**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”**

**Факультет прикладної математики**

**Лабораторна робота №2**

**Тема: «Основні принципи ООП»**

**Виконала:** студентка групи КМ-31

Буряк Софія

**КИЇВ — 2024**

# **Виконання**

## **Моїм завдання було (20 варіант):**

Графічний редактор: Створіть програму, яка дозволяє малювати різні геометричні фігури (кола, квадрати, трикутники). Застосуйте поліморфізм для обробки операцій малювання, переміщення та зміни розміру фігур.

## **1. Алгоритм виконання роботи:**

1. За умовою потрібно мати різні типу фігур, і їхні спільні властивості та поведінку, але реалізовано по іншому, тому я створила абстрактний клас Shape. Я не обрала інтерфейс, тому що він не настільки гнучкий, як абстрактний клас, до того ж фігури мають однакові методи.
2. Далі для кожної фігури — квадрата, трикутника і кола — я створила окремі класи, які успадковують клас Shape.
3. Малювання квадрата: для кожної лінії квадрата я малюю лише перші та останні елементи, а в середині — пробіли. Це дозволяє створити порожню середину.
4. Малювання кола: щоб намалювати круглу форму, мені потрібно перевіряти відстань кожної точки від центру (за допомогою теореми Піфагора). Якщо відстань знаходиться на межі радіуса, я малюю символ \*. Таке рішення дозволяє створити правильне коло.
5. Малювання трикутника: для трикутника я використовую зсув вправо по осі x і додаю пробіли на початку рядка, щоб відформатувати правильний трикутник. Верхівка трикутника містить одну зірочку, нижні ряди мають по дві чи більше зірочок з порожніми місцями всередині.
6. Клас Main керує взаємодією з користувачем, дозволяючи вибрати фігуру (квадрат, коло, трикутник), ввести її координати та розмір, а потім намалювати її в консолі. Він використовує цикл для безперервного запиту введення, створює відповідну фігуру (об'єкт класу Square, Circle, або Triangle) та викликає метод draw() для відображення фігури. Програма завершується, якщо користувач вводить 0.

## **2. Як та коли можна замінити умовний оператор поліморфізмом**

Заміна умовних операторів поліморфізмом стає корисною, коли в коді з’являється багато умовних блоків, які перевіряють тип об’єктів або виконують різні дії на основі значень. Поліморфізм дозволяє уникнути великої кількості if-else чи switch-case блоків, замінюючи їх динамічним вибором методу на основі класу об'єкта.

Приклад на коді:

Замість:

if (shape instanceof Circle) {

// малюємо коло

} else if (shape instanceof Square) {

// малюємо квадрат

}

Робимо так:

Shape shape = new Circle();

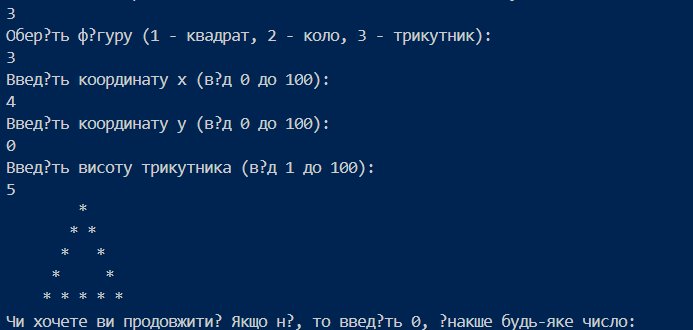
shape.draw(); // викликається метод draw для Circle

