Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет» Кафедра ИИТ

Лабораторная работа №3
По дисциплине: «СПП»
Вариант 11

Выполнил:
Студент 3 курса
Группы ПО-9
Лебедович В.А.
Проверил:
Крощенко А.А

Брест 2024

Лабораторная работа №3

Цель работы: научиться создавать и использовать классы в программах на языке программирования Java.

Задание 1: Реализовать простой класс: 1) Равнобедренный треугольник, заданный длинами сторон. Предусмотреть возможность определения площади и периметра, а также логический метод, определяющий существует или такой треугольник. Конструктор должен позволить создавать объекты с начальной инициализацией. Реализовать метод equals, выполняющий сравнение объектов данного типа. **Код программы: IsoTriangle.java**

```
package mypack; import
java.util.Objects; public class
IsoTriangle {
    private double sideA, sideB, sideC;

    public IsoTriangle(double sideA, double sideB, double sideC) {
this.sideA = sideA;          this.sideB = sideB;          this.sideC
= sideC;      }       public void setSideA(double sideA) {
this.sideA = sideA;
    }      public void setSideB(double
sideB) {
this.sideB = sideB;
    }      public void setSideC(double
sideC) {
this.sideC = sideC;
    }
    public double getSideA() {
return sideA;
    }
    public double getSideB() {
return sideB;
    }
    public double getSideC() {
return sideC;      }
public boolean isExist(){
        return sideA + sideB > sideC && sideB + sideC > sideA && sideA + sideC > sideB;
}       public boolean
isIsoTriangle(){
        return sideA==sideB || sideA==sideC || sideB==sideC;
}       public double
getPerimeter(){
return sideA + sideB + sideC;
    }       public double
getArea(){
        double semiPerim = (this.sideA + this.sideB + this.sideC)/2;
return Math.sqrt(semiPerim * (semiPerim-sideA) * (semiPerim-sideB) *  (semiPerim-
sideC));
    }

    @Override      public
String toString() {
        return "sideA: " + sideA + "; sideB: " + sideB + "; sideC: " + sideC + ";";
}
```

```
    @Override     public boolean
equals(Object o) {
if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
IsoTriangle that = (IsoTriangle) o;          return Double.compare(sideA,
that.sideA) == 0 && Double.compare(sideB,
that.sideB) == 0 && Double.compare(sideC, that.sideC) == 0;
} }
```

Main.java

```
package mypack;


public class Main {
    public static void main(String[] args) {
        IsoTriangle triangle1 = new IsoTriangle(10,10,10);
        IsoTriangle triangle2 = new IsoTriangle(10,6,6);

        if (triangle1.isExist() && triangle1.isIsoTriangle()){
            System.out.println("Равнобедренный треугольник 1 со сторонами: " +
triangle1.toString());
            System.out.println("Периметр: " + triangle1.getPerimeter() + "; площадь: "
+ triangle1.getArea() + ".\n");
        }         else
{
            System.out.println("Неверные параметры треугольника.");          }

        if (triangle2.isExist() && triangle2.isIsoTriangle()){
            System.out.println("Равнобедренный треугольник 2 со сторонами: " +
triangle2.toString());
            System.out.println("Периметр: " + triangle2.getPerimeter() + "; площадь: "
+ triangle2.getArea() + ".\n");
        }         else
{
            System.out.println("Неверные параметры треугольника.");
}          if (triangle1.isExist() && triangle1.isIsoTriangle() && triangle2.isExist()
&& triangle2.isIsoTriangle()){
            if (triangle1.equals(triangle2)){
                System.out.println("Треугольники равны.");
            }             else
{
                System.out.println("Треугольники не равны.");             }
        }
    }
}
```

Результат работы программы:

```
Равнобедренный треугольник 1 со сторонами: sideA: 10.0; sideB: 10.0; sideC: 10.0;
Периметр: 30.0; площадь: 43.30127018922193.

Равнобедренный треугольник 2 со сторонами: sideA: 10.0; sideB: 6.0; sideC: 6.0;
Периметр: 22.0; площадь: 16.583123951777.

Треугольники не равны.
```

Задание 2: Разработать автоматизированную систему на основе некоторой структуры данных, манипулирующей объектами пользовательского класса. Реализовать требуемые функции обработки данных.

