
```

Виведення тексту, що з’являється на п’ятій, теоретичній, сторінці навчальної програми:

```
                head2.Text = "Додатково про методи класу LinkedList";
                RichTextBox1.Text = "\tcontains(E) - перевірка на
наявність елементу E\nЦей метод використовується для перевірки, чи містить
список LinkedList заданий елемент. Він повертає true, якщо елемент знайдено,
і false, якщо ні.\n\n\nLinkedList<String> colors = new
LinkedList<>();\ncolors.add(\"червоний\");\ncolors.add(\"зелений\");\ncolors.
add(\"синій\");\n\nif (colors.contains(\"червоний\"))
{\n\tSystem.out.println(\"Список містить червоний колір\");\n} else
```

```
{\n\tSystem.out.println(\"Список не містить червоний
колір\");\n}\n\n\tРезультат:\nСписок містить червоний колір\n\n----------
----------------------------------------------------------------
\n\n\tsize() - повертає розмір списку\nЦей метод повертає кількість елементів
у списку LinkedList.\n\n\nLinkedList<Integer> numbers = new
LinkedList<>();\nnumbers.add(10);\nnumbers.add(20);\nnumbers.add(30);\n\nint
size = numbers.size();\nSystem.out.println(\"Розмір списку: \" +
size);\n\n\tРезультат:\nРозмір списку: 3\n\n--------------------------------
-----------------------------------------------\n\n\tisEmpty() -
Перевіряє, чи пустий список\nЦей метод повертає true, якщо список LinkedList
порожній, і false, якщо він містить хоча б один
елемент.\n\n\nLinkedList<String> names = new LinkedList<>();\n\nif
(names.isEmpty()) {\n\tSystem.out.println(\"Список порожній\");\n} else
{\n\tSystem.out.println(\"Список не порожній\");\n}\n\n\tРезультат:\nСписок
порожній\n\n----------------------------------------------------------------
-----------------\n\n\titerator() - повертає ітератор для обходу елементів
списку\nЦей метод повертає ітератор, який можна використовувати для обходу
елементів списку LinkedList.\n\n\nLinkedList<String> fruits = new
LinkedList<>();\nfruits.add(\"яблуко\");\nfruits.add(\"банан\");\nfruits.add(
\"апельсин\");\n\nIterator<String> iterator = fruits.iterator();\nwhile
(iterator.hasNext()) {\n\tString fruit =
iterator.next();\n\tSystem.out.println(fruit);\n}\n\n\tРезультат:\nяблуко\nба
нан\nапельсин";
```

Параметри для стилізації тексту, що був виведений у вікно програми вище:

```
                    TForm();
                    BoldText(RichTextBox1, "contains(E)");
                    BoldText(RichTextBox1, "size()");
                    BoldText(RichTextBox1, "isEmpty()");
                    BoldText(RichTextBox1, "iterator()");

                    ItalicText(RichTextBox1, "перевірка на наявність елементу
E");
                    ItalicText(RichTextBox1, "повертає розмір списку");
                    ItalicText(RichTextBox1, "Перевіряє, чи пустий список");
                    ItalicText(RichTextBox1, "повертає ітератор для обходу
елементів списку");

                    RegularText(RichTextBox1, "LinkedList<>();");
                    RegularText(RichTextBox1, "LinkedList<String> colors =");
                    RegularText(RichTextBox1, "LinkedList<Integer> numbers
=");
                    RegularText(RichTextBox1, "LinkedList<String> names =");
                    RegularText(RichTextBox1, "LinkedList<String> fruits =");
                    RegularText(RichTextBox1, ".isEmpty()");
                    RegularText(RichTextBox1, ".iterator()");


                    ColorText(RichTextBox1, "\"Список містить червоний
колір\"", Color.Green);
                    ColorText(RichTextBox1, "\"Список не містить червоний
колір\"", Color.Green);
                    ColorText(RichTextBox1, "\"червоний\"", Color.Green);
                    ColorText(RichTextBox1, "\"зелений\"", Color.Green);
                    ColorText(RichTextBox1, "\"синій\"", Color.Green);
```

```
                    ColorText(RichTextBox1, "\"Розмір списку: \"",
Color.Green);
                    ColorText(RichTextBox1, "\"Список порожній\"",
Color.Green);
                    ColorText(RichTextBox1, "\"Список не порожній\"",
Color.Green);
                    ColorText(RichTextBox1, "10", Color.Blue);
                    ColorText(RichTextBox1, "20", Color.Blue);
                    ColorText(RichTextBox1, "30", Color.Blue);
                    break;

                case 15:
```

Виведення завдання, що з’являється на десятій, тестовій, сторінці навчальної програми:

```
                    RichTextBox1.Text = "\tДодайте елемент \"манго\" в кінець
списку fruits.\n\nLinkedList<String> fruits = new
LinkedList<>();\nfruits.add(\"яблуко\");\nfruits.add(\"банан\");\nfruits.add(
\"апельсин\");";
```

Варіанти відповідей до нього:

```
                    radioButton1.Text = "fruits.addLast(\"манго\");";
                    radioButton2.Text = "fruits.removeFirst();";
                    radioButton3.Text = "fruits.add(1,\"манго\");";
                    radioButton4.Text = "fruits.clear();";

                    PForm();
                    correctAnswer = 1;
                    break;

                case 16:
```

Виведення завдання, що з’являється на одинадцятій, тестовій, сторінці навчальної програми:

```
                    RichTextBox1.Text = "\tПеревірте, скільки елементів має
список fruits\n\nLinkedList<String> fruits = new
LinkedList<>();\nfruits.add(\"яблуко\");\nfruits.add(\"банан\");\nfruits.add(
\"апельсин\");";
```

Варіанти відповідей до нього:

```
                    radioButton1.Text = "System.out.println(fruits);";
                    radioButton2.Text = "int size =
fruits.size();\nSystem.out.println(\"Розмір списку: \" + size)";
                    radioButton3.Text = "fruits.add(1,\"полуниця\")";
                    radioButton4.Text = "System.out.println(list.getLast())";

                    PForm();
                    correctAnswer = 2;
                    break;

                case 17:
```

Виведення завдання, що з’являється на дванадцятій, тестовій, сторінці навчальної програми:

```
                    RichTextBox1.Text = "\tОчистіть список
fruits\n\nLinkedList<String> fruits = new
LinkedList<>();\nfruits.add(\"яблуко\");\nfruits.add(\"банан\");\nfruits.add(
\"апельсин\");";
```

Варіанти відповідей до нього:

```
                    radioButton1.Text = "System.out.println(fruits);";
                    radioButton2.Text = "fruits.removeFirst();";
                    radioButton3.Text = "fruits.clear();";
                    radioButton4.Text =
"System.out.println(list.getLast());";

                    PForm();
                    correctAnswer = 3;
                    break;

                case 18:
```

Виведення тексту, що з’являється на шостій, теоретичній, сторінці навчальної програми:

```
                    head2.Text = "Застосування Collections.LinkedList";
                    RichTextBox1.Text = "\tCollections.LinkedList може
використовуватися для реалізації різних структур даних та алгоритмів,
наприклад:\n\n• Черги (FIFO) - елементи додаються в кінець списку,
видаляються з початку;\n• Стеки (LIFO) - елементи додаються в кінець списку,
видаляються з кінця;\n\n• Двоспрямовані списки - LinkedList може
використовуватися для представлення двоспрямованих списків, де можна обходити
список у двох напрямках: вперед і назад. Це неможливо реалізувати ефективно
за допомогою ArrayList;\n\n• Графи - LinkedList може використовуватися для
представлення вершин та ребер графа. Кожен вузол LinkedList може містити
інформацію про вершину графа, а посилання можуть використовуватися для
представлення зв'язків між вершинами.\n\n\tКоли використовувати
Collections.LinkedList?\nВикористовуйте LinkedList, коли вам потрібна
структура даних, що дозволяє:\n\n• Ефективно додавати та видаляти елементи в
середині списку;\n• Динамічно змінювати розмір списку;\n• Реалізувати чергу
(FIFO) або стек (LIFO);\n• Представляти двоспрямовані списки.\n\n\tКоли НЕ
використовувати Collections.LinkedList?\nНе використовуйте LinkedList,
коли:\n\n• Вам потрібен швидкий доступ до елементів за індексом (в цьому
випадку ArrayList більш ефективний);\n• Пам'ять є обмеженим ресурсом
(LinkedList використовує більше пам'яті через додаткові посилання).";
```

Параметри для стилізації тексту, що був виведений у вікно програми вище:

```
                    TForm();
                    BoldText(RichTextBox1, "Двоспрямовані списки");
                    BoldText(RichTextBox1, "Графи");
                    BoldText(RichTextBox1, "Черги (FIFO)");
                    BoldText(RichTextBox1, "Стеки (LIFO)");
```

```
                    BoldText(RichTextBox1, "Коли використовувати
Collections.LinkedList?");
                    BoldText(RichTextBox1, "Коли НЕ використовувати
Collections.LinkedList?");

                    ItalicText(RichTextBox1, "структур");
                    ItalicText(RichTextBox1, "алгоритмів");
                    ItalicText(RichTextBox1, "Ефективно додавати та видаляти
елементи в середині списку");
                    ItalicText(RichTextBox1, "Динамічно змінювати розмір
списку");
                    ItalicText(RichTextBox1, "Реалізувати чергу (FIFO) або
стек (LIFO)");
                    ItalicText(RichTextBox1, "Представляти двоспрямовані
списки");
                    ItalicText(RichTextBox1, "Вам потрібен швидкий доступ до
елементів за індексом ");
                    ItalicText(RichTextBox1, "Пам'ять є обмеженим ресурсом
");
                    break;
                case 19:
```

Виведення завдання, що з’являється на тринадцятій, тестовій, сторінці навчальної програми:

```
                    RichTextBox1.Text = "\tКоли слід використовувати
LinkedList?";
```

Варіанти відповідей до нього:

```
                    radioButton1.Text = "Коли потрібен швидкий доступ до
елементів за індексом;";
                    radioButton2.Text = "Коли потрібна динамічна структура
даних, що легко змінює розмір;";
                    radioButton3.Text = "Коли потрібні великі списки;";
                    radioButton4.Text = "Коли пам'ять є обмеженим ресурсом;";

                    PForm();


                    correctAnswer = 2;
                    break;
                case 20:
```

Виведення завдання, що з’являється на чотирнадцятій, тестовій, сторінці навчальної програми:

```
                    RichTextBox1.Text = "\tКоли НЕ слід використовувати
LinkedList?";
```

Варіанти відповідей до нього:

```
                    radioButton1.Text = "Коли потрібна динамічна структура
даних, що легко змінює розмір;";
```