## **3. Рефакторинг tРепрррппппп**

|  |  |  |
| --- | --- | --- |
| **№** | **Мої роздуми, запити та коментарі** | **Код та відповіді чату** |
| 1 | Дивлячись на методи draw() у кожному класі фігур, вони займають більше 7 рядків, тому я запитую “Як мені зменшити розмір цих методів?” | **GPT-4o**  @Override  void draw() {  String line = " ".repeat(x) + "\* ".repeat(sideLength);  for (int i = 0; i < y; i++) System.out.println();  System.out.println(line);  for (int i = 1; i < sideLength - 1; i++)  System.out.println(" ".repeat(x) + "\*" + " ".repeat(2 \* sideLength - 3) + "\*");  System.out.println(line);  }  @Override  void draw() {  for (int i = 0; i < y; i++) System.out.println();  for (int i = 0; i <= 2 \* radius; i++) {  System.out.print(" ".repeat(x));  for (int j = 0; j <= 2 \* radius; j++)  System.out.print(Math.abs(Math.sqrt(Math.pow(i - radius, 2) + Math.pow(j - radius, 2)) - radius) < 0.5 ? "\* " : " ");  System.out.println();  }  }  @Override      void draw() {          printEmptyLines(y);          for (int i = 0; i < height; i++) {              System.out.print(" ".repeat(x + height - i - 1) + "\*");              if (i > 0 && i < height - 1) System.out.print(" ".repeat(2 \* i - 1) + "\*");              else if (i == height - 1) System.out.print(" \*".repeat(i));              System.out.println();          }      } |
| 2 | Мені не сподобалась відповідь ШІ, бо виглядає нечитабельно | **DeepSeek Coder**  class Square extends Shape {  private int x, y, sideLength;  public Square(int x, int y, int sideLength) { this.x = x; this.y = y; this.sideLength = sideLength; }  @Override void draw() { String line = " ".repeat(x) + "\* ".repeat(sideLength); for (int i = 0; i < y; i++) System.out.println(); System.out.println(line); for (int i = 1; i < sideLength - 1; i++) System.out.println(" ".repeat(x) + "\*" + " ".repeat(2 \* sideLength - 3) + "\*"); System.out.println(line); }  @Override void move(int x, int y) { this.x = x; this.y = y; System.out.println("Переміщуємо квадрат на нові координати (" + x + ", " + y + ")"); }  @Override void resize(double factor) { this.sideLength \*= factor; System.out.println("Змінили розмір квадрата. Нова довжина сторони: " + sideLength); }  } |
| 3 | Також метод main() займає забагато місця, тому я запитую “Можеш мені порекомедувати як зменшити розмір?” | **Anthropic Claude 3**  Поради щодо рефакторингу методу main()   1. Виділення основних операцій:    * Створіть окремі методи для створення фігури, перевірки продовження програми та обробки винятків. 2. Метод createShape():    * Інкапсулюйте всю логіку вибору та створення фігури в цьому методі. 3. Метод getIntInput():    * Реалізуйте для обробки користувацького введення з перевіркою діапазону. 4. Метод continueProgram():    * Додайте для обробки запиту на продовження роботи програми. 5. Метод handleException():    * Реалізуйте для централізованої обробки всіх можливих винятків. 6. Цикл while (true) у main():    * Використовуйте для постійного виконання програми до явного виходу. 7. Статичне поле scanner:    * Зробіть scanner статичним полем класу для використання в усіх методах.   Застосування цих порад допоможе значно скоротити метод main() та покращити загальну структуру коду. |
