

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ”
КАФЕДРА ИИТ

ОТЧЁТ
по лабораторной работе №3

Выполнил:

студент 3 курса
группы ПО-9
Мельничук В.М.

Проверил:
Крощенко А.А.

Брест 2024

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

Вариант 1

Задание 1

Равнобедренный треугольник, заданный длинами сторон. Предусмотреть возможность определения площади и периметра, а так же логический метод, определяющий существует или такой треугольник. Конструктор должен позволять создавать объекты с начальной инициализацией. Реализовать метод equals, выполняющий сравнение объектов данного типа.

Выходные данные:

```
Площадь: 7.154544010627092
Периметр: 13.0
Существует ли?:true
Схожи ли два?: false
```

Код программы:

```
import static java.lang.Math.sqrt;
public class task1{
    public static void main(String[] args){
        IsoscelesTriangle a = new IsoscelesTriangle(3, 5);
        IsoscelesTriangle b = new IsoscelesTriangle(15, 21);
        System.out.println(a.square());
        System.out.println(a.perimeter());
        System.out.println(a.isExist());
        System.out.println(a.equals(b));
    }
}

class IsoscelesTriangle {
    private double base;
    private double sides;

    public IsoscelesTriangle(double base, double sides) {
        this.base = base;
        this.sides = sides;
    }

    public double square(){
        double height = sqrt(Math.pow(sides, 2) - Math.pow((base / 2), 2));
        return (base * height) / 2;
    }

    public double perimeter(){
        return 2 * sides + base;
    }
}
```

```

    public boolean isExist(){
        if(sides >= base / 2){
            return true;
        }
        return false;
    }

    public boolean equals(Object obj){
        if(this == obj){
            return true;
        }
        if(obj == null || getClass() != obj.getClass()){
            return false;
        }
        IsoscelesTriangle other = (IsoscelesTriangle) obj;
        return Double.compare(this.sides, other.sides) == 0 &&
            Double.compare(this.base, other.base) == 0;
    }
}

```

Задание 2

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.

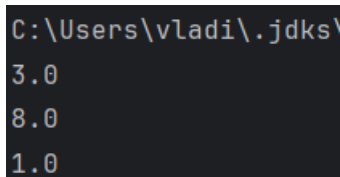
Входные данные:

```

PUSH 5
PUSH 3
PRINT
+
PRINT
PUSH 9
-
PRINT

```

Выходные данные:



```

C:\Users\vladi\.jdkс\
3.0
8.0
1.0

```

Код программы

```

import java.util.*;
import java.io.*;

class InvalidCommandException extends Exception {
    public InvalidCommandException(String message) {
        super(message);
    }
}

```

```

class OperationExecutionException extends Exception {
    public OperationExecutionException(String message) {
        super(message);
    }
}

class ExecutionContext {
    private Stack<Double> stack;
    private Map<String, Double> parameters;

    public ExecutionContext() {
        stack = new Stack<>();
        parameters = new HashMap<>();
    }

    public Stack<Double> getStack() {
        return stack;
    }

    public Map<String, Double> getParameters() {
        return parameters;
    }
}

public class task2 {
    public static void main(String[] args) {
        ExecutionContext context = new ExecutionContext();
        Scanner scanner = null;

        try {
            if (args.length > 0) {
                scanner = new Scanner(new File(args[0]));
            } else {
                scanner = new Scanner(System.in);
            }

            while (scanner.hasNextLine()) {
                String line = scanner.nextLine().trim();

                if (line.isEmpty() || line.startsWith("#")) {
                    continue;
                }

                executeCommand(line, context);
            }
        } catch (FileNotFoundException e) {
            System.out.println("Файл не найден: " + e.getMessage());
        } finally {
            if (scanner != null) {
                scanner.close();
            }
        }
    }

    public static void executeCommand(String command, ExecutionContext context) {
        String[] parts = command.split("\\s+");
        double operand1, operand2;

        try {
            switch (parts[0]) {
                case "POP":
                    if (context.getStack().isEmpty()) {
                        throw new OperationExecutionException("Стек пуст");
                    }
            }
        }
    }
}

```

```

        }
        context.getStack().pop();
        break;
    case "PUSH":
        if (parts.length < 2) {
            throw new InvalidCommandException("Отсутствует аргумент для PUSH");
        }
        context.getStack().push(Double.parseDouble(parts[1]));
        break;
    case "+":
        operand1 = context.getStack().pop();
        if(context.getParameters().containsKey("+")) {
            operand2 = context.getParameters().get("+");
        } else{
            if (context.getStack().isEmpty()) {
                throw new OperationExecutionException("Недостаточно элементов в стеке для
операции +");
            }
            operand2 = context.getStack().pop();
        }
        context.getStack().push(operand1 + operand2);
        break;
    case "-":
        operand1 = context.getStack().pop();
        if(context.getParameters().containsKey("-")) {
            operand2 = context.getParameters().get("-");
        } else{
            if (context.getStack().isEmpty()) {
                throw new OperationExecutionException("Недостаточно элементов в стеке
для операции -");
            }
            operand2 = context.getStack().pop();
        }
        context.getStack().push(operand1 - operand2);
        break;
    case "*":
        operand1 = context.getStack().pop();
        if(context.getParameters().containsKey("*")) {
            operand2 = context.getParameters().get("*");
        } else{
            if (context.getStack().isEmpty()) {
                throw new OperationExecutionException("Недостаточно элементов в стеке
для операции *");
            }
            operand2 = context.getStack().pop();
        }
        context.getStack().push(operand1 * operand2);
        break;
    case "/":
        operand1 = context.getStack().pop();
        if(context.getParameters().containsKey("/")) {
            operand2 = context.getParameters().get("/");
        } else{
            if (context.getStack().isEmpty()) {
                throw new OperationExecutionException("Недостаточно элементов в стеке
для операции /");
            }
            operand2 = context.getStack().pop();
        }
        if (operand2 == 0) {
            throw new OperationExecutionException("Деление на ноль");
        }

```

```

        context.getStack().push(operand1 / operand2);
        break;
    case "SQRT":
        if (context.getStack().isEmpty()) {
            throw new OperationExecutionException("Стек пуст");
        }
        operand1 = context.getStack().pop();
        if (operand1 < 0) {
            throw new OperationExecutionException("Извлечение квадратного корня из отрицательного числа");
        }
        context.getStack().push(Math.sqrt(operand1));
        break;
    case "PRINT":
        if (context.getStack().isEmpty()) {
            throw new OperationExecutionException("Стек пуст");
        }
        System.out.println(context.getStack().peek());
        break;
    case "DEFINE":
        if (parts.length < 3) {
            throw new InvalidCommandException("Некорректное количество аргументов для DEFINE");
        }
        String paramName = parts[1];
        double paramValue = Double.parseDouble(parts[2]);
        context.getParameters().put(paramName, paramValue);
        break;
    default:
        throw new InvalidCommandException("Неизвестная команда: " + parts[0]);
    }
} catch (InvalidCommandException | OperationExecutionException e) {
    System.out.println("Ошибка выполнения команды: " + e.getMessage());
} catch (NumberFormatException e) {
    System.out.println("Ошибка преобразования числа: " + e.getMessage());
}
}
}

```