```
                    radioButton2.Text = "Коли потрібна реалізація черги
(FIFO) або стека (LIFO);";
                    radioButton3.Text = "Коли потрібен швидкий доступ до
елементів за індексом;";
                    radioButton4.Text = "Коли потрібен невідомий заздалегідь
розмір списку;";

                    PForm();

                    correctAnswer = 3;
                    break;
                case 21:
```

Виведення тексту, що з’являється на кінцевій, сторінці навчальної програми:

```
                    head2.Text = "Завершення";
                    RichTextBox1.Text = "\tВітаємо з успішним проходженням
тренажера з теми Collections.LinkedList. Під час його проходження Ви здобули
такі знання та вміння як:\n\n• Основні знання про Collections.LinkedList,
його структуру та особливості;\n\n• Слабкі та сильні сторони цієї структури
даних, відносно її альтернатив;\n\n• Дізналися, для реалізації яких структур
потрібен LinkedList;\n\n• Знання, як і коли краще використовувати LinkedList,
а коли краще уникати його використовування;\n\n• Освоїли методи та
конструктори потрібні для роботи з LinkedList;\n\n• Навчилися працювати з
кодом на Java, котрий реалізує методи LinkedList\n\n\tСподіваюся цей тренажер
був корисним для вас та допоміг освоїти та закріпити весь матеріал, котрий
був поданий у рамках цієї програми. ";

                    TForm();
                    Final_window();
                    BoldText(RichTextBox1, "Вітаємо");

                    break;

            }
        }
```

Next_button(ClickEvent) виконується при натисканні кнопок “Розпочати”, “Кінець” та “Далі” для відображення наступного кроку:

```
public void Next_button(object sender, EventArgs e)
        {
            x1 = x;
            RichTextBox1.Clear();
            CalculateStart();
            step++;
            Step(step);
            if (x1 == x && x1 == 0)
            {
                num++;
```

```
            head2.Text = "Тестування з пройденого матеріалу №" + num;
        }
        if (x1 != x && x1 == 1)
        {
            head2.Text = "Тестування з пройденого матеріалу №" + num;
        }
        if (x1 != x && x1 == 0)
        {
            num++;

        }
          }
```

Back_button (ClickEvent). виконується при натисканні кнопки "Назад" для відображення попереднього кроку.

```
public void Back_button(object sender, EventArgs e)
        {
            x1 = x;
            RichTextBox1.Clear();
            CalculateStart();
            step--;
            Step(step);

            if (x1 == x && x1 == 0)
            {
                num--;
                head2.Text = "Тестування з пройденого матеріалу №" + num;
            }
            if (x1 != x && x1 == 1)
            {
                num--;
                head2.Text = "Тестування з пройденого матеріалу №" + num;
            }
        }
```

Перевірка відповіді відбувається за допомогою CheckAnswer(ClickEvent) при натисканні кнопки “Далі”:

```
private void CheckAnswer(object sender, EventArgs e)
        {
            if (radioButton1.Checked)
            {
                RID = 1;
            }

            if (radioButton2.Checked)
            {
```

```
            RID = 2;
        }

        if (radioButton3.Checked)
        {
            RID = 3;
        }

        if (radioButton4.Checked)
        {
            RID = 4;
        }



        if (RID == correctAnswer)
        {
            x1 = x;
            step++;
            Step(step);
            if (x1 == x && x1 == 0)
            {
                num++;
                head2.Text = "Тестування з пройденого матеріалу №" + num;
            }
            if (x1 != x && x1 == 1)
            {
                head2.Text = "Тестування з пройденого матеріалу №" + num;
            }
            if (x1 != x && x1 == 0)
            {
                num++;

            }
        }
        else
        {
            Random();
            FALSE.Location = new Point((ClientSize.Width -
FALSE.Size.Width) / 2, ClientSize.Height - (FALSE.Size.Height + 5));
        }
          }
```

Через метод Random випадково обирається один з варіантів напису для повідомлення про помилкову відповідь:

```
private void Random()
        {
            FALSE.Visible = true;
            Random random = new Random();
            int randomNumber = random.Next(1, 4);

            if (randomNumber == 1)
```

```
            {
                FALSE.Text = "Неправильна відповідь.\nБудь ласка, поверніться
до теоретичної частини";
            }
            if (randomNumber == 2)
            {
                FALSE.Text = "Помилка.\nРекомендую Вам краще ознайомитися з
теоретичною частиною";
            }
            if (randomNumber == 3)
            {
                FALSE.Text = "Ви допустилися помилки.\nРекомендую повернутися
до теоретичного матеріалу";
            }
        }
```

На останньому кроці виконується метод Final_window, котрий демонструє “Прощальне" вікно програми:

```
  private void Final_window()
        {
            CalculateStart();
            button2.Text = "Кінець";
            head2.Visible = true;
            button1.Visible = false;
            button2.Visible = true;
            button3.Visible = false;
            FALSE.Visible = false;