| 4 | Клас Main порушує принцип SRP. Для того, щоб дотриматися SRP, можна розділити цей клас на кілька менших класів, кожен з яких буде відповідати за одну конкретну функціональність.  Тому я запитую, як створити клас для взаємодії користувача | **DeepSeek Coder**  Можливий рефакторинг  Клас для взаємодії з користувачем  class UserInteraction {  private Scanner scanner;  public UserInteraction(Scanner scanner) {  this.scanner = scanner;  }  public int getIntInput(String prompt, int min, int max) {  while (true) {  System.out.println(prompt);  int input = scanner.nextInt();  if (input >= min && input <= max) return input;  System.out.println("Невірне значення. Будь ласка, спробуйте ще раз.");  }  }  public boolean continueProgram() {  System.out.println("Чи хочете ви продовжити? Якщо ні, то введіть 0, інакше будь-яке число:");  return scanner.nextInt() != 0;  }  public void handleException(Exception e) {  if (e instanceof InputMismatchException) {  System.out.println("Помилка: введено некоректне значення. Будь ласка, введіть число.");  scanner.nextLine(); // Очистка буфера  } else {  System.out.println("Сталася помилка: " + e.getMessage());  }  }  }  class ShapeFactory {  private UserInteraction userInteraction;  public ShapeFactory(UserInteraction userInteraction) {  this.userInteraction = userInteraction;  }  public Shape createShape() {  int choice = userInteraction.getIntInput("Оберіть фігуру (1 - квадрат, 2 - коло, 3 - трикутник):", 1, 3);  int x = userInteraction.getIntInput("Введіть координату x (від 0 до 100):", 0, 100);  int y = userInteraction.getIntInput("Введіть координату y (від 0 до 100):", 0, 100);  switch (choice) {  case 1: return new Square(x, y, userInteraction.getIntInput("Введіть довжину сторони квадрата (від 1 до 100):", 1, 100));  case 2: return new Circle(x, y, userInteraction.getIntInput("Введіть радіус кола (від 1 до 100):", 1, 100));  case 3: return new Triangle(x, y, userInteraction.getIntInput("Введіть висоту трикутника (від 1 до 100):", 1, 100));  default: throw new IllegalArgumentException("Невірний вибір фігури");  }  }  } |
| 4 | Клас ShapeFactory не закритий для модифікації, тому його потрібно розбити | **DeepSeek Coder**  interface ShapeFactory {  Shape createShape(int x, int y, UserInteraction userInteraction);  }  class SquareFactory implements ShapeFactory {  @Override  public Shape createShape(int x, int y, UserInteraction userInteraction) {  int sideLength = userInteraction.getIntInput("Введіть довжину сторони квадрата (від 1 до 100):", 1, 100);  return new Square(x, y, sideLength);  }  }  class CircleFactory implements ShapeFactory {  @Override  public Shape createShape(int x, int y, UserInteraction userInteraction) {  int radius = userInteraction.getIntInput("Введіть радіус кола (від 1 до 100):", 1, 100);  return new Circle(x, y, radius);  }  }  class TriangleFactory implements ShapeFactory {  @Override  public Shape createShape(int x, int y, UserInteraction userInteraction) {  int height = userInteraction.getIntInput("Введіть висоту трикутника (від 1 до 100):", 1, 100);  return new Triangle(x, y, height);  }  } |
| 5 | Метод DRY не виконується у методі getInput(). Тому я запитую, як можна його змінити. | **DeepSeek Coder**  class UserInteraction {  private Scanner scanner;  public UserInteraction(Scanner scanner) {  this.scanner = scanner;  }  public int getIntInput(String prompt, int min, int max) {  while (true) {  System.out.println(prompt);  int input = scanner.nextInt();  if (isValidInput(input, min, max)) return input;  System.out.println("Невірне значення. Будь ласка, спробуйте ще раз.");  }  }  private boolean isValidInput(int input, int min, int max) {  return input >= min && input <= max;  }  public boolean continueProgram() {  System.out.println("Чи хочете ви продовжити? Якщо ні, то введіть 0, інакше будь-яке число:");  return scanner.nextInt() != 0;  }  public void handleException(Exception e) {  if (e instanceof InputMismatchException) {  System.out.println("Помилка: введено некоректне значення. Будь ласка, введіть число.");  scanner.nextLine(); // Очистка буфера  } else {  System.out.println("Сталася помилка: " + e.getMessage());  }  }  } |
| 6 | Висновок | Мені дуже допомогли ШІ не лише з виконанням завдання, а й з кращим розуміння матеріалу та технік рефакторингу. |

