ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ
«ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ БІЗНЕСУ ТА СУЧАСНИХ ТЕХНОЛОГІЙ

ФОРМА НАВЧАННЯ ДЕННА
КАФЕДРА МАТЕМАТИЧНОГО МОДЕЛЮВАННЯ ТА СОЦІАЛЬНОЇ ІНФОРМАТИКИ

Допускається до захисту

Завідувач кафедри _____О.О. Ємець
(підпис)

«_____»_____2021 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА ДО БАКАЛАВРСЬКОЇ РОБОТИ

на тему
Тренажер з теми «Мови, що задаються регулярними виразами» дистанційного навчального курсу «Теорія програмування» та розробка його програмного забезпечення

зі спеціальності 122 «Комп'ютерні науки»

Виконавець роботи Дзяткевич Максим Олегович
_____ «___»_____2021р.
(підпис)
Науковий керівник к.ф.-м.н., доц. Черненко Оксана Олексіївна
_____ «___»_____2021р.
(підпис)

ПОЛТАВА 2021 р.

ЗМІСТ

ВСТУП

Дистанційна форма та технології навчання є значно більш самостійною і індивідуалізованою формою навчання. Так, це важче ніж «пасивні» форми навчання у студентській аудиторії, проте, це значно ефективніше та комфортніше з огляду на витрати часу, коштів і зручностей, ніж інші форми та технології навчання. Дистанційне навчання можливе будь-де, будь-коли та доступне будь-кому і вже сьогодні стало важливою складовою віртуальної соціалізації особистості.

Дистанційне навчання надає доступ до нетрадиційних джерел інформації, підвищує ефективність самостійної роботи, дає абсолютно нові можливості для творчого самовираження, знаходження та закріплення професійних навичок.

Дистанційне навчання – нова організація освітнього процесу, сукупність інформаційних технологій, що забезпечують доставку студенту основного обсягу навчального матеріалу; інтерактивну взаємодію студентів та викладачів у процесі навчання; надання можливості самостійної роботи із засвоєння навчального матеріалу; а також прозоре оцінювання знань та умінь у процесі навчання.

Метою роботи є розробка тренажеру з теми «Мови, що задаються регулярними виразами» дистанційного навчального курсу «Теорія програмування» та розробка його програмного забезпечення.

Об’єктом розробки є процес дистанційного навчання математичним дисциплінам.

Предметом розробки є алгоритм роботи тренажеру з теми «Мови, що задаються регулярними виразами».

Перелік використаних методів – мова програмування Java, середовище розробки NetBeans.

Тренажер готовий до використання в дистанційному курсі «Теорія програмування».

Робота складається з трьох розділів.

Обсяг пояснювальної записки: 32 стор., в т.ч. основна частина - 30 стор., джерела - 7 назв.

1. ПОСТАНОВКА ЗАДАЧІ

Головне завдання даної роботи – є розробка алгоритму тренажеру за темою «Мови, що задаються регулярними виразами» дистанційного навчального курсу «Теорія програмування».

Планується, що розроблений у рамках курсової роботи алгоритм тренажеру буде запрограмовано та впроваджено як складова дистанційного курсу «Теорія програмування». Це накладає умову необхідності врахування можливості інтеграцій тренажеру до системи дистанційного навчання Moodle (на якій розміщуються дистанційні курси (ДК)).

Розглянемо етапи поставленої задачі:

- ознайомитися з теоретичним матеріалом;
- провести огляд прикладів;
- побудувати алгоритм тренажеру з теми «Мови, що задаються регулярними виразами»;
- обрати мову програмування;
- програмно реалізувати алгоритм;
- провести тестування та налагодження програмного продукту.

Мета алгоритму, що розробляється – це отримання практичних навичок студентами дистанційного курсу, тому перш за все, тренажер повинен бути зручним у користуванні. Необхідно розробити можливість студента звернутися до теоретичного матеріалу з теми, що допоможе йому у проходженні завдання.

Пропонується, щоб тренажер містив:

- стартову сторінку;
- сторінку з теоретичним матеріалом або окремий файл;
- сторінку з умовою прикладу, завданням, варіантами відповідей;
- кінцеву сторінку.

2. ТЕОРЕТИЧНА ЧАСТИНА

2.1. Огляд матеріалу з теми

«**Алфавіт –** це скінченна множина символів. Позначатимемо його X ».

Приклади

- Множина {0,1} являє собою бінарний алфавіт.

- ASCII і Unicode є прикладами комп'ютерних алфавітів.

Рядком у даному алфавіті називають скінченну послідовність символів з цього алфавіту. У теорії мов рядок ще називають ланцюжком, реченням, словом.

Символ ε позначає порожній рядок. Порожній рядок має нульову довжину.

Терміни “підрядок” і “підпослідовність” будемо розрізняти. Підрядок одержують видаленням початку і (або) кінця з рядка, підпослідовність – видаленням декількох символів, не обов'язково послідовних.

Множина всіх скінченних слів у алфавіті X позначається X^*. Зауважимо, що вона нескінченна. Вона містить ***порожнє слово*** – послідовність довжиною 0, позначену буквою ε. Множину $X^* \setminus \{\varepsilon\}$ позначимо X^+, а слово вигляду $ww..w$, де слово w із X^+ записано n разів – w^n. Вважатимемо, що $w^0 = \varepsilon$.

Довільна підмножина множини X^* називається ***формальною мовою***. Далі вона буде називатися просто *мовою*. **Мова –** це довільна множина рядків у деякому алфавіті.

Приклади.

1. «Множина всіх слів у алфавіті $\{a\}$ позначається $\{a\}^* = \{\varepsilon, a, aa, aaa, ...\}$ $= \{a^n \mid n \geq 0\}$. $\{a^n \mid n -$ непарне $\}$ позначає множину, або мову слів непарної довжини в алфавіті $\{a\}$; обидві мови нескінченні.»

2. «Ідентифікатор є послідовністю букв і цифр, що починається буквою. Множина всіх ідентифікаторів у алфавіті $X = \{a, b, 1\}$ нескінченна. Якщо записати їх за зростанням довжини, то початок буде таким: $\{a, b, a1, aa, ab, b1, ba, bb, ...\}$.»

Задача перевірки, чи належить слово w мові L, називається ***задачею належності***, або ***проблемою слів***. Як правило, множина L задається певним ***скінченним описом***, що визначає не тільки її саму, а й структуру її елементів.

Задача належності розв'язується найчастіше шляхом перевірки, чи має слово відповідну структуру, тобто шляхом ***синтаксичного аналізу***, або ***розпізнавання***. Наприклад, структура всіх можливих синтаксично правильних Паскаль-програм визначається скінченною та відносно невеликою сукупністю БНФ. Саме на її основі будуються ***синтаксичні аналізатори*** в трансляторах, тобто програми аналізу синтаксичної правильності вхідних програм.

Формальні мови розглядатимуться далі як мови, задані саме *скінченним описом*. Отже, *головним у вивченні формальних мов стає засіб їх задання* [2].

Регулярна мова (регулярна множина) — формальна мова третього (найвужчого) класу в ієрархії Чомскі.

Регулярну мову можна задати регулярною граматикою або регулярним виразом, або ж ДСкА чи НДСкА, що її розпізнають.

Від контекстно-вільних мов регулярні відрізняються додатковими умовами: права частина правил виведення має бути порожнім словом,

термінальним, нетермінальним, або нетермінальний вслід за яким стоїть термінальний символ.