            Switch(0);
        }
```

4.3. Інструкція по використанню навчального програмного забезпечення

Стартове вікно містить тему тренажера, текст з інструкціями з користування програмою та кнопку “Розпочати” (Рис 4.1), котра при

натисканні відображає, теоретичне вікно програми (Рис 4.2).

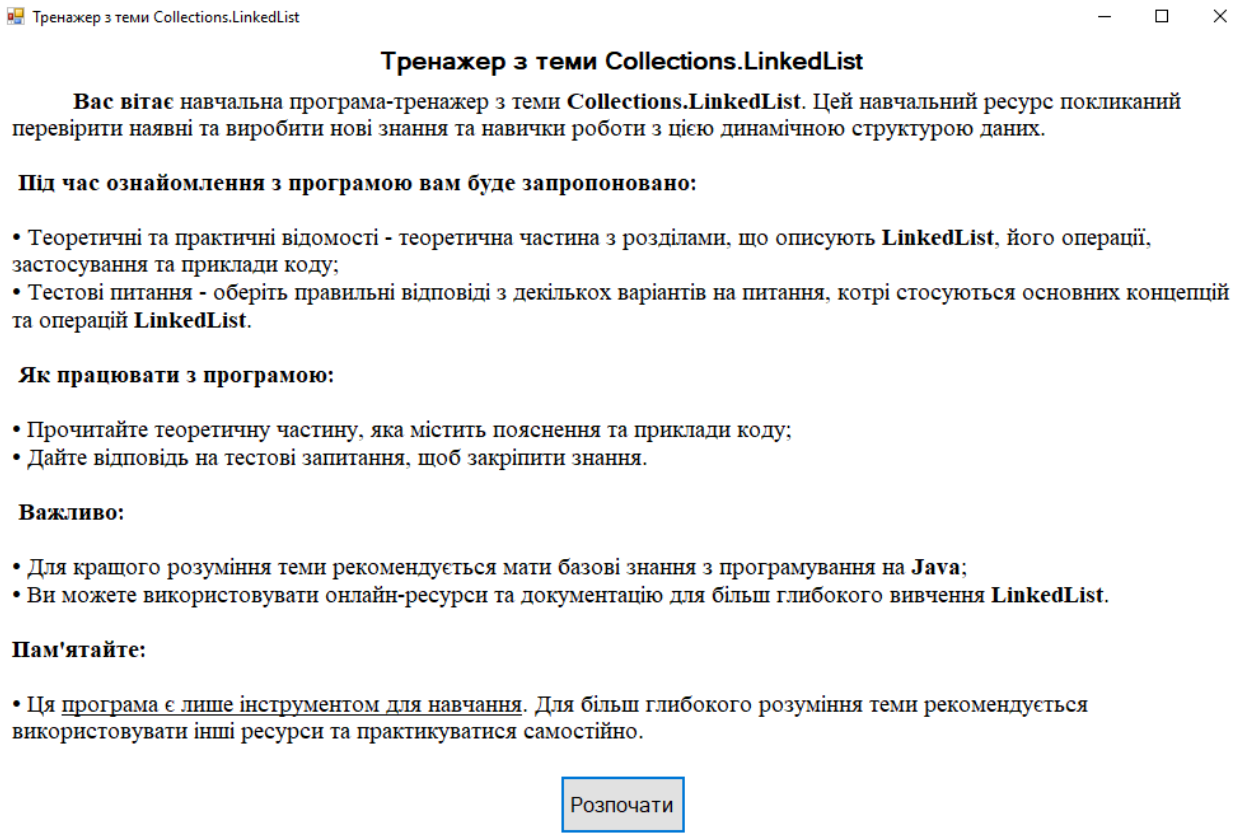

Рисунок 4.1 – стартове вікно тренажера

Теоретичне вікно містить назву теми, пояснення до теми та кнопки “Далі” і “Назад” (Рис 4.2). При натисканні кнопки “Далі” тема змінюється на наступну або з’являється тестове вікно програми (Рис 4.3) на якому знаходиться тестове завдання з чотирма варіантами відповіді.

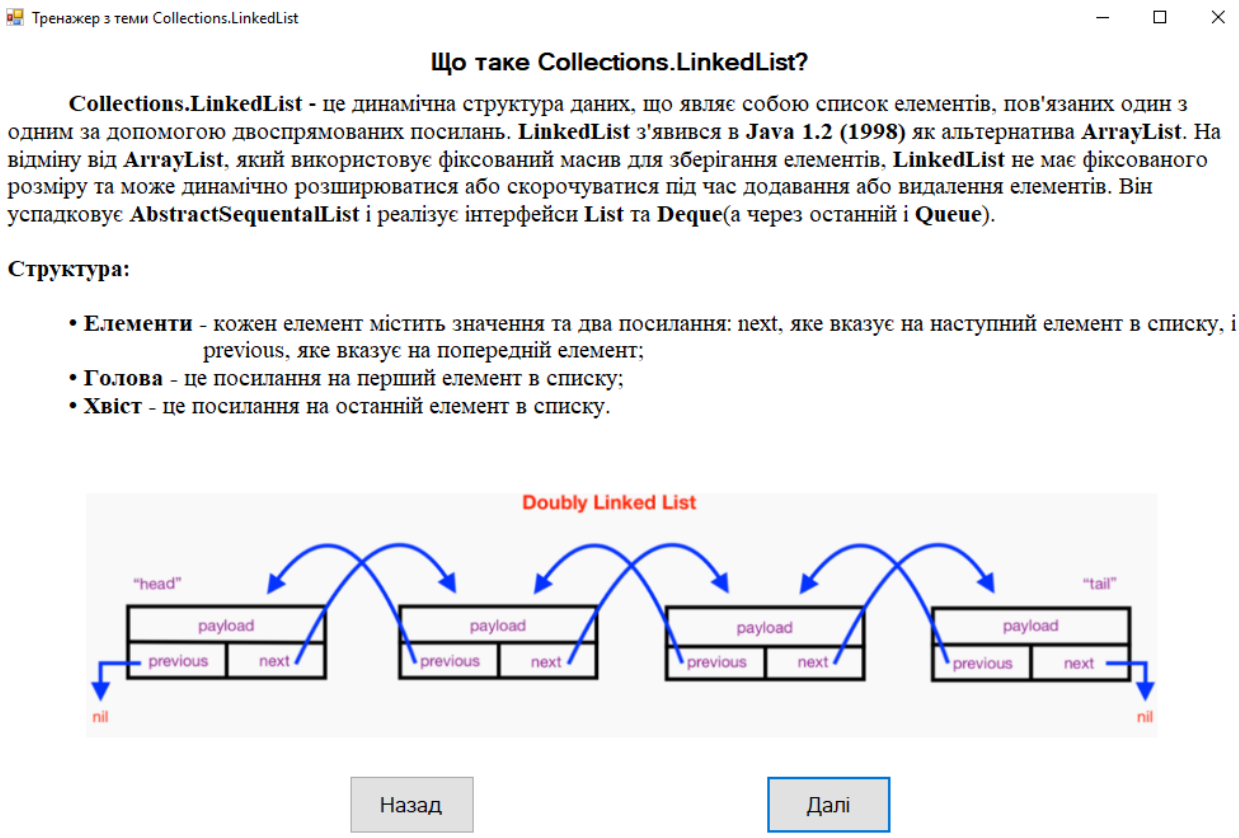

Рисунок 4.2 – теоретичне вікно тренажера

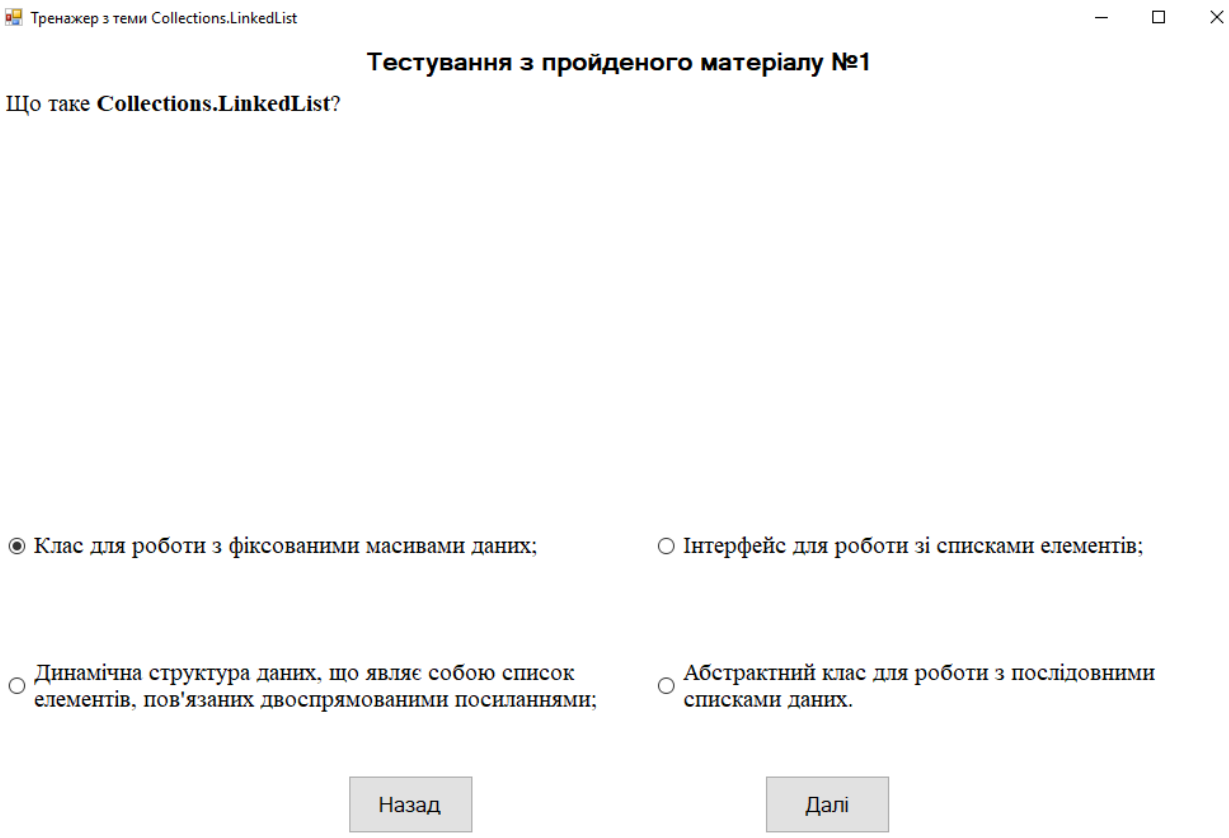

Рисунок 4.3 – тестове вікно тренажера

Тестове вікно тренажера має чотири кнопки для вибору відповіді та відображає завдання. Обравши одну з відповідей користувач повинен натиснути кнопку “Далі”, щоб перевірити правильність відповіді та перейти до наступного завдання. Якщо обрана відповідь буде хибною користувач побачить одне з трьох можливих повідомлень (Рисунок 4.4 – 4.6), а при обиранні вірної відповіді перейде до наступної задачі.

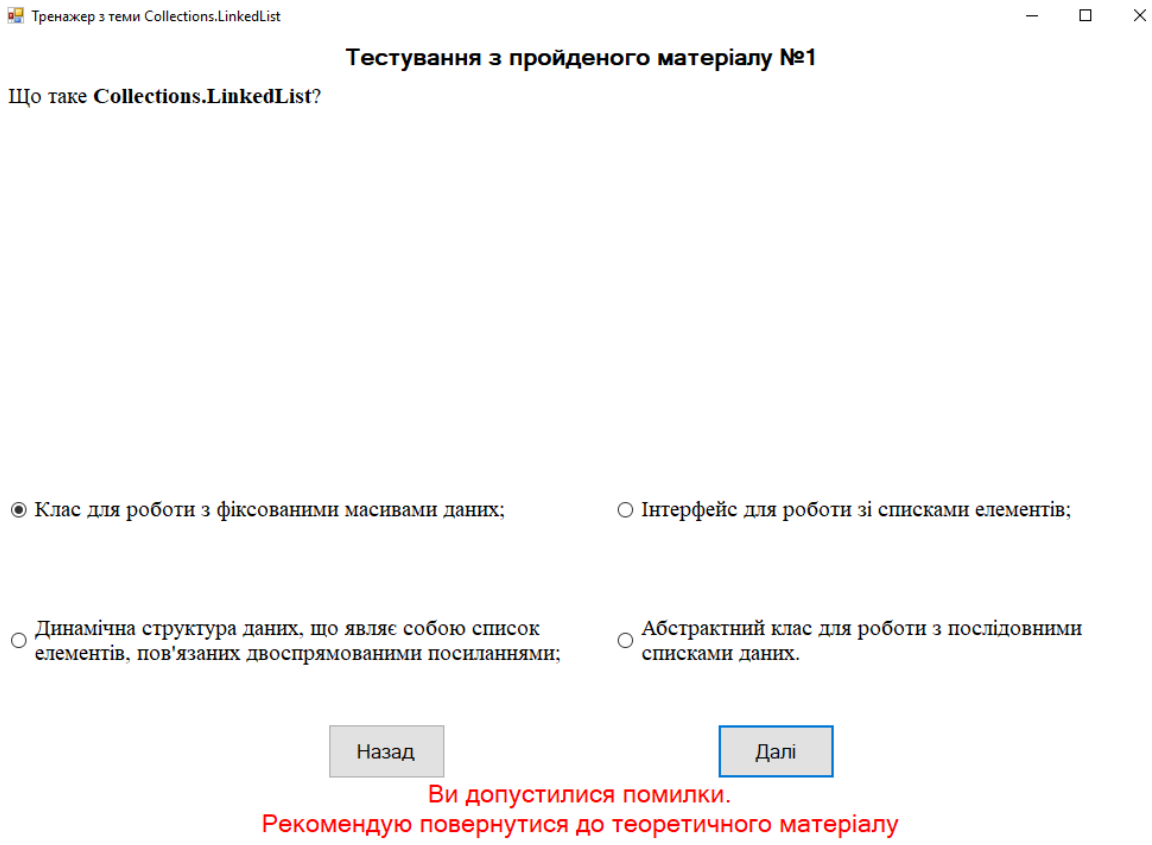

Рисунок 4.4 – Хибна відповідь викликає повідомлення №1 у вікні тренажера

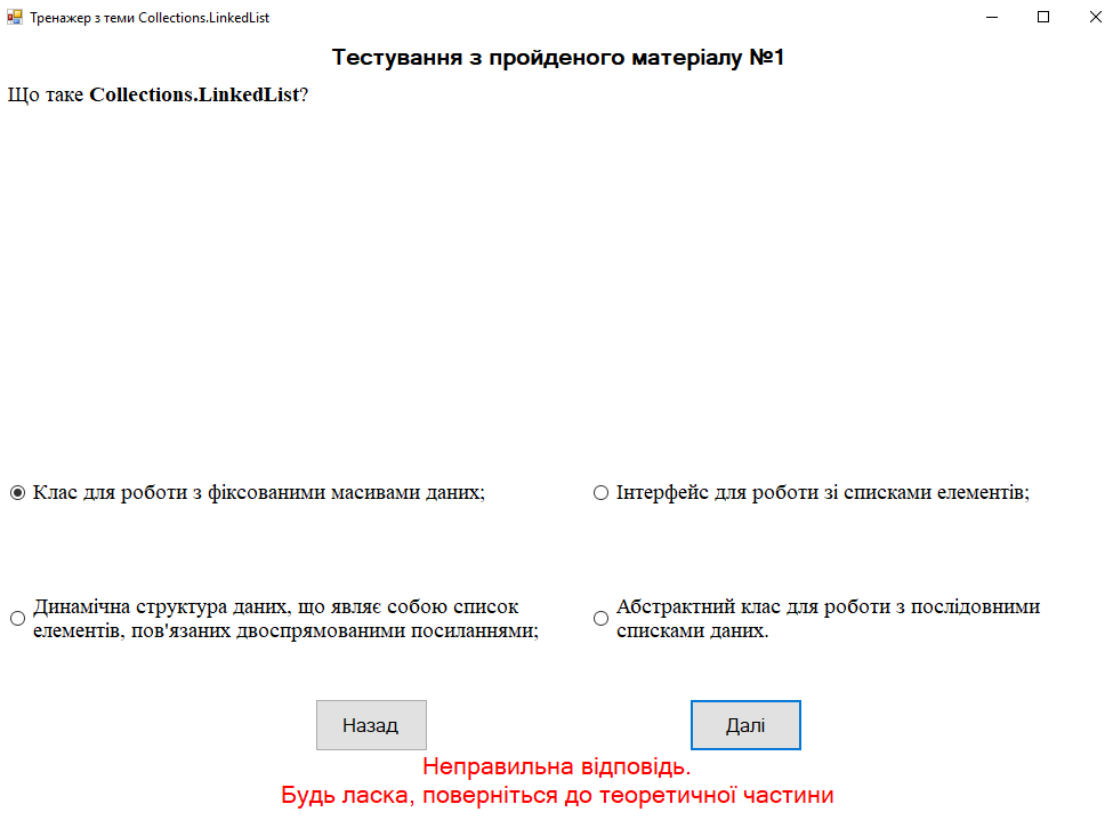

Рисунок 4.5 – Хибна відповідь викликає повідомлення №2 у вікні тренажера

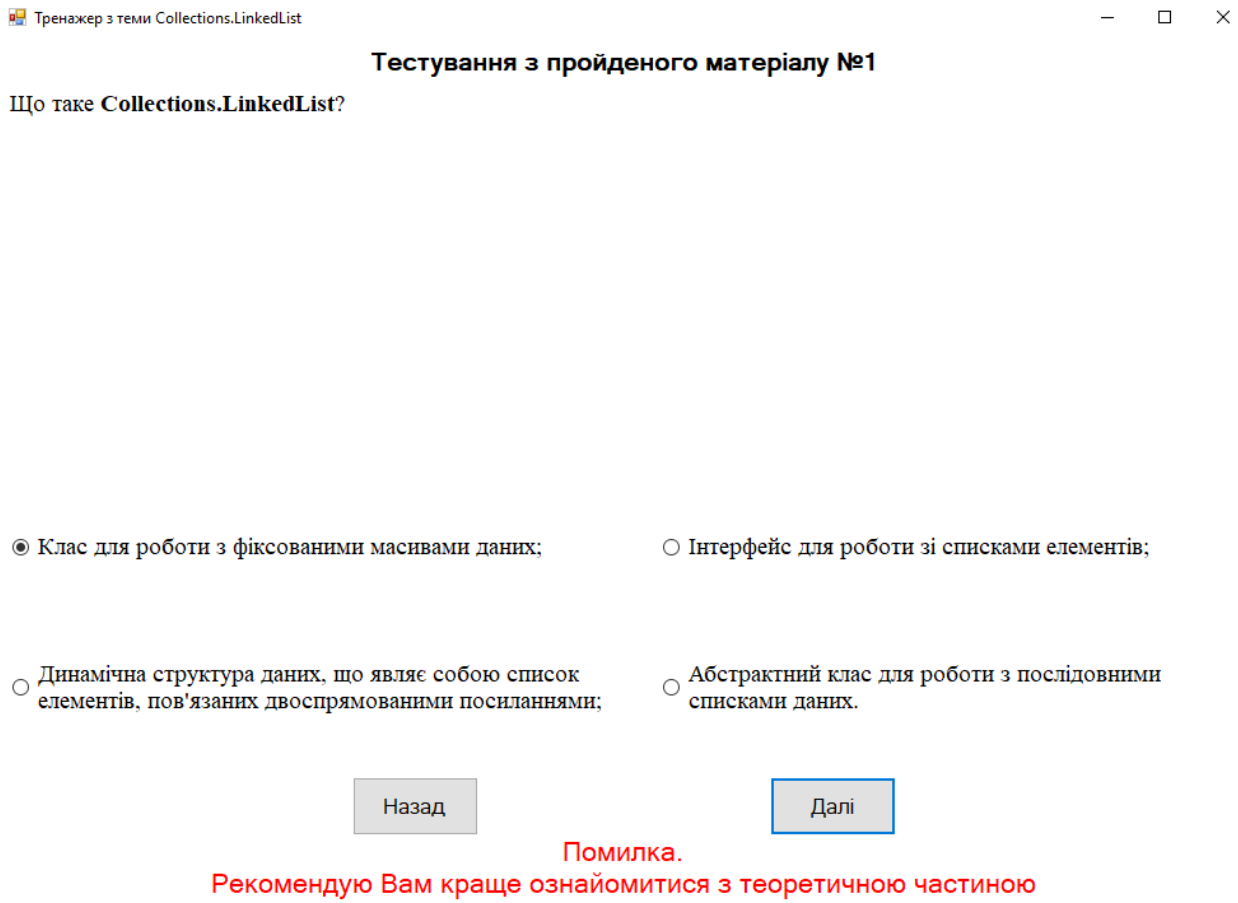

Рисунок 4.6 – Хибна відповідь викликає повідомлення №3 у вікні тренажера

Після повного опрацювання всіх завдань та теоретичних відомостей користувач бачить “Прощальне” вікно програми (Рисунок 4.7) котре містить текст з вітальним повідомленням та переліком здобутих користувачем за час користуванням тренажером знань та вмінь. Також є кнопка “Кінець”, котра при натисканні повертає користувача до “Стартового” вікна програми (Рисунок 4.1).

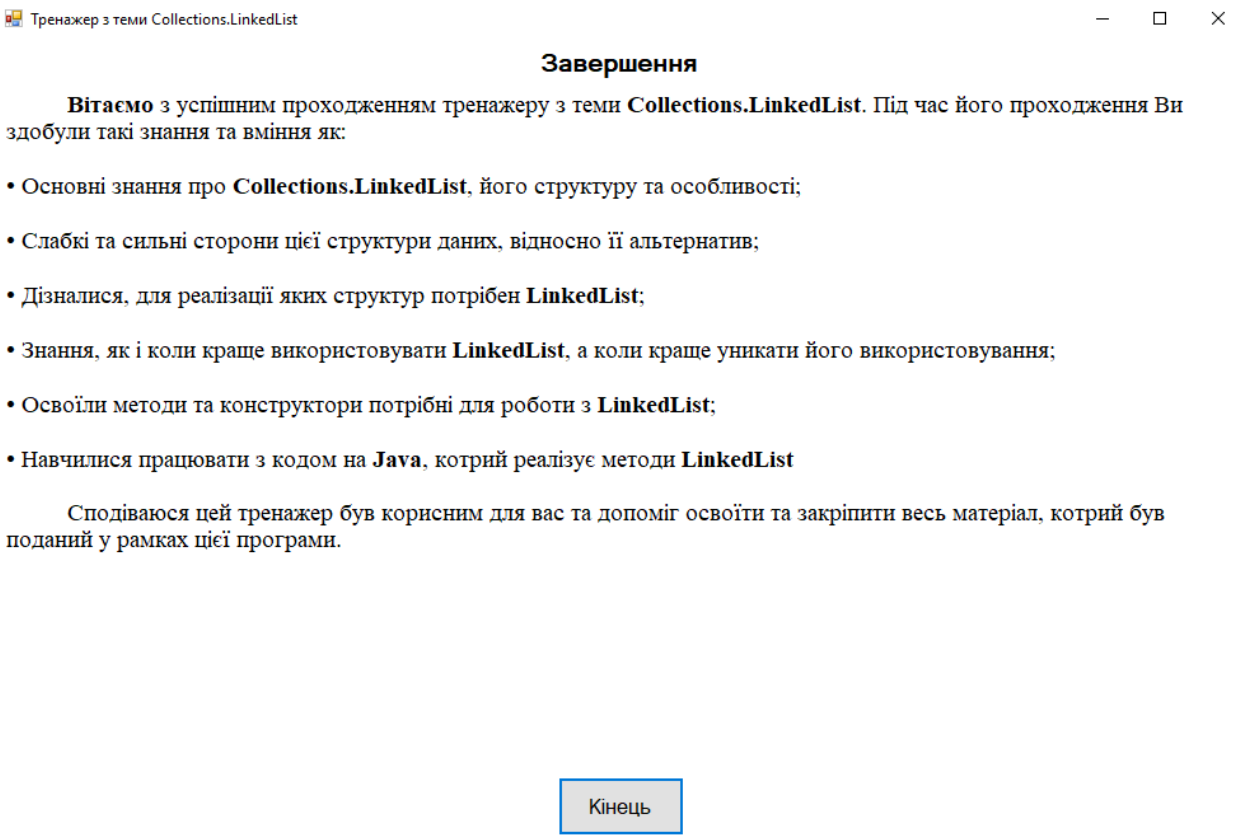

Рисунок 4.7 – “Прощальне” вікно програми

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи, було розроблено навчальне програмне забезпечення з теми «Collections.LinkedList» для мови програмування Java. Розробка даного навчального програмного забезпечення була важливим та актуальним завданням у сучасному інформаційному суспільстві, оскільки навчальне забезпечення такого типу є дуже потрібними в час стрімкого розвитку дистанційної освіти, оскільки надають студентам можливість отримувати нові та закріплювати вже існуючи навички та вміння.

Під час розробки навчального програмного забезпечення з теми «Collections. LinkedList» дисципліни «Програмування» було виконано ряд завдань та досягнуто основних цілей проекту:

1. Розроблено алгоритм навчального програмного забезпечення з теми «Collections. LinkedList»;.
2. Складено блок-схему алгоритму роботи навчального програмного забезпечення;
3. Розроблено навчальне програмне забезпечення з даної теми.

В алгоритмі описується процес ознайомлення з такими темами:

1. Що таке Collections. LinkedList?;
2. Переваги та недоліки LinkedList;
3. Конструктори, методи та робота з ними;
4. Додатково про методи класу LinkedList;
5. Застосування Collections. LinkedList.

В тестах котрі слідують після теоретичної частини перевіряється засвоєння матеріалу

Розроблене програмне навчальне забезпечення має таку структуру:

- Стартова сторінка:
  - Тема тренажера;
  - Вітальне повідомлення.
- Теоретична частина:
  - Назва теми;
  - Теоретичні відомості по ній.
- Частина теоретичних завдань:
  - Завдання;
  - Варіанти відповідей.
- Кінцева сторінка:
  - Прощальне повідомлення.

Для роботи всіх кнопок було зроблено функції до яких вони були закріплені. Було створено функції для виведення вмісту панелі кожної теми та кожного завдання.

Всі вимоги, описані в постановці задачі, були виконані.

СПИСОК ЛІТЕРАТУРИ

1. Е. Шилдт "Java. Бібліотека класів". -1996 – 1176 с.
2. Брюс Эккель "Java. Ефективне програмування" – 2017 – 412 с.
3. В.І. Соловйов "Дистанційне навчання: теорія, технології, практика" – 2006 – 448 с.
4. Сергій Омельчук “Дистанційне навчання студентів: теорія і практика” – 2020 – 78с.
5. Герберт Шлідт "Java: The Complete Reference" – 2005 – 1216 с.
6. Вільям Андерсон та Майкл Грехам Мур "The Handbook of Distance Education". – 2003 – 898с.
7. Джон Скіт. " C# 11 and .NET 6: A Beginner's Guide ". – 2021 – 528 с.
8. Мазур М.П. Розвиток дистанційного навчання в Україні як складової інформатизації сучасного суспільства / М.П.Мазур // Інформатика та інформаційні технології в навчальних закладах. – 2007.
9. О. В. Ольховська, О. О. Черненко. Методичні рекомендації до виконання кваліфікаційної роботи для студентів спеціальності 122 Комп’ютерні науки освітня програма «Комп’ютерні науки» ступеня бакалавра / О. В. Ольховська, О. О. Черненко. – Полтава : ПУЕТ, 2022. – 67 с.
10. Мельнийький Ян Віталійович – ПОЯСНЮВАЛЬНА ЗАПИСКА ДО БАКАЛАВРСЬКОЇ РОБОТИ на тему РОЗРОБКА ТРЕНАЖЕРУ ДЛЯ ВИВЧЕННЯ ОСНОВ МОВИ ПРОГРАМУВАННЯ PYTHON – 2020. - (http://dspace.puet.edu.ua/handle/123456789/9015).
11. Лєбєдєва М. О. – ПОЯСНЮВАЛЬНА ЗАПИСКА ДО ДИПЛОМНОЇ РОБОТИ на тему ТРЕНАЖЕР З ТЕМИ «КОНТЕКСТОВІЛЬНІ МОВИ» ДИСТАНЦІЙНОГО НАВЧАЛЬНОГО КУРСУ «ТЕОРІЯ ПРОГРАМУВАННЯ» - 2021. - (http://dspace.puet.edu.ua/handle/123456789/11339)
12. Соболєв Денис Андрійович – ПОЯСНЮВАЛЬНА ЗАПИСКА ДО БАКАЛАВРСЬКОЇ РОБОТИ на тему ТРЕНАЖЕР З ТЕМИ «АЛГОРИТМИ СИНТАКСИЧНОГО АНАЛІЗУ» ДИСТАНЦІЙНОГО НАВЧАЛЬНОГО КУРСУ «ТЕОРІЯ ПРОГРАМУВАННЯ» - 2021. - (http://dspace.puet.edu.ua/handle/123456789/10354)
13. Олексійчук Ю. Ф. – Розробка та впровадження дистанційного курсу з дисципліни «Програмування» / Ю. Ф. Олексійчук // Дистанційна освіта: забезпечення доступності та неперервної освіти впродовж життя

(e-learning and university education-2017): матеріали XLII Міжнародної науково-методичної конференції (м. Полтава, 9–10 лютого 2017 року) – Полтава: ПУЕТ, 2017. – С. 167-169.
14. Мандрика В. М. – Тренажер з теми «1-R алгоритм» дисципліни «Комп'ютерний аналіз статистичних даних» / В. М. Мандрика, Ю. Ф. Олексійчук // Комп'ютерні науки і прикладна математика (КНіПМ-2018): матеріали науково-практичного семінару. Випуск 1 – Полтава: Кафедра ММСІ ПУЕТ, 2018. – С. 27-30.
15. Ярмоленко А. В. – Алгоритм роботи тренажеру з теми «Асимптотичні оцінки функцій» дисципліни «Аналіз алгоритмів» / А. В. Ярмоленко, Ю. Ф. Олексійчук // Комп'ютерні науки і прикладна математика (КНіПМ-2018): матеріали науково-практичного семінару. Випуск 2 – Полтава: Кафедра ММСІ ПУЕТ, 2018. – С.14-16.
16. Кривошей О. С. – Програмна реалізація елементів тренажера з теми «Виявлення аномальних спостережень за допомогою критерію Томпсона» дисципліни «Аналіз даних і прикладні пакети статистичної обробки». / О. С. Кривошей , Ю. Ф. Олексійчук // Актуальні питання розвитку науки та забезпечення якості освіти у XXI столітті: тези доповідей XLIII Міжнародної наукової студентської конференції за підсумками науково-дослідних робіт студентів за 2019 рік (м. Полтава, 07–08 квітня 2020 року). Частина 2 – Полтава: ПУЕТ, 2020 – С.123-125.