# **Результати виконаної роботи:**

# **Посилання на виконану роботу:** [**Лаб 2**](https://github.com/sofiia-buriak/OOP/tree/%D0%9B%D0%B0%D0%B1%D0%B0-2/%D0%9B%D0%B0%D0%B1%D0%B0%202)



# **Порівняння випробуваних помічників**

|  |  |  |  |
| --- | --- | --- | --- |
| **ШІ** | **Сподобалось** | **Не сподобалось** | **Мій рейтинг, або для чого користуватись** |
| DeepSeek Coder | Чудово підходить для програмування, чіткі відповіді. Якщо порівнювати з іншими нейронками, то статистика каже, що його відповіді краще. Також необмежене кількість запитів. | Якщо довго користуватися починає видавати погані і дивні відповіді. Не завжди відповідає на теоретичні питання. | Мені найбільше подобається користуватися ним з усіх ШІ для програмування. З досвіду використання його, він найчастіше допомагав вирішити проблеми коду. |
| GPT-4o | Зазвичай дає непогані відповіді. Чудово пояснює незрозумілі моменти. Швидший за більшість нейронок. Краще читає файли. | Коли ви пояснюєте йому свої причини, чому ви зробили те, що ви представляєте, таким, яким ви це зробили, він майже не звертає уваги або навіть здається, що розуміє, що ви маєте на увазі. Він радше просто вибачається, а потім переписує все, що ви йому представили, у іншій узагальненій версії, яка така ж погана, як і попередня. Маленький ліміт. | Мені подобається більшість його відповідей, але з програмування він не дотягує. Може дати гарну ідею. |
| Anthropic Claude 3.5 | Він постійно створює код майже без помилок з першої спроби. Відповіді більш схожі на людськи. Гарне оформлення відповідей. | Маленький ліміт. Відповіді іноді взагалі не те, що потрібно | Використовую менше, бо якщо порівнювати з іншими, його відповідь рідко допомагали. Для робити з текстом він найкращий, але не для програмування. |

# **Висновок**

Розробляти за допомогою AI сподобалось, оскільки він допомагав на кожному етапі, надаючи поради, які покращили мій підхід і оптимізували код. Це дозволило краще зрозуміти принципи ООП і застосувати їх на практиці.