Для визначення приналежності мови до класу регулярних існує Лема про накачку для регулярних мов та теорема Майхілла-Нероде.

Нехай Σ — скінченний алфавіт. Регулярна мова в цьому алфавіті визначається рекурсивно:

1. ∅ — пуста множина — це регулярна множина в алфавіті Σ
2. ε — пусте слово — це регулярна множина в алфавіті Σ
3. $\{a\}$, $a \in \Sigma$ — однолітерна множина — регулярна множина в алфавіті Σ
4. Якщо P та Q — регулярні множини, то такими є наступні множини:
   - $P \cup Q$ (операція об‘єднання);
   - $P * Q$ (операція конкатенації);
   - $\{P\} = \varepsilon \cup P \cup P * P \cup \ldots$ (операція ітерації).
5. Ніякі інші множини, окрім побудованих на основі ПП 1-4 не є регулярними множинами.

«Стандартними» способами опису регулярної мови є представлення її у вигляді скінченного автомату, регулярної граматики, або ж регулярного виразу.

Формальна мова є регулярною тоді й лише тоді, коли існує детермінований скінченний автомат здатний її розпізнати (акцептор).

При чому, для кожної регулярної мови можна побудувати недетермінований скінченний автомат, що її розпізнає. Та навпаки, кожен недетермінований скінченний автомат розпізнає певну регулярну мову.

Регулярні мови замкнені відносно операцій об'єднання, перетину, конкатенації, доповнення та зірочки Кліні. Тобто, якщо L_1 та L_2 регулярні мови, тоді регулярними будуть мови: $L_1 \cup L_2$, $L_1 \cap L_2$, L_1L_2, $L_1\emptyset$, та L_1^*.

Існують й інші операції, відносно яких замкнені регулярні мови. Так, наприклад, гомоморфне відображення регулярної мови також є

регулярною мовою. Таким чином, регулярні мови замкнені відносно гомоморфізму.

Крім того, для будь-якої регулярної мови у стандартному представленні існує алгоритм, що визначає приналежність заданого слова до цієї мови[6]. Стандартне представлення регулярної мови також дає можливість створити алгоритм, який би перевіряв чи ця мова порожня, скінченна, або ж нескінченна.

Стандартне представлення регулярних мов також дає можливість створити алгоритм, який би перевіряв їх на рівність [3].

2.2. Алгоритмізація задачі за темою роботи

На стартовій сторінці повинна виводитися тема, інформація про студента, академічна група студента. Пропонується перейти до виконання завдань.

Крок 1. Користувачу виводиться умова: «Алфавіт – це».

Потрібно вибрати один з варіантів:

- скінченна множина символів.
- нескінченна множина символів.
- множина символів.

Якщо користувач вибрав перший варіант, то перехід відбувається на наступний крок. В іншому разі виводиться пояснення про помилку.

Крок 2. Користувачу виводиться умова: «Множина {0,1} являє собою».

Потрібно вибрати один з варіантів:

- комп'ютерний алфавіт.
- бінарний алфавіт.
- бінарний і комп'ютерний алфавіти.

Якщо користувач вибрав другий варіант, то перехід відбувається на наступний крок. В іншому разі виводиться пояснення про помилку.

Крок 3. Користувачу виводиться умова: «ASCII і Unicode є прикладами».

Потрібно вибрати один з варіантів:

- комп'ютерних алфавітів.
- бінарних алфавітів.
- бінарного і комп'ютерного алфавітів.

Якщо користувач вибрав перший варіант, то перехід відбувається на наступний крок. В іншому разі виводиться пояснення про помилку.

Крок 4. Користувачу виводиться умова: «Рядком у алфавіті називають».

Потрібно вибрати один з варіантів:

- скінченну послідовність символів.
- нескінченну послідовність символів з цього алфавіту.
- скінченну послідовність символів з цього алфавіту.

Якщо користувач вибрав третій варіант, то перехід відбувається на наступний крок. В іншому разі виводиться пояснення про помилку.

Крок 5. Користувачу виводиться умова: «Символ _1_ позначає порожній рядок. Порожній рядок має __2__ довжину».

Заповніть пусті комірки. Для 1:

- ε.
- 0.
- X^*.

Для 2:

- скінченну.
- нескінченну.
- нульову.

Якщо відповідь: для 1 перший, для 2 третій варіанти, то перехід відбувається на наступний крок. В іншому разі виводиться пояснення про помилку.

Крок 6. Користувачу виводиться умова: «Терміни “підрядок” і “підпослідовність” будемо розрізняти».

Продовжіть речення:

- Підрядок одержують
- Підпослідовність одержують

Варіанти відповіді:

- видаленням декількох символів, не обов'язково послідовних.
- видаленням початку і (або) кінця з рядка, підпослідовність.

Якщо відповідь: для першого речення перший варіант, а для другого – перший варіант, то перехід відбувається на наступний крок. В іншому разі виводиться пояснення про помилку.

Крок 7. Користувачу виводиться умова: «Множина всіх скінченних слів у алфавіті X позначається».

Потрібно вибрати один з варіантів:

- X^+.
- X^*.
- X.

Якщо відповідь: другий варіант, то перехід відбувається на наступний крок. В іншому разі виводиться пояснення про помилку.

Крок 8. Користувачу виводиться умова: «Множина всіх скінченних слів X^* містить порожнє слово – послідовність довжиною _1_, позначену буквою __2__».

Заповніть пусті комірки. Для 1 потрібно ввести значення.

Для 2:

- ε.
- 0.
- X^0.

Якщо відповідь: для 1 – 0, для 2 перший варіант, то перехід відбувається на наступний крок. В іншому разі виводиться пояснення про помилку.

Крок 9. Користувачу виводиться умова: «Мова – це».

Потрібно вибрати один з варіантів:

- довільна множина рядків.
- рядки у деякому алфавіті.
- довільна множина рядків у деякому алфавіті.

Якщо користувач вибрав третій варіант, то перехід відбувається на наступний крок. В іншому разі виводиться пояснення про помилку.

Крок 10. Користувачу виводиться умова: «Задача перевірки, чи належить слово w мові L, називається задачею належності, або проблемою слів. Як правило, множина L задається певним _____, що визначає не тільки її саму, а й структуру її елементів».

Заповніть пусту комірку:

- скінченним описом.
- нескінченним описом.

Якщо користувач вибрав перший варіант, то перехід відбувається на наступний крок. В іншому разі виводиться пояснення про помилку.

Крок 11. Користувачу виводиться умова: «Задача належності розв'язується найчастіше шляхом перевірки, чи має слово відповідну структуру, тобто шляхом».

Потрібно вибрати один з варіантів:

- синтаксичного аналізу.
- розпізнавання.
- аналізу.

Якщо користувач вибрав перший або другий варіант, то виводиться повідомлення про ознайомлення з теорією, пропонується перейти до прикладів. Якщо так – перехід на наступний крок, інакше – виводиться стартова сторінка.

Приклади.

Крок 1. Користувачу виводиться умова: «Множина всіх слів у алфавіті $\{a\}$ позначається».

Вкажіть 4 перших слова (символ ε заміняти на e).

$$\{a\}^* = \{\_\_\_\_\_,....\}.$$

Якщо відповідь: $\{e, a, aa, aaa\}$, то перехід на наступний крок. В іншому разі виводиться пояснення про помилку.

Крок 2. Користувачу виводиться умова: «Множина всіх слів у алфавіті $\{a\}$ позначається».

