МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

**КАФЕДРА ИИТ**

ОТЧЁТ

по лабораторной работе №3

Выполнила:

студентка 3 курса

группы ПО-9

Шубич Дарья Константинова

Проверил:

Крощенко А.А.

Брест 2024

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

**Вариант 11**

**Задание 1**

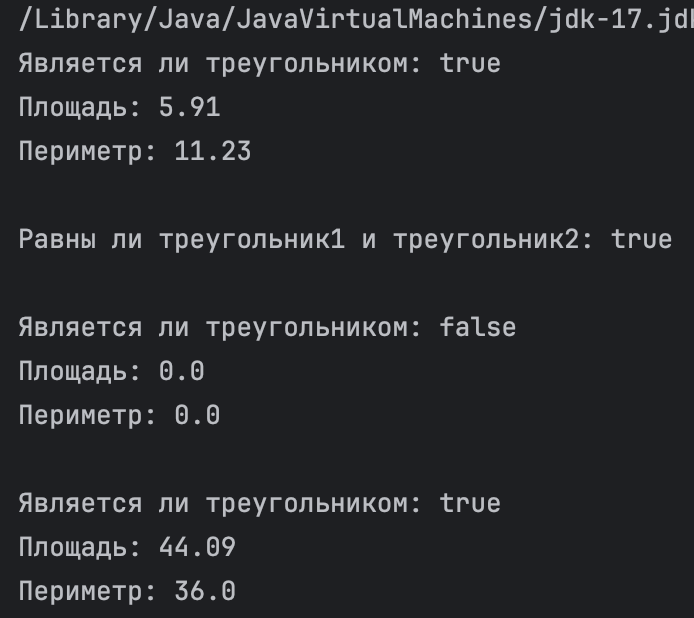
Равнобедренный треугольник, заданный длинами сторон – Предусмотреть возмож-

ность определения площади и периметра, а так же логический метод, определяющий существует или такой треугольник. Конструктор должен позволить создавать объекты с начальной инициализацией. Реализовать метод equals, выполняющий сравнение объектов данного типа.

**Входные данные:**

Zad1 traingle = new Zad1(4, 4 , 3.23);  
System.*out*.println("Является ли треугольником: " + traingle.isIsoscelesTraingle());  
System.*out*.println("Площадь: " + traingle.area());  
System.*out*.println("Периметр: " + traingle.perimeter());  
  
System.*out*.println();  
  
Zad1 traingle1 = new Zad1(4, 4 , 3.23);  
System.*out*.println("Равны ли треугольник1 и треугольник2: " + traingle1.equals(traingle));  
  
System.*out*.println();  
  
traingle1.setA(10);  
traingle1.setB(8);  
traingle1.setC(6);  
System.*out*.println("Является ли треугольником: " + traingle1.isIsoscelesTraingle());  
System.*out*.println("Площадь: " + traingle1.area());  
System.*out*.println("Периметр: " + traingle1.perimeter());  
  
System.*out*.println();  
  
Zad1 traingle2 = new Zad1(6, 15 , 15);  
System.*out*.println("Является ли треугольником: " + traingle2.isIsoscelesTraingle());  
System.*out*.println("Площадь: " + traingle2.area());  
System.*out*.println("Периметр: " + traingle2.perimeter());  
  
System.*out*.println();

**Результат программы**



**Код программы:**

package org.example;  
  
import java.util.Objects;  
  
  
public class Zad1 {  
 private final static double pi = 3.14;  
 private double a;  
 private double b;  
 private double c;  
  
 public Zad1(double a, double b, double c) {  
 this.a = a;  
 this.b = b;  
 this.c = c;  
 }  
  
 public boolean isIsoscelesTraingle() {  
 if (a <= 0 || b <= 0 || c <= 0) {  
 System.out.println("Стороны треугольника должны быть больше нуля и" +  
 "\n для равнобедренного треугольника две стороны должны быть равны!");  
 return false;  
 }  
 return a == b || a == c || b == c;  
 }  
  
 public Zad1() {  
 }  
  
 public double perimeter(){  
 if (!isIsoscelesTraingle()) {  
 return 0.0;  
 }  
  
 return a+b+c;  
 }  
  
 public double area(){  
 if (!isIsoscelesTraingle()) {  
 return 0.0;  
 }  
 if (a == b) {  
 return areaForIsosceles(a, c);  
 } else if (c == a) {  
 return areaForIsosceles(a, b);  
 } else {  
 return areaForIsosceles(b, a);  
 }  
 }  
  
 private double areaForIsosceles(double a, double b) {  
 return Math.round(0.5 \* b \* Math.sqrt(Math.pow(a,2) - (Math.pow(b,2)/4)) \* 100.0) / 100.0 ;  
 }  
  
 @Override  
 public String toString() {  
 return "Zad1{" +  
 "a=" + a +  
 ", b=" + b +  
 ", c=" + c +  
 '}';  
 }  
  
 public double getA() {  
 return a;  
 }  
  
 public double getB() {  
 return b;  
 }  
  
 public double getC() {  
 return c;  
 }  
  
 public void setA(double a) {  
 this.a = a;  
 }  
  
 public void setB(double b) {  
 this.b = b;  
 }  
  
 public void setC(double c) {  
 this.c = c;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof Zad1 zad1)) return false;  
 return Double.compare(a, zad1.a) == 0 && Double.compare(b, zad1.b) == 0 && Double.compare(c, zad1.c) == 0;  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.hash(a, b, c);  
 }  
}

**Задание 2**

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.