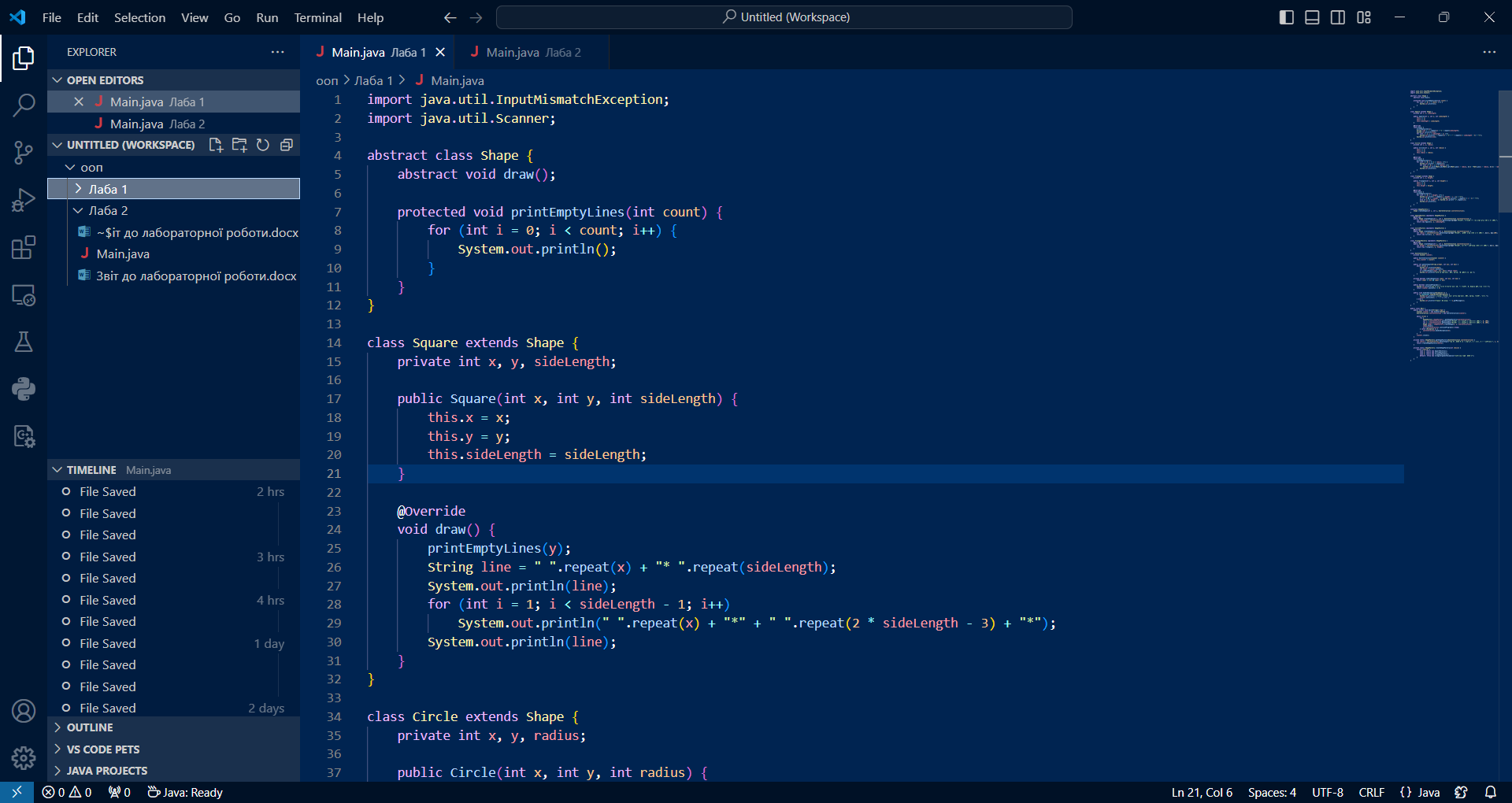
Найкориснішими були поради:

1. Створити окремий клас для взаємодії з користувачем, що полегшило обробку введених даних та дозволило уникнути дублювання коду в основному класі;
2. Рефакторинг методу main, оскільки виділення окремих операцій значно покращило читабельність та спростило підтримку програми.

Водночас, не всі поради були доцільними. Я вирішила не змінювати спосіб виклику фабрики для створення фігур на складніший підхід із рефлексією, адже поточна реалізація є простою та достатньо ефективною для моїх завдань.

# **Лістинг**

Скрін робочого середовища



# **Фінальний код:**

import java.util.InputMismatchException;

import java.util.Scanner;

abstract class Shape {

    abstract void draw();

    protected void printEmptyLines(int count) {

        for (int i = 0; i < count; i++) {

            System.out.println();

        }

    }

}

class Square extends Shape {

    private int x, y, sideLength;

    public Square(int x, int y, int sideLength) {

        this.x = x;

        this.y = y;

        this.sideLength = sideLength;

    }

    @Override

    void draw() {

        printEmptyLines(y);

        String line = " ".repeat(x) + "\* ".repeat(sideLength);

        System.out.println(line);

        for (int i = 1; i < sideLength - 1; i++)

            System.out.println(" ".repeat(x) + "\*" + " ".repeat(2 \* sideLength - 3) + "\*");

        System.out.println(line);

    }

}

class Circle extends Shape {

    private int x, y, radius;

    public Circle(int x, int y, int radius) {

        this.x = x;

        this.y = y;

        this.radius = radius;

    }

    @Override

    void draw() {

        printEmptyLines(y);

        for (int i = 0; i <= 2 \* radius; i++) {

            System.out.print(" ".repeat(x));

            for (int j = 0; j <= 2 \* radius; j++)