17. Фролов Д. О. – Розробка тренажеру з теми «Формальні граматики» дистанційного навчального курсу «Теорія програмування» – 2023.

ДОДАТОК А.

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace WindowsFormsApp1
{
    public partial class Form1 : Form
    {
        byte step;
        byte maxStep = 22;
        byte correctAnswer;
        byte RID;
        byte x = 1;
        byte x1;
        byte num;
        public Form1()
        {
            InitializeComponent();
            Start_window();
        }

        private void CalculateStart()
        {
            button2.Size = new Size(button1.Size.Width, button1.Size.Height);
            button3.Size = new Size(button1.Size.Width, button1.Size.Height);
```

```
        int a = (button1.Size.Width / 2);

        int b = (ClientSize.Width / 3);

        int c = ((ClientSize.Width / 2) - a);


        int d = button1.Size.Height + 55;

        int f = radioButton1.Size.Height + d + 30;


        head2.Location = new Point((ClientSize.Width - head2.Size.Width) / 2, -2);


        radioButton1.Size = new Size(ClientSize.Width / 2, 80);

        radioButton2.Size = radioButton1.Size;

        radioButton3.Size = new Size(ClientSize.Width / 2 - 42, 80);

        radioButton4.Size = radioButton3.Size;

        radioButton1.Location = new Point(12, ClientSize.Height - (f + 110));

        radioButton2.Location = new Point(12, ClientSize.Height - f);

        radioButton3.Location = new Point(radioButton1.Size.Width + 30, ClientSize.Height - (f + 110));

        radioButton4.Location = new Point(radioButton1.Size.Width + 30, ClientSize.Height - f);


        button1.Location = new Point(b - a, ClientSize.Height - d);

        button2.Location = new Point(c, ClientSize.Height - d);

        button3.Location = new Point(b * 2 - a, ClientSize.Height - d);

        FALSE.Location = new Point((ClientSize.Width - FALSE.Size.Width) / 2, ClientSize.Height -
(FALSE.Size.Height + 5));


        pictureBox1.Image = Properties.Resources.image1;

        pictureBox1.Location = new Point((ClientSize.Width / 2) - (pictureBox1.Width / 2),
ClientSize.Height - d - pictureBox1.Height - 30);
```

```
        if (x == 1)
        {
            RichTextBox1.Size = new Size(ClientSize.Width - 15, ClientSize.Height - (d +
button1.Size.Height + 15));
        }
        else
        {
            RichTextBox1.Size = new Size(ClientSize.Width - 15, ClientSize.Height - (f + 170));
        }
    }

    public void Next_button(object sender, EventArgs e)
    {
        x1 = x;
        RichTextBox1.Clear();
        CalculateStart();
        step++;
        Step(step);
        if (x1 == x && x1 == 0)
        {
            num++;
            head2.Text = "Тестування з пройденого матеріалу №" + num;
        }
        if (x1 != x && x1 == 1)
        {
            head2.Text = "Тестування з пройденого матеріалу №" + num;
```

```
    }
    if (x1 != x && x1 == 0)
    {
        num++;

    }
}

public void Back_button(object sender, EventArgs e)
{
    x1 = x;
    RichTextBox1.Clear();
    CalculateStart();
    step--;
    Step(step);

    if (x1 == x && x1 == 0)
    {
        num--;
        head2.Text = "Тестування з пройденого матеріалу №" + num;
    }
    if (x1 != x && x1 == 1)
    {
        num--;
        head2.Text = "Тестування з пройденого матеріалу №" + num;
    }
}
```

```
private void Start_window()

{

    CalculateStart();

    button2.Text = "Розпочати";

    head2.Visible = true;

    head2.Text = "Тренажер з теми Collections.LinkedList";

    button1.Visible = false;

    button2.Visible = true;

    button3.Visible = false;

    FALSE.Visible = false;

    pictureBox1.Visible = false;

    RichTextBox1.Text = "\tВас вітає навчальна програма-тренажер з теми
Collections.LinkedList. Цей навчальний ресурс покликаний перевірити наявні та виробити нові
знання та навички роботи з цією динамічною структурою даних. \n\n Під час ознайомлення з
програмою вам буде запропоновано: \n\n• Теоретичні та практичні відомості - теоретична частина
з розділами, що описують LinkedList, його операції, застосування та приклади коду;\n• Тестові
питання - оберіть правильні відповіді з декількох варіантів на питання, котрі стосуються основних
концепцій та операцій LinkedList.\n\n Як працювати з програмою:\n\n• Прочитайте теоретичну
частину, яка містить пояснення та приклади коду;\n• Дайте відповідь на тестові запитання, щоб
закріпити знання.\n\n Важливо:\n\n• Для кращого розуміння теми рекомендується мати базові
знання з програмування на Java;\n• Ви можете використовувати онлайн-ресурси та документацію
для більш глибокого вивчення LinkedList.\n\nПам'ятайте:\n\n• Ця програма є лише інструментом
для навчання. Для більш глибокого розуміння теми рекомендується використовувати інші ресурси
та практикуватися самостійно.";



    BoldText(RichTextBox1, "Вас вітає");

    BoldText(RichTextBox1, "Під час ознайомлення з програмою вам буде запропоновано:");

    BoldText(RichTextBox1, "Як працювати з програмою:");

    BoldText(RichTextBox1, "Важливо:");

    BoldText(RichTextBox1, "Пам'ятайте:");

    UnderlineText(RichTextBox1, "програма є лише інструментом для навчання");



    Switch(0);
```

```
}

private void Final_window()
{
    CalculateStart();
    button2.Text = "Кінець";
    head2.Visible = true;
    button1.Visible = false;
    button2.Visible = true;
    button3.Visible = false;
    FALSE.Visible = false;

    Switch(0);
}

private void BoldText(RichTextBox box, string text)
{
    int i = box.Text.IndexOf(text);
    while (i >= 0)
    {
        box.Select(i, text.Length);
        box.SelectionFont = new Font(box.Font, FontStyle.Bold);
        i = box.Text.IndexOf(text, i + text.Length);
    }
}

private void ColorText(RichTextBox box, string text, Color color)
```

```
{
    int i = box.Text.IndexOf(text);
    while (i >= 0)
    {
        box.Select(i, text.Length);
        box.SelectionColor = color;
        i = box.Text.IndexOf(text, i + text.Length);
    }
}

private void UnderlineText(RichTextBox box, string text)
{
    int i = box.Text.IndexOf(text);
    while (i >= 0)
    {
        box.Select(i, text.Length);
        box.SelectionFont = new Font(box.Font, FontStyle.Underline);
        i = box.Text.IndexOf(text, i + text.Length);
    }
}

private void ItalicText(RichTextBox box, string text)
{
    int i = box.Text.IndexOf(text);
    while (i >= 0)
    {
        box.Select(i, text.Length);
```

```
        box.SelectionFont = new Font(box.Font, FontStyle.Italic);
        i = box.Text.IndexOf(text, i + text.Length);
    }
}

private void RegularText(RichTextBox box, string text)
{
    int i = box.Text.IndexOf(text);
    while (i >= 0)
    {
        box.Select(i, text.Length);
        box.SelectionFont = new Font(box.Font, FontStyle.Regular);