Содержимое стека и список определенных именованных параметров передавать команде в ви-

де специального объекта – контекста исполнения. Разработать группу классов исключений,

которые будут выбрасывать команды при исполнении. В случае возникновения исключения –

выводить информацию об ошибке и продолжать исполнение программы (из файла или команд

вводимых с консоли)

**Входные данные:**

PUSH 5

PUSH 3

+

PRINT

CLEAR

#

PUSH 3

SQRT

PRINT

CLEAR

#

НЕПРАВИЛЬНАЯ

#

PUSH 49

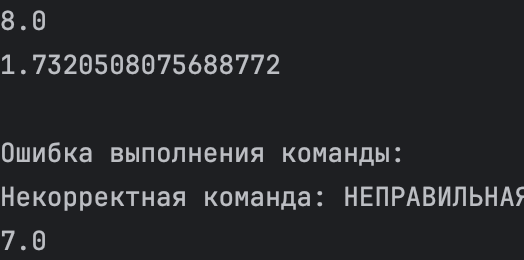
PUSH 7

/

PRINT

CLEAR

**Выходные данные:**



**Код программы:**

package org.example;  
import java.util.Stack;  
import java.util.HashMap;  
  
public class Zad2 {  
 private Stack<Double> stack;  
 private HashMap<String, Double> parameters;  
  
 public Zad2() {  
 stack = new Stack<>();  
 parameters = new HashMap<>();  
 }  
  
 public void executeCommand(String command) {  
 String[] tokens = command.split("\\s+");  
  
 if (tokens.length == 0) {  
 return;  
 }  
  
 String operator = tokens[0];  
  
 try {  
 switch (operator) {  
 case "#":  
 // Комментарий, игнорируем  
 break;  
 case "POP":  
 pop();  
 break;  
 case "PUSH":  
 if (tokens.length > 1) {  
 double value = Double.parseDouble(tokens[1]);  
 push(value);  
 } else {  
 throw new IllegalArgumentException("\nНедостаточно аргументов для команды PUSH");  
 }  
 break;  
 case "+":  
 add();  
 break;  
 case "-":  
 subtract();  
 break;  
 case "\*":  
 multiply();  
 break;  
 case "/":  
 divide();  
 break;  
 case "SQRT":  
 sqrt();  
 break;  
 case "PRINT":  
 print();  
 break;  
 case "CLEAR":  
 stack.clear();  
 break;  
 case "DEFINE":  
 if (tokens.length > 2) {  
 String parameterName = tokens[1];  
 double parameterValue = Double.parseDouble(tokens[2]);  
 define(parameterName, parameterValue);  
 } else {  
 throw new IllegalArgumentException("\nНедостаточно аргументов для команды DEFINE");  
 }  
 break;  
 default:  
 throw new IllegalArgumentException("\nНекорректная команда: " + operator);  
 }  
 } catch (Exception e) {  
 System.out.println("\nОшибка выполнения команды: " + e.getMessage());  
 }  
 }  
  
 private void pop() {  
 if (!stack.isEmpty()) {  
 stack.pop();  
 } else {  
 throw new IllegalStateException("\nСтек пуст");  
 }  
 }  
  
 private void push(double value) {  
 stack.push(value);  
 }  
  
 private void add() {  
 if (stack.size() >= 2) {  
 double operand2 = stack.pop();  
 double operand1 = stack.pop();  
 double result = operand1 + operand2;  
 stack.push(result);  
 } else {  
 throw new IllegalStateException("\nНедостаточно операндов для операции +");  
 }  
 }  
  
 private void subtract() {  
 if (stack.size() >= 2) {  
 double operand2 = stack.pop();  
 double operand1 = stack.pop();  
 double result = operand1 - operand2;  
 stack.push(result);  
 } else {  
 throw new IllegalStateException("\nНедостаточно операндов для операции -");  
 }  
 }  
  
 private void multiply() {  
 if (stack.size() >= 2) {  
 double operand2 = stack.pop();  
 double operand1 = stack.pop();  
 double result = operand1 \* operand2;  
 stack.push(result);  
 } else {  
 throw new IllegalStateException("\nНедостаточно операндов для операции \*");  
 }  
 }  
  
 private void divide() {  
 if (stack.size() >= 2) {  
 double operand2 = stack.pop();  
 if (operand2 != 0) {  
 double operand1 = stack.pop();  
 double result = operand1 / operand2;  
 stack.push(result);  
 } else {  
 throw new IllegalArgumentException("\nДеление на ноль");  
 }  
 } else {  
 throw new IllegalStateException("\nНедостаточно операндов для операции /");  
 }  
 }  
  
 private void sqrt() {  
 if (!stack.isEmpty()) {  
 double operand = stack.pop();  
 if (operand >= 0) {  
 double result = Math.sqrt(operand);  
 stack.push(result);  
 } else {  
 throw new IllegalArgumentException("\nИзвлечение квадратного корня из отрицательного числа");  
 }  
 } else {  
 throw new IllegalStateException("\nСтек пуст");  
 }  
 }  
  
 private void print() {  
 if (!stack.isEmpty()) {  
 double value = stack.peek();  
 System.out.println(value);  
 } else {  
 throw new IllegalStateException("\nСтек пуст");  
 }  
 }  
  
 private void define(String parameterName, double parameterValue) {  
 parameters.put(parameterName, parameterValue);  
 }  
}