1) Стековый калькулятор
Написать стековый калькулятор, который принимает в качестве аргумента командой строки имя файла, содержащего команды. Если аргумента нет, то использовать стандартный поток ввода для чтения команд. Для вычислений допускается использовать вещественные числа. Реализовать следующий набор команд:

- # – строка с комментарием.
- POP , PUSH – снять/положить число со/на стек(а).
- + , - , * , /, SQRT – арифметические операции. Используют один или два верхних элемента стека, изымают их из стека, помещая результат назад
- PRINT – печать верхнего элемента стека (без удаления).
- DEFINE – задать значение параметра. В дальнейшем везде использовать вместо параметра это значение.

Код программы: Calculator.java

```
import java.util.Stack;
import java.util.Map; import java.util.HashMap;

public class Calculator {
private Stack<Double> stack;
    private Map<String, Double> defineParams;

    public Calculator() {
stack = new Stack<>();
        defineParams = new HashMap<>();
    }
    public void performCommand(String command) {
String[] params = command.split("\\s+");
String operation = params[0];          try {
            switch (operation) {
case "PRINT":
                    if (stack.isEmpty()) {
                        throw new OperationException("Стек пуст!");
}
                    System.out.println(stack.peek());                    break;
                case "#":
break;                  case
"PUSH":                     try
```

```
{
stack.push(Double.parseDouble(params[1]));
                    }
                    catch (Exception e) {
                        throw new OperationException("Не удалось добавить элемент!");
                    }                       break;
case "POP":                         if (stack.isEmpty()) {
throw new OperationException("Стек пуст!");
                    }
stack.pop();                        break;
case "+":
                    if (stack.size() < 2) {
                        throw new OperationException("Недостаточно операндов в
стеке!");
                    }
double a = stack.pop();
double b = stack.pop();
stack.push(a + b);                      break;
case "-":
                    if (stack.size() < 2) {
                        throw new OperationException("Недостаточно операндов в
стеке!");
                    }
a = stack.pop();
b = stack.pop();
stack.push(a - b);                      break;
case "*":
                    if (stack.size() < 2) {
                        throw new OperationException("Недостаточно операндов в
стеке!");
                    }
a = stack.pop();
b = stack.pop();
stack.push(a * b);                      break;
case "/":
                    if (stack.size() < 2) {
                        throw new OperationException("Недостаточно операндов в
стеке!");
                    }
a = stack.pop();
b = stack.pop();

                    if (b == 0) {
stack.push(b);                          stack.push(a);
                        throw new OperationException("Деление на 0!");
                    }
stack.push(a / b);
break;                  case "SQRT":
                    if (stack.isEmpty()) {
throw new OperationException("Стек пуст!");
                    }
                    a = stack.pop();

                    if (a < 0) {
stack.push(a);
                        throw new OperationException("Извлечение корня из
отрицательного числа!");                        }
                    stack.push(Math.sqrt(a));
break;                  case "DEFINE":
                    defineParams.put(params[1], Double.parseDouble(params[2]));
break;                  default:
```

```
                    throw new CommandException("Неизвестная операция:" + operation +
"!");
                }
            } catch (CalculatorException e) {
                System.out.println("Ошибка:" + e);
            }
        } }
```

CalculatorException.java

```
public class CalculatorException extends Exception {
public CalculatorException(String msg){          super(msg);
    } }
```

CommandException.java

```
public class CommandException extends CalculatorException {
public CommandException(String msg){          super(msg);
} }
```

OperationException.java

```
public class OperationException extends CalculatorException {
public OperationException(String msg){          super(msg);
    } }
```

Main.java

```
import java.io.BufferedReader;
import java.io.FileReader; import java.io.IOException;
import java.io.InputStreamReader;

public class Main {
    public static void main(String[] args) {
Calculator calculator = new Calculator();
        try (BufferedReader reader = args.length > 0 ? new BufferedReader(new
FileReader(args[0])) : new BufferedReader(new InputStreamReader(System.in))) {
String line;              while ((line = reader.readLine()) != null) {
calculator.performCommand(line);
            }
        } catch (IOException e) {
System.out.println("Ошибка чтения файла: " + e);          }
} }
```

Результаты работы программы:

Test.txt


```
PS D:\JavaProject\lab3_2\src> java Main Test.txt
7.0
6.0
PS D:\JavaProject\lab3_2\src>
```

Вывод: научился создавать и использовать классы в программах на языке программирования Java.