        i = box.Text.IndexOf(text, i + text.Length);
    }
}

private void TForm()
{
    x = 1;

    Switch(0);
    button2.Text = "Далі";
    head2.Visible = true;
    button1.Visible = true;
    button2.Visible = true;
    button2.Location = new Point(button3.Location.X, button3.Location.Y);
    button3.Visible = false;
```

```
    FALSE.Visible = false;

}

private void PForm()
{
    x = 0;

    Switch(1);
    button2.Text = "Далі";
    head2.Visible = true;
    button1.Visible = true;
    button2.Visible = false;
    button3.Visible = true;
    FALSE.Visible = false;
}

private void Switch(double n)
{
    BoldText(RichTextBox1, "Collections.LinkedList");
    BoldText(RichTextBox1, "LinkedList");
    BoldText(RichTextBox1, "ArrayList");
    BoldText(RichTextBox1, "•");
    BoldText(RichTextBox1, "Java");
    BoldText(RichTextBox1, "AbstractSequentalList");
    BoldText(RichTextBox1, "List");
    BoldText(RichTextBox1, "Deque");
```

```
BoldText(RichTextBox1, "Queue");

BoldText(RichTextBox1, "while");

BoldText(RichTextBox1, "if");

BoldText(RichTextBox1, "int");

BoldText(RichTextBox1, "else");

BoldText(RichTextBox1, "new");

BoldText(RichTextBox1, "for");

BoldText(RichTextBox1, "double");

BoldText(RichTextBox1, "Результат:");


ColorText(RichTextBox1, "while", Color.DarkBlue);

ColorText(RichTextBox1, "if", Color.DarkBlue);

ColorText(RichTextBox1, "int", Color.DarkBlue);

ColorText(RichTextBox1, "out", Color.MediumPurple);

ColorText(RichTextBox1, "else", Color.DarkBlue);

ColorText(RichTextBox1, "for", Color.DarkBlue);

ColorText(RichTextBox1, "new", Color.DarkBlue);

ColorText(RichTextBox1, "double", Color.DarkBlue);

ColorText(RichTextBox1, "println", Color.Black);

ColorText(RichTextBox1, "print", Color.Black);

ColorText(RichTextBox1, "\"яблуко\"", Color.Green);

ColorText(RichTextBox1, "\"банан\"", Color.Green);

ColorText(RichTextBox1, "\"апельсин\"", Color.Green);

ColorText(RichTextBox1, "\"манго\"", Color.Green);


ItalicText(RichTextBox1, "out");
```

```
    RegularText(RichTextBox1, "println");
    RegularText(RichTextBox1, "print");
    RegularText(RichTextBox1, "LinkedList<String> ");
    RegularText(RichTextBox1, "LinkedList<>()");

    CalculateStart();

    if (n == 0)
    {
        RichTextBox1.Visible = true;
        radioButton1.Visible = false;
        radioButton2.Visible = false;
        radioButton3.Visible = false;
        radioButton4.Visible = false;
    }
    if (n == 1)
    {
        RichTextBox1.Visible = true;
        radioButton1.Visible = true;
        radioButton2.Visible = true;
        radioButton3.Visible = true;
        radioButton4.Visible = true;
    }
}

private void CheckAnswer(object sender, EventArgs e)
{
```

```
if (radioButton1.Checked)
{
    RID = 1;
}

if (radioButton2.Checked)
{
    RID = 2;
}

if (radioButton3.Checked)
{
    RID = 3;
}

if (radioButton4.Checked)
{
    RID = 4;
}




if (RID == correctAnswer)
{
    x1 = x;
    step++;
```

```
            Step(step);
            if (x1 == x && x1 == 0)
            {
               num++;
               head2.Text = "Тестування з пройденого матеріалу №" + num;
            }
            if (x1 != x && x1 == 1)
            {
               head2.Text = "Тестування з пройденого матеріалу №" + num;
            }
            if (x1 != x && x1 == 0)
            {
               num++;


            }
         }
         else
         {
            Random();
            FALSE.Location = new Point((ClientSize.Width - FALSE.Size.Width) / 2, ClientSize.Height -
(FALSE.Size.Height + 5));
         }
      }


      private void Random()
      {
         FALSE.Visible = true;
         Random random = new Random();
```

```
            int randomNumber = random.Next(1, 4);



            if (randomNumber == 1)

            {

                FALSE.Text = "Неправильна відповідь.\nБудь ласка, поверніться до теоретичної частини";

            }

            if (randomNumber == 2)

            {

                FALSE.Text = "Помилка.\nРекомендую Вам краще ознайомитися з теоретичною частиною";

            }

            if (randomNumber == 3)

            {

                FALSE.Text = "Ви допустилися помилки.\nРекомендую повернутися до теоретичного матеріалу";

            }

        }



        private void Step(double Step)

        {

            RichTextBox1.Location = new Point(9, 41);



            if (Step == maxStep)

            {

                Start_window();

                step = 0;

            }

            if (Step == 0)
```

```
{
    Start_window();
}



switch (Step)
{


    case 1:
        num = 1;
        pictureBox1.Visible = true;
        pictureBox1.Image = Properties.Resources.image1;
        head2.Text = "Що таке Collections.LinkedList?";
        RichTextBox1.Text = "\tCollections.LinkedList - це динамічна структура даних, що являє собою список елементів, пов'язаних один з одним за допомогою двоспрямованих посилань. LinkedList з'явився в Java 1.2 (1998) як альтернатива ArrayList. На відміну від ArrayList, який використовує фіксований масив для зберігання елементів, LinkedList не має фіксованого розміру та може динамічно розширюватися або скорочуватися під час додавання або видалення елементів. Він успадковує AbstractSequentalList і реалізує інтерфейси List та Deque(а через останній і Queue).\n\nСтруктура:\n\n\t• Елементи - кожен елемент містить значення та два посилання: next, яке вказує на наступний елемент в списку, і \n\t\t\t  previous, яке вказує на попередній елемент;\n\t• Голова - це посилання на перший елемент в списку;\n\t• Хвіст - це посилання на останній елемент в списку.";
        TForm();

        BoldText(RichTextBox1, "Елементи -");
        BoldText(RichTextBox1, "Структура:");
        BoldText(RichTextBox1, "Голова -");
        BoldText(RichTextBox1, "Хвіст -");
        BoldText(RichTextBox1, "Java 1.2 (1998)");
        break;
```

```
            case 2:
                pictureBox1.Visible = false;
                RichTextBox1.Text = "Що таке Collections.LinkedList?";

                radioButton1.Text = "Клас для роботи з фіксованими масивами даних;";
                radioButton2.Text = "Динамічна структура даних, що являє собою список елементів, пов'язаних двоспрямованими посиланнями;";
                radioButton3.Text = "Інтерфейс для роботи зі списками елементів;";
                radioButton4.Text = "Абстрактний клас для роботи з послідовними списками даних.";

                PForm();

                correctAnswer = 2;
                break;

            case 3:
                RichTextBox1.Text = "Яка з наступних структур даних не має фіксованого розміру?";

                radioButton1.Text = "ArrayList;";
                radioButton2.Text = "HashMap;";
                radioButton3.Text = "LinkedList;";
                radioButton4.Text = "Vector.";

                PForm();

                correctAnswer = 3;
                break;
```