Заповніть пусті комірки.

$$\{a\}^* = \{\varepsilon, a, aa, aaa,....\} = \{\underset{1}{\_\_\_}^{\,\underset{2}{\_\_\_}} \mid \underset{3}{\_\_\_}\ \underset{4}{\_\_\_}\}.$$

Для 1, 2, 4 потрібно ввести значення.

Для 3:

- $\geq$
- $\leq$
- $=$

Якщо відповідь: $\{a^n \mid n \geq 0\}$, то перехід на наступний крок. В іншому разі виводиться пояснення про помилку.

Крок 3. Користувачу виводиться умова: « $\{a^n \mid n -$ непарне $\}$ позначає».

Потрібно вибрати один з варіантів:

- множину непарної довжини в алфавіті $\{a\}$.
- мову слів непарної довжини в алфавіті $\{a\}$.
- множину, або мову слів непарної довжини в алфавіті $\{a\}$.

Якщо користувач вибрав третій варіант, то перехід на наступний крок. В іншому разі виводиться пояснення про помилку.

Крок 4. Користувачу виводиться умова: «Множина всіх слів у алфавіті $\{a\}$ позначається $\{a\}^* = \{\varepsilon, a, aa, aaa, ...\} = \{a^n \mid n \geq 0\}$. $\{a^n \mid n -$ непарне$\}$ позначає множину, або мову слів непарної довжини в алфавіті $\{a\}$ ».

Потрібно вибрати один з варіантів:

- обидві мови нескінченні.
- обидві мови скінченні.
- перша нескінченна, друга скінченна.

Якщо користувач вибрав перший варіант, то перехід на наступний крок. В іншому разі виводиться пояснення про помилку.

Крок 5. Користувачу виводиться умова: «Ідентифікатор є послідовністю букв і цифр, що починається буквою. Множина всіх ідентифікаторів у алфавіті $X = \{a, b, 1\}$».

Потрібно вибрати один з варіантів:

- скінченна.
- нескінченна.

Якщо користувач вибрав другий варіант, то перехід на наступний крок. В іншому разі виводиться пояснення про помилку.

Крок 6. Користувачу виводиться умова: «Якщо записати множину всіх ідентифікаторів у алфавіті $X = \{a, b, 1\}$ за зростанням довжини, то початок буде таким».

Вкажіть 8 перших слів (символ ε заміняти на e).

{_____,...}.

Якщо відповідь: $\{a, b, a1, aa, ab, b1, ba, bb, ...\}$, то перехід на наступний крок. В іншому разі виводиться пояснення про помилку.

Крок 7. Користувачу виводиться умова: «Нехай Σ — скінченний алфавіт. Регулярна мова в цьому алфавіті визначається рекурсивно:».

Встановіть послідовність правил.

- Якщо P та Q — регулярні множини, то такими є наступні множини: $P \cup Q$ (операція об‘єднання); $P * Q$ (операція конкатенації); $\{P\} = \varepsilon \cup P \cup P * P \cup \ldots$ (операція ітерації).
- $\emptyset$ — пуста множина — це регулярна множина в алфавіті Σ
- ε — пусте слово — це регулярна множина в алфавіті Σ
- Ніякі інші множини, окрім побудованих на основі ПП 1-4 не є регулярними множинами.
- $\{a\}, \quad a \in \Sigma$ — однолітерна множина — регулярна множина в алфавіті Σ

Якщо відповідь:

1. $\emptyset$ — пуста множина — це регулярна множина в алфавіті Σ
2. ε — пусте слово — це регулярна множина в алфавіті Σ
3. $\{a\}, \quad a \in \Sigma$ — однолітерна множина — регулярна множина в алфавіті Σ
4. Якщо P та Q — регулярні множини, то такими є наступні множини: $P \cup Q$ (операція об‘єднання); $P * Q$ (операція конкатенації); $\{P\} = \varepsilon \cup P \cup P * P \cup \ldots$ (операція ітерації).
5. Ніякі інші множини, окрім побудованих на основі ПП 1-4 не є регулярними множинами.

то перехід на наступний крок. В іншому разі виводиться пояснення про помилку.

Крок 8. Виводиться кінцевий результат і пропонується повернутися на стартову сторінку або закрити тренажер.

3. ПРАКТИЧНА ЧАСТИНА

3.1. Розробка блок-схеми, яка підлягає програмуванню

На рисунку 3.1 зображено блок-схему алгоритму роботи тренажеру.

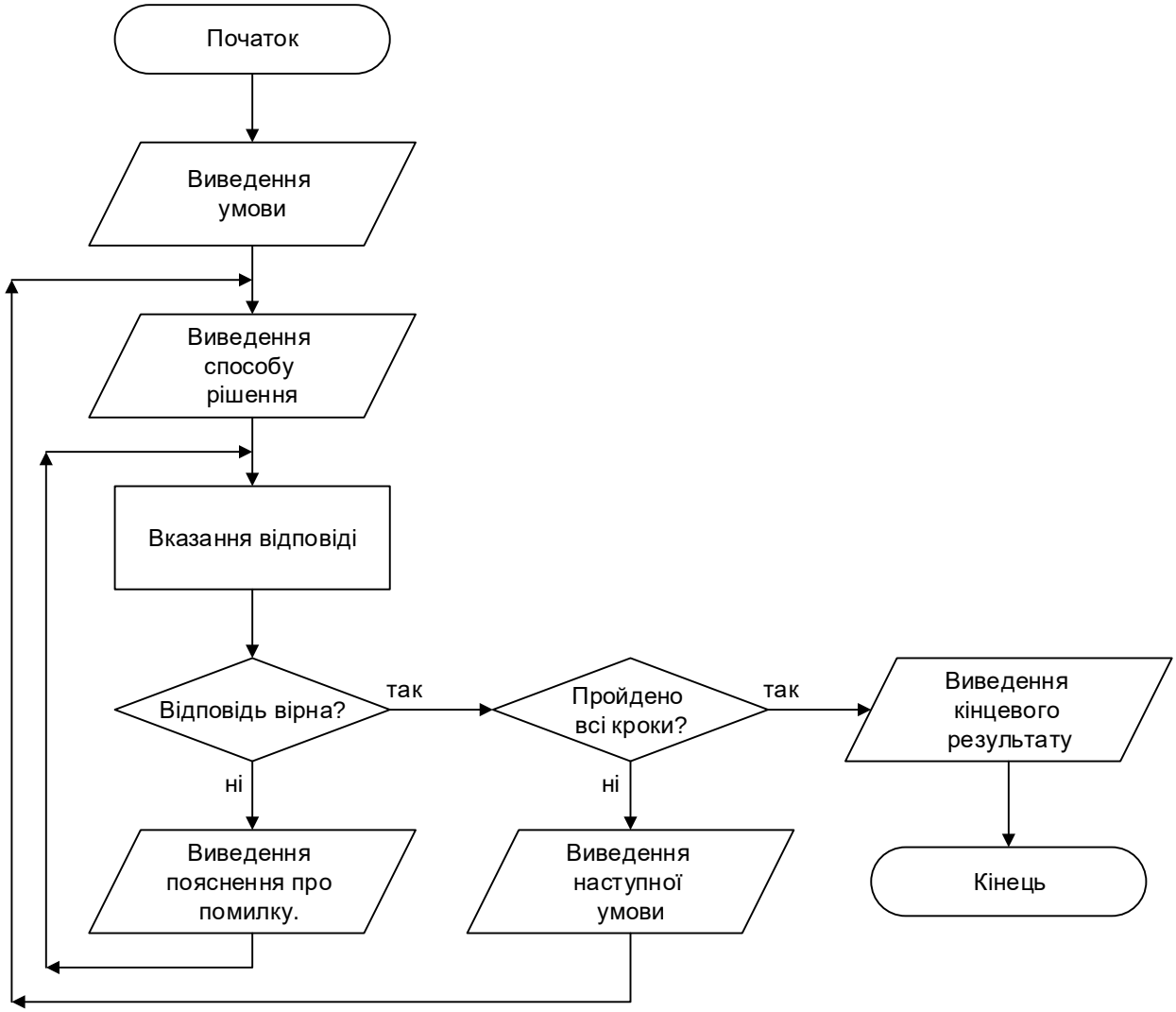

Рисунок 3.1 – Блок-схема алгоритму роботи тренажеру

3.2. Обґрунтування вибору програмних засобів для реалізації завдання роботи

Java (вимовляється Джава; інколи — Ява) — об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java.