                System.out.print(Math.abs(Math.sqrt(Math.pow(i - radius, 2) + Math.pow(j - radius, 2)) - radius) < 0.5 ? "\* " : "  ");

            System.out.println();

        }

    }

}

class Triangle extends Shape {

    private int x, y, height;

    public Triangle(int x, int y, int height) {

        this.x = x;

        this.y = y;

        this.height = height;

    }

    @Override

    void draw() {

        printEmptyLines(y);

        for (int i = 0; i < height; i++) {

            System.out.print(" ".repeat(x + height - i - 1) + "\*");

            if (i > 0 && i < height - 1) System.out.print(" ".repeat(2 \* i - 1) + "\*");

            else if (i == height - 1) System.out.print(" \*".repeat(i));

            System.out.println();

        }

    }

}

interface ShapeFactory {

    Shape createShape(int x, int y, UserInteraction userInteraction);

}

class SquareFactory implements ShapeFactory {

    @Override

    public Shape createShape(int x, int y, UserInteraction userInteraction) {

        int sideLength = userInteraction.getIntInput("Введіть довжину сторони квадрата (від 1 до 100):", 1, 100);

        return new Square(x, y, sideLength);

    }

}

class CircleFactory implements ShapeFactory {

    @Override

    public Shape createShape(int x, int y, UserInteraction userInteraction) {

        int radius = userInteraction.getIntInput("Введіть радіус кола (від 1 до 100):", 1, 100);

        return new Circle(x, y, radius);

    }

}

class TriangleFactory implements ShapeFactory {

    @Override

    public Shape createShape(int x, int y, UserInteraction userInteraction) {

        int height = userInteraction.getIntInput("Введіть висоту трикутника (від 1 до 100):", 1, 100);

        return new Triangle(x, y, height);

    }

}

class UserInteraction {

    private Scanner scanner;

    public UserInteraction(Scanner scanner) {

        this.scanner = scanner;

    }

    public int getIntInput(String prompt, int min, int max) {

        while (true) {

            System.out.println(prompt);

            int input = scanner.nextInt();

            if (isValidInput(input, min, max)) return input;

            System.out.println("Невірне значення. Будь ласка, спробуйте ще раз.");

        }

    }

    private boolean isValidInput(int input, int min, int max) {

        return input >= min && input <= max;

    }

    public boolean continueProgram() {

        System.out.println("Чи хочете ви продовжити? Якщо ні, то введіть 0, інакше будь-яке число:");

        return scanner.nextInt() != 0;

    }

    public void handleException(Exception e) {

        if (e instanceof InputMismatchException) {

            System.out.println("Помилка: введено некоректне значення. Будь ласка, введіть число.");

            scanner.nextLine(); // Очистка буфера

        } else {

            System.out.println("Сталася помилка: " + e.getMessage());

        }

    }

}

public class Main {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        UserInteraction userInteraction = new UserInteraction(scanner);

        while (true) {

            try {

                ShapeFactory shapeFactory = getShapeFactory(userInteraction);

                int x = userInteraction.getIntInput("Введіть координату x (від 0 до 100):", 0, 100);

                int y = userInteraction.getIntInput("Введіть координату y (від 0 до 100):", 0, 100);

                Shape shape = shapeFactory.createShape(x, y, userInteraction);

                shape.draw();

                if (!userInteraction.continueProgram()) break;

            } catch (Exception e) {

                userInteraction.handleException(e);

            }

        }

        scanner.close();

    }

    private static ShapeFactory getShapeFactory(UserInteraction userInteraction) {

        int choice = userInteraction.getIntInput("Оберіть фігуру (1 - квадрат, 2 - коло, 3 - трикутник):", 1, 3);

        return createShapeFactory(choice);

    }

    private static ShapeFactory createShapeFactory(int choice) {

        switch (choice) {

            case 1: return new SquareFactory();

            case 2: return new CircleFactory();

            case 3: return new TriangleFactory();

            default: throw new IllegalArgumentException("Невірний вибір фігури");

        }

    }

}