case 4:

head2.Text = "Що таке Collections.LinkedList?";

RichTextBox1.Text = "\tЧерез динамічність та простоту вставок і видалень він краще масивів. Також він має кілька недоліків, такі як неможливість прямого доступу до елементів, натомість нам потрібно починати з початку та переходити за посиланням, щоб дістатися до елемента, який ми шукаємо, що значно збільшує час роботи операцій отримання елементів.\n\n\nПорівняння часу роботи операції отримання елементу з індексом 8000000 для LinkedList та ArrayList:\n\nint result;\nstartTime = System.nanoTime();\nresult = list1.get(8000000);\nfinishTime=System.nanoTime();\nSystem.out.println(result);\nSystem.out.println(\"ArrayList get: \"+(double)(finishTime-startTime)/1000000 +\"ms\");\n\nstartTime = System.nanoTime();\nresult = list2.get(8000000);\nfinishTime=System.nanoTime();\nSystem.out.println(result);\nSystem.out.println(\"LinkedList get: \"+(double)(finishTime-startTime)/1000000 +\"ms\");\n\n\tРезультат:\nArrayList get:0.00591ms\nLinkedList get:11.678236ms\n\n\n\tОскільки LinkedList діє як динамічний масив, і нам не потрібно вказувати розмір під час його створення, розмір списку автоматично збільшується, коли ми додаємо та видаляємо елементи. LinkedList реалізований з використанням структури даних двозв'язкового списку\n\tОсновна відмінність між звичайним зв'язаним списком та подвійним зв'язаним списком полягає в тому, що двозв'язаковий список містить додатковий покажчик, який зазвичай називають попереднім покажчиком. Ці додаткові посилання збільшують обсяг пам'яті, необхідний для зберігання LinkedList.";

TForm();

BoldText(RichTextBox1, "Порівняння часу роботи операції отримання елементу з індексом 8000000 для LinkedList та ArrayList:");

ItalicText(RichTextBox1, "nanoTime");

ColorText(RichTextBox1, "0", Color.Blue);

ColorText(RichTextBox1, "1", Color.Blue);

ColorText(RichTextBox1, "8000000", Color.Blue);

ColorText(RichTextBox1, "println", Color.Black);

ColorText(RichTextBox1, "list1", Color.Black);

ColorText(RichTextBox1, "\"ArrayList get: \"", Color.Green);

ColorText(RichTextBox1, "\"ms\"", Color.Green);

ColorText(RichTextBox1, "\"LinkedList get: \"", Color.Green);