Розробники Javaстворювали мову з метою втілення таких базових принципів:

◊простота;

◊безпека;

◊перенесення;

◊об’єктозорієнтована спрямованість;

◊стійкість щодо помилок;

◊багатопоточність;

◊незалежність від архітектури;

◊інтерпретація;

◊висока продуктивність;

◊розподіленість;

◊динамічність.

До простих мов програмування належать ті, які працюють з інтерпретатором (наприклад, Бейсік). Перші персональні комп’ютери поставляли з інтерпретатором Бейсіка. Сьогодні їхнє місце займають HTMLі мови сценаріїв для Web. Вивчати і використовувати мови програмування, які компілю-ються, набагато складніше, ніж ті, які інтерпретуються. Тобто є мови програмування для професіоналів. Наприклад, C++, де використання покажчиків і керування пам’яттю є складними не тільки для початківців, але й для досвідчених програмістів. Один оператор програми, який звертається до недозволеного місця в пам’яті, може спричинити до збоїв не тільки програми, але й комп’ютера загалом. Java− це мова, програми якою компілюються та інтерпретуються, а водночас вона має просту структуру мови високого рівня. Написана програма компілюється у проміжну форму − байткод.

Пізніше ця програма виконується, тобто інтерпретується виконавчим середовищем Java.

Байткод створено для машини, яка реально не існує. Цю машину називають віртуальною Java-машиною (JVM), вона існує тільки в пам'яті комп'ютера. Створення компілятором Javaбайткоду для неіснуючої машини − це тільки половина процесу, який забезпечує незалежність від архітектури. Другу частину виконує інтерпретатор Java, який виконує роль посередника між віртуальною Java-машиною та реальним комп'ютером.

Головна вимога щодо мови для роботи в розподі-леному просторі комп'ютерів(наприклад, в Internet) − це можли-вість працювати на різноманітних і розподілених платформах. Мова Javaє пристосованою до перенесення завдяки під-тримці стандартів IEEE для структур даних, наприклад, цілих чисел, чисел з плаваючою комою і рядків. До мови Javazачислено безпосередньо підтримку таких розповсюджених протоколів як FTP, HTTP, що забезпечує суміс-ність під час роботи в мережі.Javazабезпечує розподілену роботу за допомогою механізму виклику віддалених методів (RMI), тобто дає змогу використо-вувати об'єкти, розташовані на локальних і віддалених машинах.

На противагу C++, Java об'єктно-орієнтованіша. Всі дані і дії групуються в класи об'єктів. Виключенням з повної об'єктності (як скажімо в Smalltalk) є примітивні типи (int, float тощо). Це було свідомим рішенням проектувальників мови задля збільшення швидкості. Через це Java не вважається повністю об'єктно-орієнтовною мовою.

У Java всі об'єкти є похідними від головного об'єкта (він називається просто Object), з якого вони успадковують базову поведінку і властивості.

Хоча у C++ вперше стало доступне множинне успадкування, але у Java можливе тільки одинарне успадкування, завдяки чому виключається можливість конфліктів між членами класу (методи і змінні), які успадковуються від базових класів.

У намірах проектувальників Java мала замінити C++ — об'єктного наступника мови C. Проектувальники почали з аналізу властивостей C++, які є причиною найбільшого числа помилок, щоби створити просту, безпечну і безвідмовну мову програмування.

В Java існує система винятків або ситуацій, коли програма зустрічається з неочікуваними труднощами, наприклад:

- операції над елементом масиву поза його межами або над порожнім елементом
- читання з недоступного каталогу або неправильної адреси URL
- ввід недопустимих даних користувачем

Одна з особливостей концепції віртуальної машини полягає в тому, що помилки (виключення) не призводять до повного краху системи. Крім того, існують інструменти, які «приєднуються» до середовища періоду виконання і кожен раз, коли сталося певне виключення, записують інформацію з пам'яті для відлагодження програми. Ці інструменти автоматизованої обробки виключень надають основну інформацію щодо виключень в програмах на Java.

Проте мову програмування Java не рекомендується використовувати в системах, збій в роботі яких може призвести до смерті, травм чи значних фізичних ушкоджень (наприклад, програмне забезпечення для керування атомними електростанціями, польотами, систем життєзабезпечення чи систем озброєння) через ненадійність програм, написаних на мові програмування Java., пункт ліцензії Microsoft 7.7.h.

«Java використовує автоматичний збирач сміття для керування пам'яттю під час життєвого циклу об'єкта. Програміст вирішує, коли створювати об'єкти, а віртуальна машина відповідальна за звільнення пам'яті після того, як об'єкт стає непотрібним. Коли до певного об'єкта вже не залишається посилань, збирач сміття може автоматично прибирати його із пам'яті» [5]. Проте, витік пам'яті все ж може статися, якщо код, написаний

програмістом, має посилання на вже непотрібні об'єкти, наприклад на об'єкти, що зберігаються у діючих контейнерах.

Збирання сміття дозволене у будь-який час. В ідеалі воно відбувається під час бездіяльності програми. Збірка сміття автоматично форсується при нестачі вільної пам'яті в купі для розміщення нового об'єкта, що може призводити до кількасекундного зависання. Тому існують реалізації віртуальної машини Java з прибиральником сміття, спеціально створеним для програмування систем реального часу.

Java не має підтримки вказівників у стилі C/C++. Це зроблено задля безпеки й надійності, аби дозволити збирачу сміття переміщувати вказівникові об'єкти [5].

3.3. Опис процесу програмної реалізації

Для створення проекту IDE виконайте наступні дії:

1. Запустіть IDE NetBeans.

2. В меню IDE виберіть 'Файл> Створити проект', як показано на рис. 3.2.

3. У майстрі створення проекту розгорніть категорію "Java" і виберіть "Додаток Java". Потім натисніть кнопку "Далі".