ColorText(RichTextBox1, "ArrayList get:0.00591ms", Color.Black);

```
            ColorText(RichTextBox1, "LinkedList get:11.678236ms", Color.Black);


            RegularText(RichTextBox1, "List<Integer>");

            RegularText(RichTextBox1, "ArrayList<>();");

            RegularText(RichTextBox1, "LinkedList<>();");

            RegularText(RichTextBox1, "ArrayList get:");

            RegularText(RichTextBox1, "LinkedList get:");

            RegularText(RichTextBox1, "ArrayList get:0.00591ms");

            RegularText(RichTextBox1, "LinkedList get:11.678236ms");


            break;


        case 5:

            head2.Text = "Переваги та недоліки";

            RichTextBox1.Text = "\tНадалі, порівняння LinkedList буде переважно проводитися саме з ArrayList, адже він створювався саме як альтернатива йому.\n\n\tПереваги Collections.LinkedList:\n\n• Ефективне додавання та видалення елементів в середині списку - є основною перевагою, завдяки двоспрямованим посиланням, додавання та видалення елементів в середині списку LinkedList не потребує переміщення інших елементів, як це відбувається в ArrayList. Це робить LinkedList більш ефективним для цих операцій;\n\n• Не потрібне попереднє виділення пам'яті - LinkedList не потребує попереднього виділення пам'яті для зберігання елементів, оскільки розмір списку може динамічно змінюватися. Це може бути корисно для програм, де розмір списку невідомий заздалегідь;\n\n• Можливість реалізації черги (FIFO) або стека (LIFO) - LinkedList може використовуватися для реалізації черги (FIFO), де елементи видаляються в тому ж порядку, в якому вони були додані, або стека (LIFO), де елементи видаляються в зворотному порядку до їх додавання.\n\n\tНедоліки Collections.LinkedList:\n\n• Менш ефективний доступ до елементів за індексом - доступ до елементів LinkedList за індексом може бути менш ефективним, ніж в ArrayList, оскільки LinkedList не використовує фіксований масив. Це пов'язано з тим, що для доступу до елемента за індексом потрібно пройти через список, починаючи з першого елемента, доки не буде знайдений елемент з потрібним індексом;\n\n• Більше використання пам'яті - LinkedList використовує більше пам'яті, ніж ArrayList, через додаткові посилання, необхідні для зв'язку елементів.";


            BoldText(RichTextBox1, "Переваги Collections.LinkedList:");

            BoldText(RichTextBox1, "Ефективне додавання та видалення елементів в середині списку -");

            BoldText(RichTextBox1, "Не потрібне попереднє виділення пам'яті -");
```

```
            BoldText(RichTextBox1, "Можливість реалізації черги (FIFO) або стека (LIFO) -");
            BoldText(RichTextBox1, "Недоліки Collections.LinkedList:");
            BoldText(RichTextBox1, "Менш ефективний доступ до елементів за індексом -");
            BoldText(RichTextBox1, "Більше використання пам'яті -");

            TForm();
            break;

        case 6:
            RichTextBox1.Text = "Яку з наступних структур даних можна реалізувати за допомогою LinkedList?";

            radioButton1.Text = "Черга (FIFO);";
            radioButton2.Text = "Стек (LIFO);";
            radioButton3.Text = "Двоспрямований список;";
            radioButton4.Text = "Все вищезазначене.";

            PForm();

            correctAnswer = 4;

            break;

        case 7:
            RichTextBox1.Text = "Яка з наступних характеристик є недоліком LinkedList?";

            radioButton1.Text = "Ефективне додавання та видалення елементів в середині списку;";
```

```
            radioButton2.Text = "Не потребує попереднього виділення пам'яті;";
            radioButton3.Text = "Менш ефективний доступ до елементів за індексом;";
            radioButton4.Text = "Можливість реалізації черги (FIFO) або стека (LIFO).";

            PForm();

            correctAnswer = 3;

            break;

        case 8:
            RichTextBox1.Text = "Яка основна перевага LinkedList над ArrayList?";

            radioButton1.Text = "Швидший доступ до елементів за індексом;";
            radioButton2.Text = "Більш ефективне додавання та видалення елементів в середині списку;";
            radioButton3.Text = "Менше використання пам'яті;";
            radioButton4.Text = "Фіксований розмір списку.";

            PForm();

            correctAnswer = 2;

            break;

        case 9:
            head2.Text = "Конструктори, методи та робота з ними";
```

RichTextBox1.Text = "\tКлас LinkedList має наступні конструктори:\n\n• LinkedList() - котрий створює порожній список;\n• LinkedList(Collection<? extends E> col) - створює список, в який додає всі елементи колекції col.\n\n\tLinkedList містить всі методи, котрі містять в собі інтерфейси List, Queue, Deque, основні з них:\n\n• addFirst() / offerFirst() - додає елемент в початок списку;\n• addLast() / offerLast() - додає елемент в кінець списку;\n• removeFirst() / pollFirst() - видаляє перший елемент з початку списку;\n• removeLast() / pollLast() - видаляє останній елемент з кінця списку;\n• getFirst() / peekFirst() - отримує перший елемент;\n• getLast() / peekLast() - отримує останній елемент.\n\n• add(n, E) - додає елемент E за індексом n;\n• remove(n) - видаляє елемент за індексом n;\n• contains(E) - перевірка на наявність елементу E;\n• size() - повертає розмір списку;\n• isEmpty() - Перевіряє, чи пустий список;\n• iterator() - повертає ітератор для обходу елементів списку;\n• celar() - очищує список.\n\n\tПриклад:\n\n\nLinkedList <String> list = new LinkedList<>();\n\nlist.addFirst(\"A\"); // Додавання елемента \"A\" в початок списку\nlist.addFirst(\"B\"); // Додавання елемента \"B\" в початок списку\nlist.addLast(\"C\"); // Додавання елемента \"C\" в кінець списку\nlist.add(\"D\"); // Додавання елемента \"D\" в кінець списку\nlist.add(1,\"E\"); // Додавання елемента \"E\" в список за індексом 1\n\nfor(String str:list){\n\tSystem.out.println(str);\n}\n\n\tРезультат:\n\nB\nE\nA\nC\nD";

TForm();

BoldText(RichTextBox1, "Клас LinkedList має наступні конструктори:");

BoldText(RichTextBox1, "LinkedList()");

BoldText(RichTextBox1, "LinkedList(Collection<? extends E> col)");

BoldText(RichTextBox1, "LinkedList містить всі методи, котрі містять в собі інтерфейси List, Queue, Deque, основні з них:");

BoldText(RichTextBox1, "addFirst() / offerFirst()");

BoldText(RichTextBox1, "addLast() / offerLast()");

BoldText(RichTextBox1, "removeFirst() / pollFirst()");

BoldText(RichTextBox1, "removeLast() / pollLast()");

BoldText(RichTextBox1, "getFirst() / peekFirst()");

BoldText(RichTextBox1, "getLast() / peekLast()");

BoldText(RichTextBox1, "add(n, E)");

BoldText(RichTextBox1, "contains(E)");

BoldText(RichTextBox1, "remove(n)");

BoldText(RichTextBox1, "size()");

BoldText(RichTextBox1, "isEmpty()");

BoldText(RichTextBox1, "iterator()");