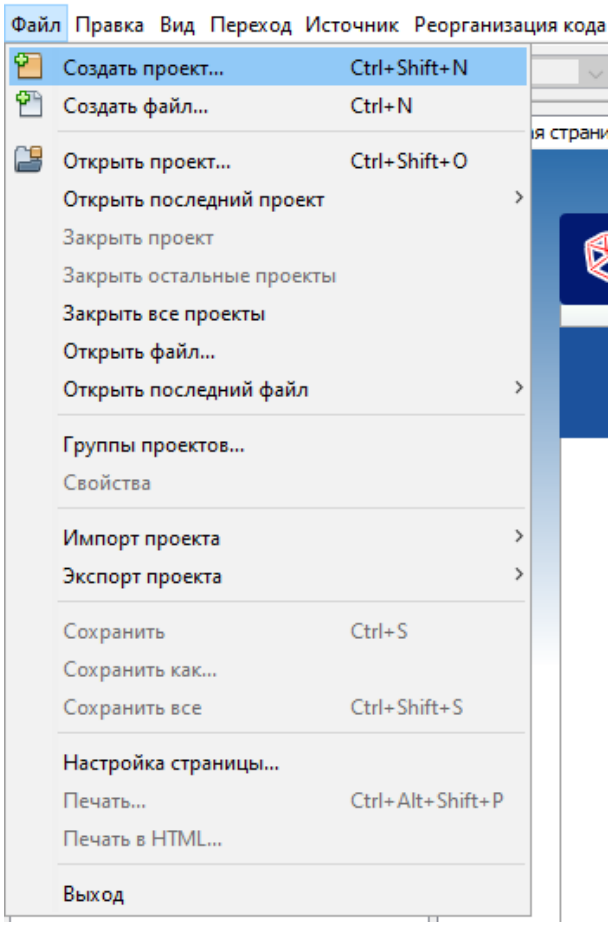

Рисунок 3.2 – 'Файл> Створити проект'

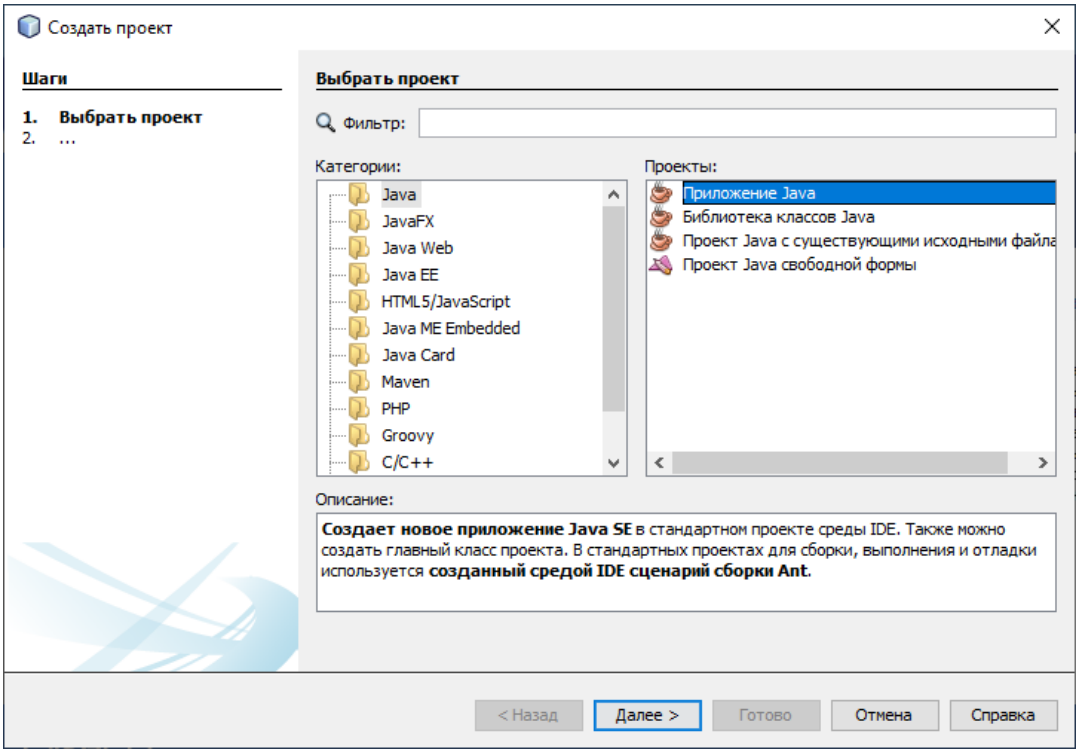

Рисунок 3.3 – Майстер створення проекту

4. На сторінці "Ім'я і місце розташування" виконайте наступні дії:

- введіть назву в поле "Ім'я проекту";
- вкажіть місце розташування;
- Не встановлюйте прапорець "Використовувати окрему папку для зберігання бібліотек".
- в поле "Створити головний клас" введіть назву класу;

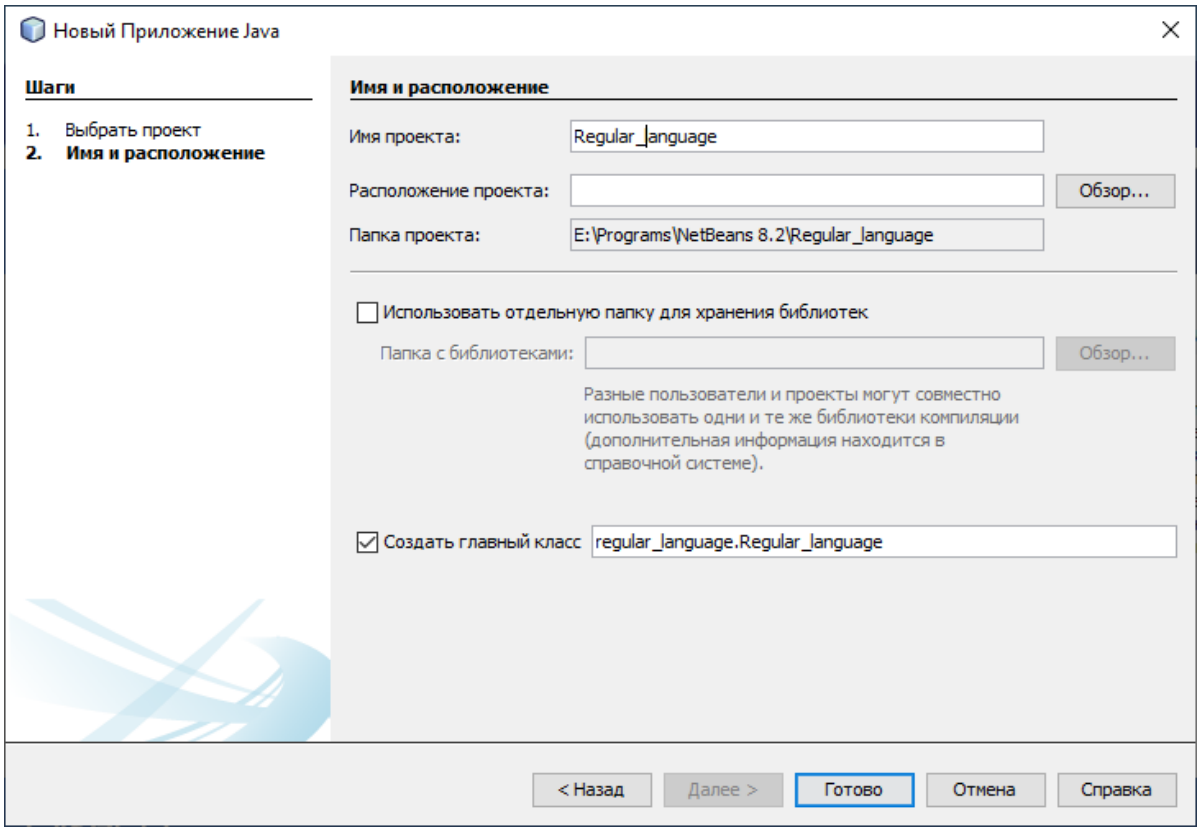

Рисунок 3.4 – Сторінка "Ім'я і місце розташування"

5. Натисніть кнопку "Завершити".