BoldText(RichTextBox1, "celar()");

```
            BoldText(RichTextBox1, "Приклад:");


            RegularText(RichTextBox1, ".isEmpty()");

            RegularText(RichTextBox1, ".iterator()");

            RegularText(RichTextBox1, ".size();");

            RegularText(RichTextBox1, "LinkedList <String> ");

            RegularText(RichTextBox1, "LinkedList<>()");


            ColorText(RichTextBox1, "1", Color.Blue);

            ColorText(RichTextBox1,"// Додавання елемента \"A\" в початок
списку",Color.DarkGreen);

            ColorText(RichTextBox1, "// Додавання елемента \"B\" в початок списку",
Color.DarkGreen);

            ColorText(RichTextBox1, "// Додавання елемента \"C\" в кінець списку",
Color.DarkGreen);

            ColorText(RichTextBox1, "// Додавання елемента \"D\" в кінець списку",
Color.DarkGreen);

            ColorText(RichTextBox1, "// Додавання елемента \"E\" в список за індексом 1",
Color.DarkGreen);

            ColorText(RichTextBox1, "\"A\"", Color.Green);

            ColorText(RichTextBox1, "\"B\"", Color.Green);

            ColorText(RichTextBox1, "\"C\"", Color.Green);

            ColorText(RichTextBox1, "\"D\"", Color.Green);

            ColorText(RichTextBox1, "\"E\"", Color.Green);

            break;


        case 10:

            RichTextBox1.Text = "Який конструктор LinkedList створює порожній список?";


            radioButton1.Text = "LinkedList();";
```

```
            radioButton2.Text = "LinkedList<String>();";

            radioButton3.Text = "LinkedList(Collection<? extends E> col);";

            radioButton4.Text = "LinkedList.create().";


            PForm();


            correctAnswer = 1;


            break;


        case 11:

            RichTextBox1.Text = "Який метод LinkedList використовується для додавання елемента в початок списку?";


            radioButton1.Text = "addFirst();";

            radioButton2.Text = "addLast();";

            radioButton3.Text = "add();";

            radioButton4.Text = "insert.";


            PForm();


            correctAnswer = 1;


            break;


        case 12:

            RichTextBox1.Text = "Який метод LinkedList використовується для отримання першого елемента списку?";
```

```
            radioButton1.Text = "getLast();";

            radioButton2.Text = "get();";

            radioButton3.Text = "getFirst();";

            radioButton4.Text = "peek().";


            PForm();


            correctAnswer = 3;


            break;


        case 13:

            RichTextBox1.Text = "Який результат виведе наступний код?\n\nLinkedList<Integer> list
= new LinkedList<>();\nlist.add(5);\nlist.add(3);\nlist.add(0,
1);\nlist.addFirst(2);\nlist.addLast(4);\n\nSystem.out.println(list.getFirst());\nSystem.out.println(list.getLa
st());\nSystem.out.println(list.get(2));";


            radioButton1.Text = "5 4 3;";

            radioButton2.Text = "2 4 1;";

            radioButton3.Text = "1 5 3;";

            radioButton4.Text = "2 5 1.";


            ColorText(RichTextBox1, "0", Color.Blue);

            ColorText(RichTextBox1, "1", Color.Blue);

            ColorText(RichTextBox1, "2", Color.Blue);

            ColorText(RichTextBox1, "3", Color.Blue);

            ColorText(RichTextBox1, "4", Color.Blue);

            ColorText(RichTextBox1, "5", Color.Blue);
```

```
            PForm();

            RegularText(RichTextBox1, "LinkedList");

            correctAnswer = 2;



            break;



        case 14:

            head2.Text = "Додатково про методи класу LinkedList";

            RichTextBox1.Text = "\tcontains(E) - перевірка на наявність елементу E\nЦей метод використовується для перевірки, чи містить список LinkedList заданий елемент. Він повертає true, якщо елемент знайдено, і false, якщо ні.\n\n\nLinkedList<String> colors = new LinkedList<>();\ncolors.add(\"червоний\");\ncolors.add(\"зелений\");\ncolors.add(\"синій\");\n\nif (colors.contains(\"червоний\")) {\n\tSystem.out.println(\"Список містить червоний колір\");\n} else {\n\tSystem.out.println(\"Список не містить червоний колір\");\n}\n\n\tРезультат:\nСписок містить червоний колір\n\n----------\n\n\tsize() - повертає розмір списку\nЦей метод повертає кількість елементів у списку LinkedList.\n\n\nLinkedList<Integer> numbers = new LinkedList<>();\nnumbers.add(10);\nnumbers.add(20);\nnumbers.add(30);\n\nint size = numbers.size();\nSystem.out.println(\"Розмір списку: \" + size);\n\n\tРезультат:\nРозмір списку: 3\n\n----------\n\n\tisEmpty() - Перевіряє, чи пустий список\nЦей метод повертає true, якщо список LinkedList порожній, і false, якщо він містить хоча б один елемент.\n\n\nLinkedList<String> names = new LinkedList<>();\n\nif (names.isEmpty()) {\n\tSystem.out.println(\"Список порожній\");\n} else {\n\tSystem.out.println(\"Список не порожній\");\n}\n\n\tРезультат:\nСписок порожній\n\n----------\n\n\titerator() - повертає ітератор для обходу елементів списку\nЦей метод повертає ітератор, який можна використовувати для обходу елементів списку LinkedList.\n\n\nLinkedList<String> fruits = new LinkedList<>();\nfruits.add(\"яблуко\");\nfruits.add(\"банан\");\nfruits.add(\"апельсин\");\n\nIterator<String> iterator = fruits.iterator();\nwhile (iterator.hasNext()) {\n\tString fruit = iterator.next();\n\tSystem.out.println(fruit);\n}\n\n\tРезультат:\nяблуко\nбанан\nапельсин";



            TForm();

            BoldText(RichTextBox1, "contains(E)");

            BoldText(RichTextBox1, "size()");

            BoldText(RichTextBox1, "isEmpty()");

            BoldText(RichTextBox1, "iterator()");
```

```
ItalicText(RichTextBox1, "перевірка на наявність елементу E");
ItalicText(RichTextBox1, "повертає розмір списку");
ItalicText(RichTextBox1, "Перевіряє, чи пустий список");
ItalicText(RichTextBox1, "повертає ітератор для обходу елементів списку");

RegularText(RichTextBox1, "LinkedList<>();");
RegularText(RichTextBox1, "LinkedList<String> colors =");
RegularText(RichTextBox1, "LinkedList<Integer> numbers =");
RegularText(RichTextBox1, "LinkedList<String> names =");
RegularText(RichTextBox1, "LinkedList<String> fruits =");
RegularText(RichTextBox1, ".isEmpty()");
RegularText(RichTextBox1, ".iterator()");

ColorText(RichTextBox1, "\"Список містить червоний колір\"", Color.Green);
ColorText(RichTextBox1, "\"Список не містить червоний колір\"", Color.Green);
ColorText(RichTextBox1, "\"червоний\"", Color.Green);
ColorText(RichTextBox1, "\"зелений\"", Color.Green);
ColorText(RichTextBox1, "\"синій\"", Color.Green);
ColorText(RichTextBox1, "\"Розмір списку: \"", Color.Green);
ColorText(RichTextBox1, "\"Список порожній\"", Color.Green);
ColorText(RichTextBox1, "\"Список не порожній\"", Color.Green);
ColorText(RichTextBox1, "10", Color.Blue);
ColorText(RichTextBox1, "20", Color.Blue);
ColorText(RichTextBox1, "30", Color.Blue);
break;
```

```
            case 15:

                RichTextBox1.Text = "\tДодайте елемент \"манго\" в кінець списку
fruits.\n\nLinkedList<String> fruits = new
LinkedList<>();\nfruits.add(\"яблуко\");\nfruits.add(\"банан\");\nfruits.add(\"апельсин\");";



                radioButton1.Text = "fruits.addLast(\"манго\");";

                radioButton2.Text = "fruits.removeFirst();";

                radioButton3.Text = "fruits.add(1,\"манго\");";

                radioButton4.Text = "fruits.clear();";



                PForm();

                correctAnswer = 1;

                break;



            case 16:

                RichTextBox1.Text = "\tПеревірте, скільки елементів має список
fruits\n\nLinkedList<String> fruits = new
LinkedList<>();\nfruits.add(\"яблуко\");\nfruits.add(\"банан\");\nfruits.add(\"апельсин\");";



                radioButton1.Text = "System.out.println(fruits);";

                radioButton2.Text = "int size = fruits.size();\nSystem.out.println(\"Розмір списку: \" +
size)";

                radioButton3.Text = "fruits.add(1,\"полуниця\")";

                radioButton4.Text = "System.out.println(list.getLast())";



                PForm();

                correctAnswer = 2;

                break;
```

```
            case 17:

                RichTextBox1.Text = "\tОчистіть список fruits\n\nLinkedList<String> fruits = new
LinkedList<>();\nfruits.add(\"яблуко\");\nfruits.add(\"банан\");\nfruits.add(\"апельсин\");";



                radioButton1.Text = "System.out.println(fruits);";

                radioButton2.Text = "fruits.removeFirst();";

                radioButton3.Text = "fruits.clear();";

                radioButton4.Text = "System.out.println(list.getLast());";



                PForm();

                correctAnswer = 3;

                break;



            case 18:

                head2.Text = "Застосування Collections.LinkedList";

                RichTextBox1.Text = "\tCollections.LinkedList може використовуватися для реалізації
різних структур даних та алгоритмів, наприклад:\n\n• Черги (FIFO) - елементи додаються в кінець
списку, видаляються з початку;\n• Стеки (LIFO) - елементи додаються в кінець списку,
видаляються з кінця;\n\n• Двоспрямовані списки - LinkedList може використовуватися для
представлення двоспрямованих списків, де можна обходити список у двох напрямках: вперед і
назад. Це неможливо реалізувати ефективно за допомогою ArrayList;\n\n• Графи - LinkedList може
використовуватися для представлення вершин та ребер графа. Кожен вузол LinkedList може
містити інформацію про вершину графа, а посилання можуть використовуватися для
представлення зв'язків між вершинами.\n\n\tКоли використовувати
Collections.LinkedList?\nВикористовуйте LinkedList, коли вам потрібна структура даних, що
дозволяє:\n\n• Ефективно додавати та видаляти елементи в середині списку;\n• Динамічно
змінювати розмір списку;\n• Реалізувати чергу (FIFO) або стек (LIFO);\n• Представляти
двоспрямовані списки.\n\n\tКоли НЕ використовувати Collections.LinkedList?\nНе використовуйте
LinkedList, коли:\n\n• Вам потрібен швидкий доступ до елементів за індексом (в цьому випадку
ArrayList більш ефективний);\n• Пам'ять є обмеженим ресурсом (LinkedList використовує більше
пам'яті через додаткові посилання).";



                TForm();

                BoldText(RichTextBox1, "Двоспрямовані списки");

                BoldText(RichTextBox1, "Графи");
```

```
                BoldText(RichTextBox1, "Черги (FIFO)");
                BoldText(RichTextBox1, "Стеки (LIFO)");
                BoldText(RichTextBox1, "Коли використовувати Collections.LinkedList?");
                BoldText(RichTextBox1, "Коли НЕ використовувати Collections.LinkedList?");

                ItalicText(RichTextBox1, "структур");
                ItalicText(RichTextBox1, "алгоритмів");
                ItalicText(RichTextBox1, "Ефективно додавати та видаляти елементи в середині списку");
                ItalicText(RichTextBox1, "Динамічно змінювати розмір списку");
                ItalicText(RichTextBox1, "Реалізувати чергу (FIFO) або стек (LIFO)");
                ItalicText(RichTextBox1, "Представляти двоспрямовані списки");
                ItalicText(RichTextBox1, "Вам потрібен швидкий доступ до елементів за індексом ");
                ItalicText(RichTextBox1, "Пам'ять є обмеженим ресурсом ");
                break;
            case 19:
                RichTextBox1.Text = "\tКоли слід використовувати LinkedList?";

                radioButton1.Text = "Коли потрібен швидкий доступ до елементів за індексом;";
                radioButton2.Text = "Коли потрібна динамічна структура даних, що легко змінює розмір;";
                radioButton3.Text = "Коли потрібні великі списки;";
                radioButton4.Text = "Коли пам'ять є обмеженим ресурсом;";

                PForm();

                correctAnswer = 2;
```

```
                break;

            case 20:

                RichTextBox1.Text = "\tКоли НЕ слід використовувати LinkedList?";


                radioButton1.Text = "Коли потрібна динамічна структура даних, що легко змінює розмір;";

                radioButton2.Text = "Коли потрібна реалізація черги (FIFO) або стека (LIFO);";

                radioButton3.Text = "Коли потрібен швидкий доступ до елементів за індексом;";

                radioButton4.Text = "Коли потрібен невідомий заздалегідь розмір списку;";


                PForm();


                correctAnswer = 3;

                break;

            case 21:

                head2.Text = "Завершення";

                RichTextBox1.Text = "\tВітаємо з успішним проходженням тренажера з теми Collections.LinkedList. Під час його проходження Ви здобули такі знання та вміння як:\n\n• Основні знання про Collections.LinkedList, його структуру та особливості;\n\n• Слабкі та сильні сторони цієї структури даних, відносно її альтернатив;\n\n• Дізналися, для реалізації яких структур потрібен LinkedList;\n\n• Знання, як і коли краще використовувати LinkedList, а коли краще уникати його використовування;\n\n• Освоїли методи та конструктори потрібні для роботи з LinkedList;\n\n• Навчилися працювати з кодом на Java, котрий реалізує методи LinkedList\n\n\tСподіваюся цей тренажер був корисним для вас та допоміг освоїти та закріпити весь матеріал, котрий був поданий у рамках цієї програми. ";


                TForm();

                Final_window();

                BoldText(RichTextBox1, "Вітаємо");


                break;
```

```
        }
    }

    public void Form1_Resize(object sender, EventArgs e)
    {
        if (step == maxStep || step == 0)
        {
            CalculateStart();
        }
        else
        {
            CalculateStart();
            button2.Location = new Point(button3.Location.X, button3.Location.Y);
        }
    }

  }
}
```