Проект буде створений і відкритий в середовищі IDE. На екрані повинні бути представлені наступні елементи:

- вікно "Проекти", яке містить дерево елементів проекту, в тому числі вихідні файли, бібліотеки, від яких залежить код, і т.д .;
- вікно редактора вихідного коду з відкритим файлом;
- вікно "Переходи", яке можна використовувати для швидкого переміщення між елементами всередині вибраного класу.

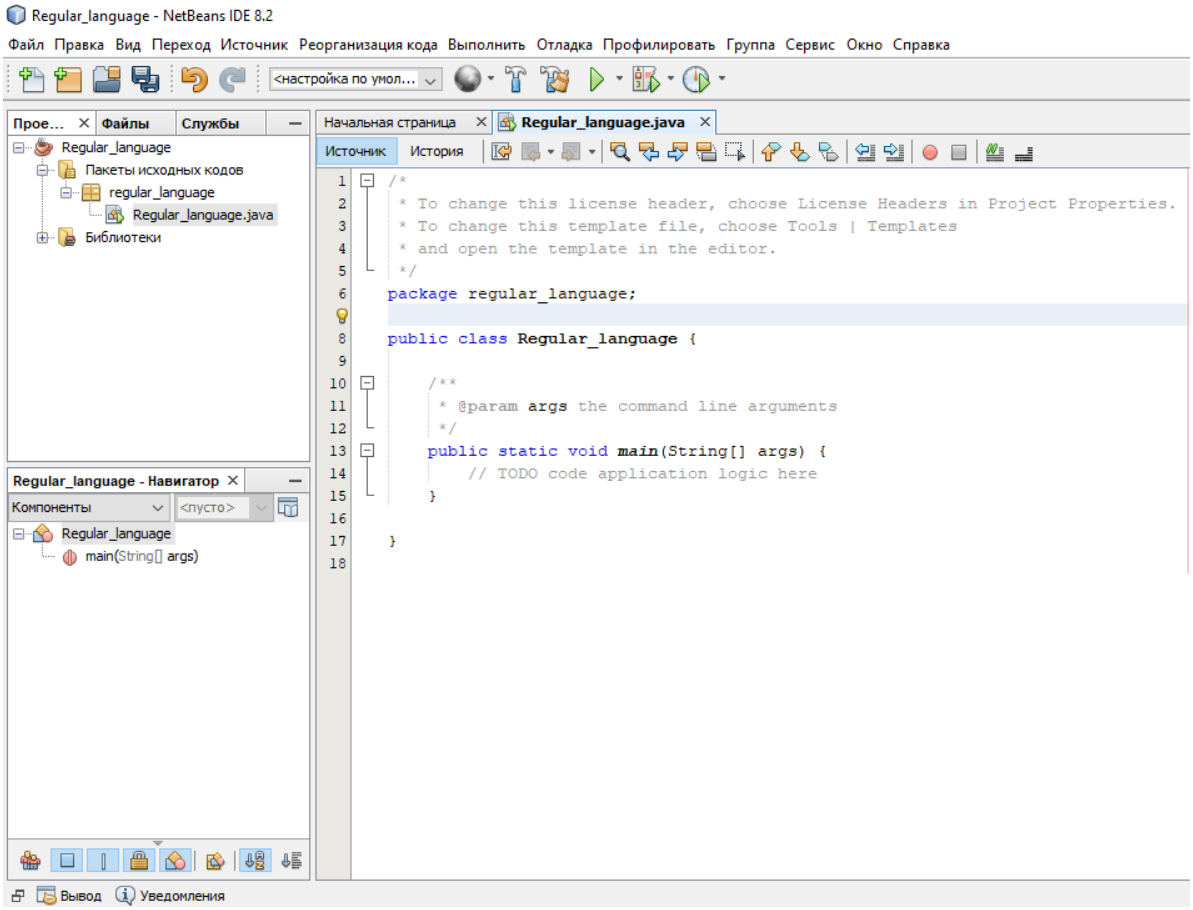

Рисунок 3.5 – Середовищі IDE

Створення файлу вихідного коду аплету

1. Натисніть вузол в вікні "Проекти" і виберіть "Новий"> "Інше" (Ctrl-N).

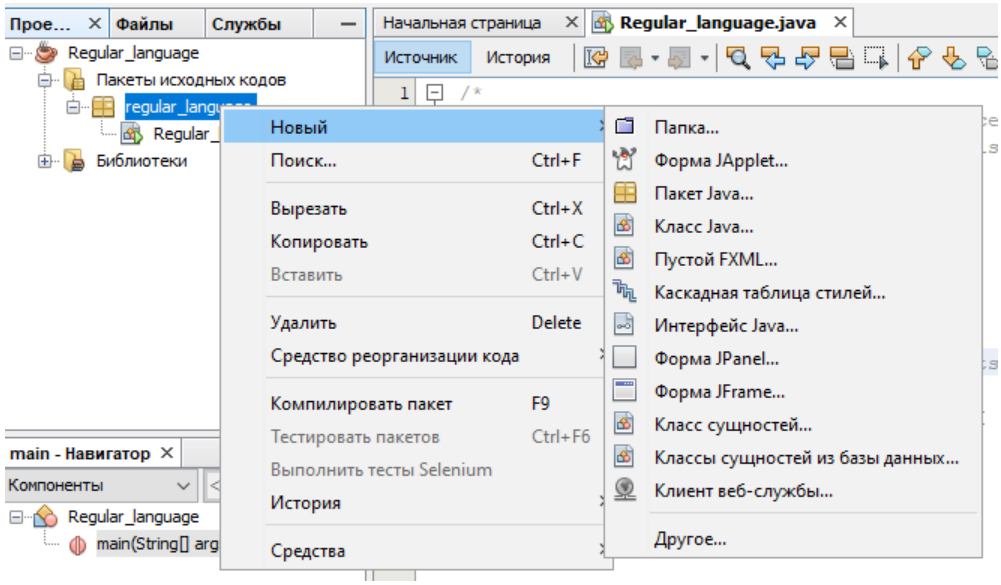

Рисунок 3.6 – Створення аплету

2. В області "Категорії" виберіть "Java". В області "Типи файлів" виберіть "Аплет".

Якщо ж потрібно використовувати візуальні засоби для розробки аплету, виберіть "Форми Swing GUI"> "Форма JApplet".

Натисніть кнопку "Далі".

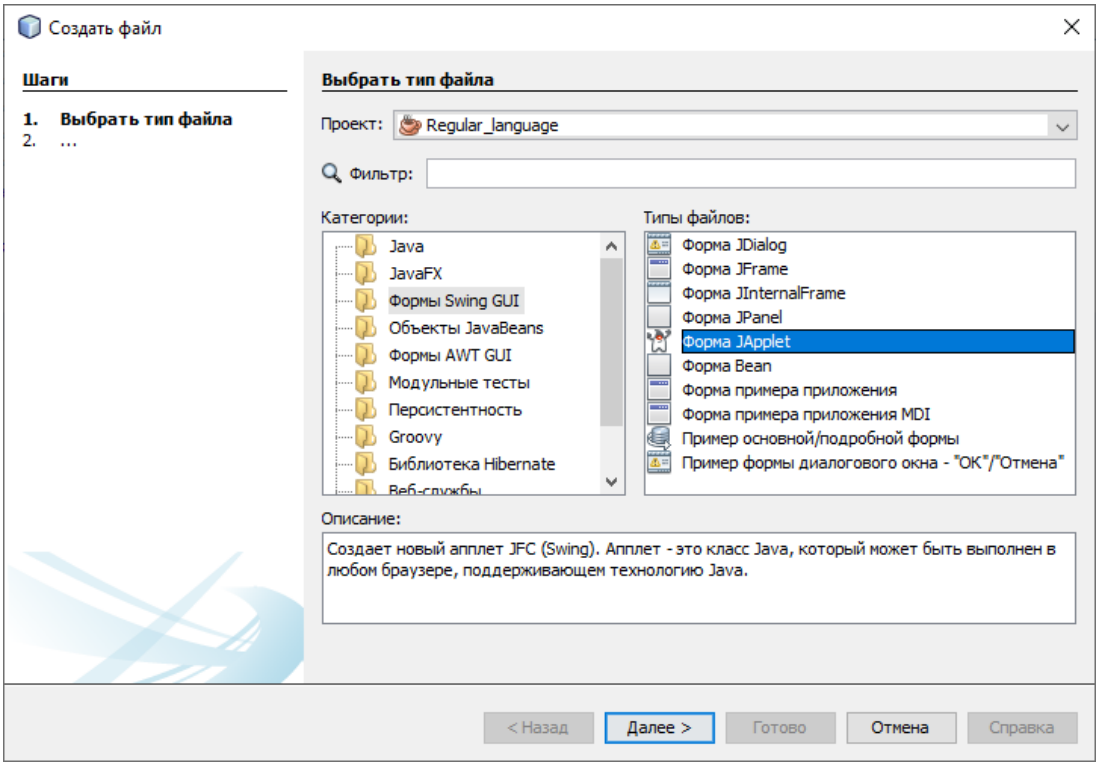

Рисунок 3.7 – Вибір типу файлу

3. В області "Ім'я класу" введіть ім’я.

4. Натисніть кнопку "Завершити".

Середовище IDE створює вихідний файл аплет в зазначеному пакеті. Вихідний файл аплету відкривається в редакторі вихідного коду.

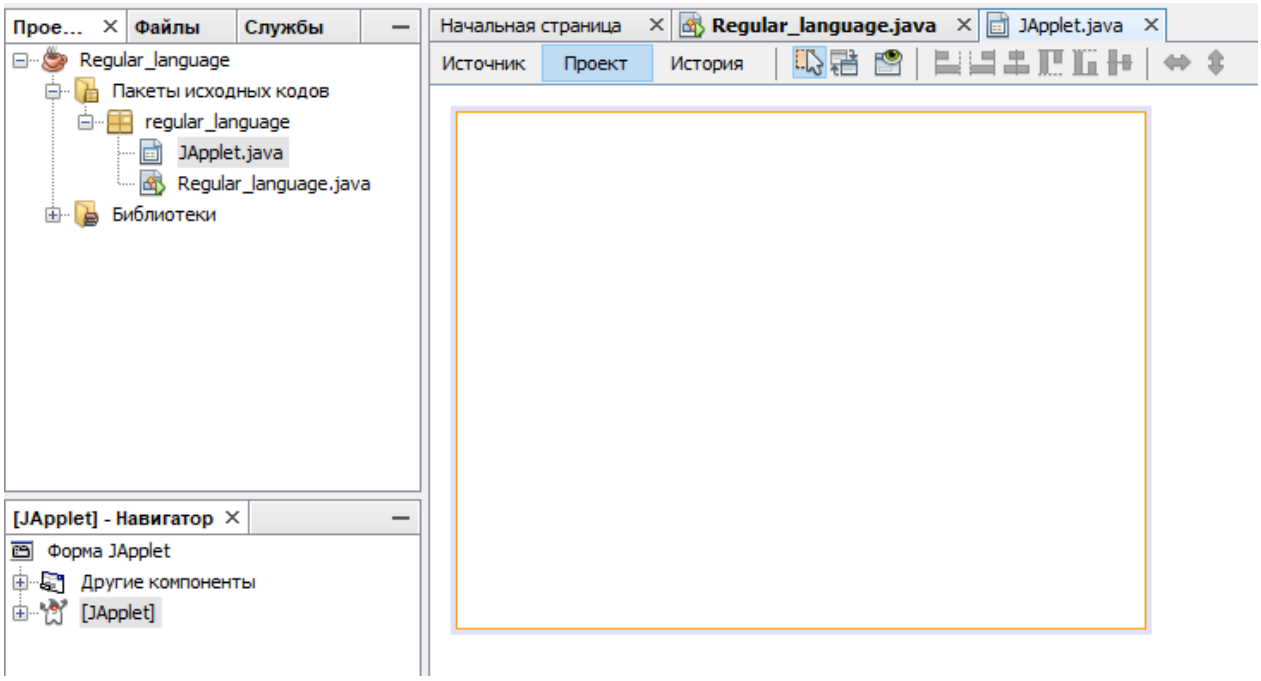

Рисунок 3.8 – Вихідний файл аплету

Після цього за допомогою контейнера Панель було розроблено структуру тренажеру.

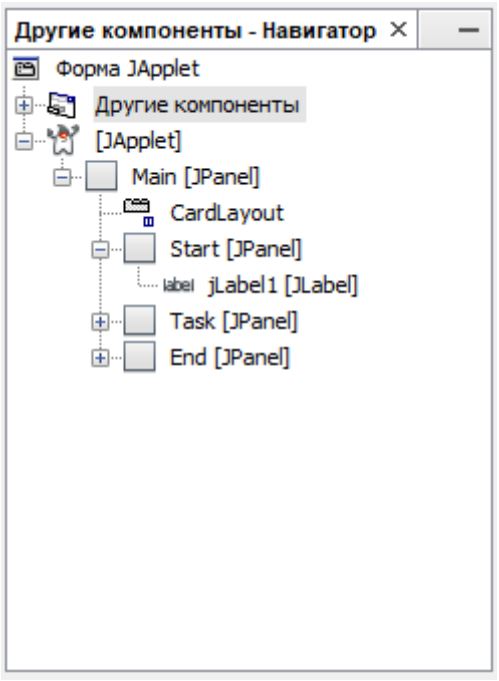

Рисунок 3.9 – Структура тренажеру

За допомогою графічних елементів панелі з інструментами (рис. 3.10) можна створити свій дизайн (рис. 3.11).

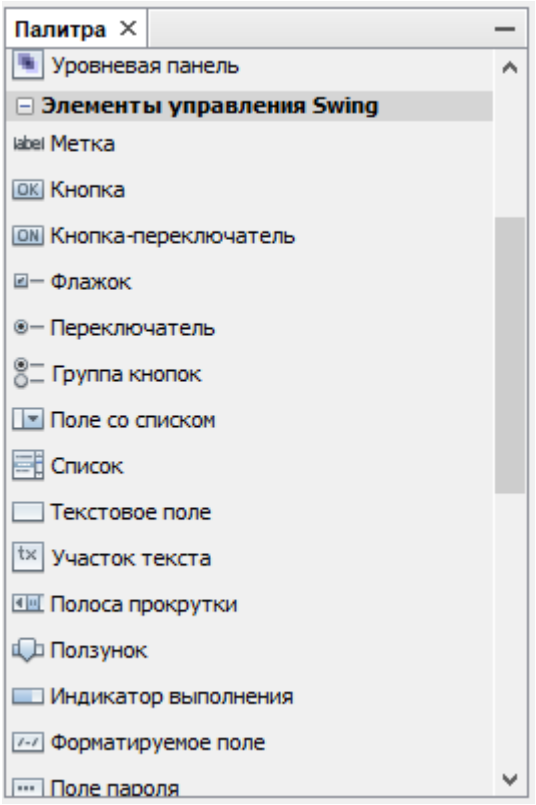

Рисунок 3.10 – Панель з інструментами

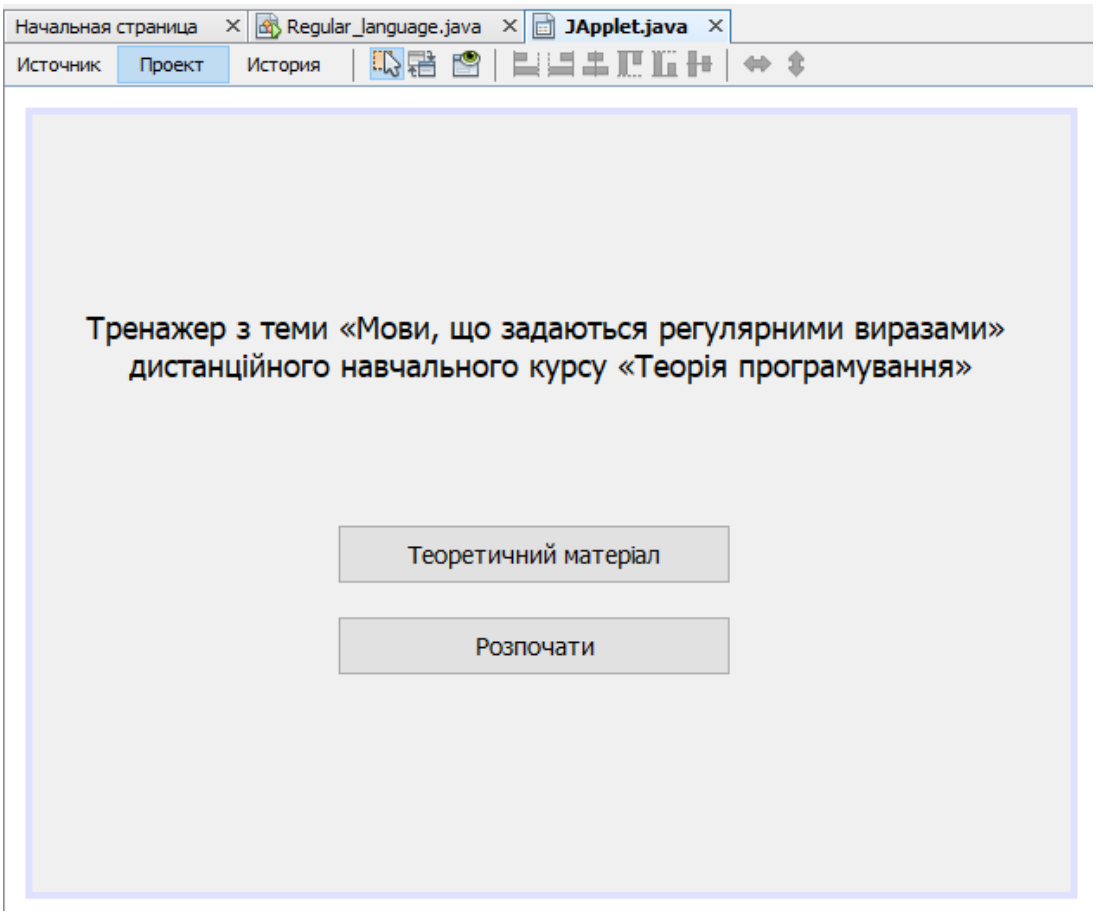

Рисунок 3.11 – Розміщення графічних елементів

Для запуску тренажеру залишилось в головному класі прописати відповідний код.

```
public static void main(String[] args) {
    JFrame frame = new JFrame("Тренажер");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JApplet applet = new JApplet();
    frame.getContentPane().add(applet);
    applet.init();
    applet.start();
    frame.pack();
    frame.setSize(600, 470);
    frame.setResizable(false);
    frame.setLocationRelativeTo(null);
    frame.setVisible(true);
}
```

ВИСНОВКИ

Одним з важливих етапів навчального процесу є доступ до навчального матеріалу, а відповідно матеріал має бути максимально легким до сприйняття і оптимально наповнений смисловим значенням і не переобтяжений зайвою інформацією.

Створення так званих віртуальних тренажерів – важливий етап при вирішенні проблеми організації навчання, де необхідні практичні навички. Основна перевага застосування віртуальних тренажерів в тому, що вони можуть використовуватися як в навчальному процесі (при проведенні лабораторних робіт або для здійснення теоретичного допуску до них), так і для самостійного навчання студентів. Застосування розробленої методики віртуальних практичних інтерактивних засобів навчальних дисциплін для дистанційного навчання дає змогу вирішити проблему впровадження інформаційних дистанційних технологій у навчальний процес з багатьох напрямів і спеціальностей підготовки.

Саме тому в рамках курсової роботи ставилася задачі по розробці алгоритму тренажера з теми «Мови, що задаються регулярними виразами» дистанційного навчального курсу «Теорія програмування» та його програмування.

Розглянемо зроблені етапи поставленої задачі:

- ознайомлено з теоретичним матеріалом;
- проведено огляд прикладів;
- побудовано алгоритм тренажеру з теми «Мови, що задаються регулярними виразами»;
- складено блок-схему алгоритму:
- обрано мову програмування.

Тренажер містить:

- стартову сторінку;
- сторінку з теоретичним матеріалом або окремий файл;

- сторінку з умовою прикладу, завданням, варіантами відповідей;
- кінцеву сторінку.

На стартовій сторінці виводитися тема тренажеру. Пропонується перейти до виконання завдань.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ємець О. О. Методичні рекомендації щодо оформлення пояснювальних записок до курсових проектів (робіт) для студентів за освітньою програмою «Комп'ютерні науки» спеціальності 122 «Комп'ютерні науки та інформаційні технології», «Комп'ютерні науки» галузь знань - 12 «Інформаційні технології» / О. О. Ємець – Полтава : РВВ ПУЕТ, 2017. – 69с.
2. Черненко О.О. Електронний навчально-методичний посібник для самостійного вивчення навчальної дисципліни «Теорія програмування» для студентів напряму 6.040302 «Інформатика»
3. Регулярна мова [Електронний ресурс] / Матеріал з Вікіпедії — вільної енциклопедії. – Режим доступу: https://uk.wikipedia.org/wiki/Регулярна_мова
4. Нікітченко М.С. Теоретичні основи програмування: Навчальний посібник [Електронний ресурс] / М.С. Нікітченко. – Київ: КНУ ім. Т.Г. Шевченка, 2009. – 200 с. – Режим доступу: http://ttp.unicyb.kiev.ua/doc/TOP.pdf.
5. Java [Електронний ресурс] / Матеріал з Вікіпедії — вільної енциклопедії.. – Режим доступу: http://uk.wikipedia.org/wiki/Java.
6. Герман О.В. Программирование на Java и C# для студента / О.В. Герман, Ю.О. Герман. – СПб. : БХВ-Петербург, 2005. – 512 с.
7. Бібліографічний запис. Бібліографічний опис. Загальні вимоги та правила складання: ДСТУ 7.1-2006. – [Чинний від 2007-07-01]. – К. : Держспоживстандарт України, 2007. – 47 с.

ДОДАТОК А