Министерство образования Республики Беларусь Учреждение образования

«Брестский государственный технический университет» Кафедра ИИТ

## Лабораторная работа №3

**По дисциплине «СПП»**

**Выполнил:**

Студент группы ПО-8 Янчук А.Г. **Проверил:**

Крощенко А.А.

Брест 2024

# Вариант 11

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

# Задание 1:

Равнобедренный треугольник, заданный длинами сторон – Предусмотреть возможность определения площади и периметра, а так же логический метод, определяющий суще- ствует ли такой треугольник. Конструктор должен позволить создавать объекты с начальной инициализацией. Реализовать метод equals, выполняющий сравнение объектов данного типа.

### Код:

public class IsoscelesTriangle {

private double base; private double side;

public IsoscelesTriangle() { base = side = 0;

}

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

this.side = side;

if(!isValid()) throw new IllegalArgumentException();

}

public boolean isValid() {

return side \* 2 > base && base > 0 && side > 0;

}

public double area() {

if(!isValid()) throw new IllegalArgumentException();

return base \* Math.sqrt(4 \* side \* side - base \* base) / 4;

}

public double perimeter() {

if(!isValid()) throw new IllegalArgumentException(); return base + side \* 2;

}

@Override

public boolean equals(Object obj) { if(this == obj) return true;

if(obj == null || obj instanceof IsoscelesTriangle) return false;

IsoscelesTriangle converted = (IsoscelesTriangle) obj;

return this.side == converted.side && this.base == converted.base;

}

@Override

public String toString() {

String area = isValid() ? String.valueOf(area()) : "N/A"; String perimeter = isValid() ? String.valueOf(area()) : "N/A";

return "Triangle: {\n" + "\tbase:" + base + ",\n" + "\tside:" + side + ",\n" +

"\tarea:" + area + ",\n" + "\tperimeter:" + perimeter + "\n}\n";

}

}

public static void main(String[] args) {

IsoscelesTriangle emptyTriangle = new IsoscelesTriangle(); System.out.println(emptyTriangle.toString());

IsoscelesTriangle validTriangle = new IsoscelesTriangle(6, 5); System.out.println(validTriangle.toString());

try {

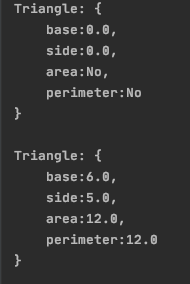
IsoscelesTriangle invalidTriangle = new IsoscelesTriangle(1000, 1);

} catch(IllegalArgumentException e) { System.out.println("Your triangle seems weird...");

}

}

**Результат:**



# 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.

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)

### Код:

package spp.lab.Calculator; public enum Ops {

comment("//", null), pop("pop", 0), push("push", 1),

add("+", 0), sub("-", 0), mult("\*", 0),

div("/", 0), sqrt("sqrt", 0), print("print", 0),

define("define", 2);

private String name; private Integer numParams;

Ops(String opName, Integer numParams) { this.name = opName; this.numParams = numParams;

}

public Integer numParams() { return this.numParams;

}

public String opName() { return this.name;

}

}

package spp.lab.Calculator; class ExecutionContext {

private HashMap<String, Double> parameters; private Stack<Object> stack;

ExecutionContext() {

parameters = new HashMap<>(); stack = new Stack<>();

}

public void setParameter(String name, Double value) { parameters.put(name, value);

}

public Double getParameter(String name) { return parameters.get(name);

}

public boolean hasParameter(String name) { return parameters.containsKey(name);

}

public boolean isEmpty() { return stack.isEmpty();

}

public void push(Object element) throws IllegalStackValueException if(!(element.getClass() == Ops.class) &&

!(element.getClass() == Double.class)

&& !(element.getClass() == String.class)) throw new IllegalStackValueException();

stack.push(element);

}

public Object pop() {

return stack.pop();

}

public Object peek() { return stack.peek();

}

}

package spp.lab.Calculator; public class Calculator {

private static final String ESC\_WORD = "quit"; private BufferedReader reader;

private ExecutionContext context;

public Calculator() { reader = null;

context = new ExecutionContext();

}

public Calculator(BufferedReader r) { setReader(r);

context = new ExecutionContext();

}

public void setReader(BufferedReader reader) { this.reader = reader;

}

public void run() throws IOException, NullPointerException { if(reader == null) throw new NullPointerException(); while(true) {

String line = reader.readLine();

if(line == null || ESC\_WORD.equals(line)) return; try {

CalculationsHandler.parse(line, context); CalculationsHandler.execute(context);

} catch(Exception e) { System.out.println(e.getMessage());

}

}

}

}

package spp.lab.Calculator; class CalculationsHandler {

public static void execute(ExecutionContext context) throws EmptyStackException, UnknownOperationException, UnknownIdentifierException, IllegalStackValueException, IllegalVariableNameException {

if(context.isEmpty()) return; Ops op = (Ops)context.pop();

switch (op) {

case add:

add(context); break;

case div:

div(context); break;

case sub:

sub(context); break;

case sqrt:

sqrt(context); break;

case mult:

mult(context); break;

case push:

push(context); break;

case pop:

pop(context); break;

case print:

print(context); break;

case define:

define(context); break;

case comment: break;

}

}

private static void add(ExecutionContext context) throws IllegalStackValueException, EmptyStackException {

double result = (double)context.pop() + (double)context.pop(); context.push(result);

}

private static void div(ExecutionContext context) throws IllegalStackValueException, EmptyStackException {

double result = (double)context.pop() / (double)context.pop(); context.push(result);

}

private static void sub(ExecutionContext context) throws IllegalStackValueException, EmptyStackException {

double result = (double)context.pop() - (double)context.pop(); context.push(result);

}

private static void sqrt(ExecutionContext context) throws IllegalStackValueException, EmptyStackException {

double result = Math.sqrt((double)context.pop()); context.push(result);

}

private static void mult(ExecutionContext context) throws IllegalStackValueException, EmptyStackException {

double result = (double)context.pop() \* (double)context.pop(); context.push(result);

}

private static void push(ExecutionContext context) throws IllegalStackValueException, UnknownIdentifierException {

String val = (String)context.pop(); if(!context.hasParameter(val)) {

try {

context.push(Double.parseDouble(val));

} catch(Exception e) {

throw new UnknownIdentifierException(val);

}

} else {

context.push(context.getParameter(val));

}

}

private static void pop(ExecutionContext context) { context.pop();

}

private static void print(ExecutionContext context) { System.out.println(context.peek());

}

private static void define(ExecutionContext context) throws UnknownIdentifierException, IllegalVariableNameException {

String name = (String)context.pop(); String val = (String)context.pop();

if(Character.isDigit(name.charAt(0))) {

throw new IllegalVariableNameException(name);

}

double convertedVal; if(!context.hasParameter(val)) {

try { convertedVal = Double.parseDouble(val); }

catch(Exception e) { throw new UnknownIdentifierException(val);}

} else {

convertedVal = context.getParameter(val);

}

context.setParameter(name, convertedVal);

}

public static ExecutionContext parse(String op, ExecutionContext context) throws IllegalNumberOfArgs, UnknownOperationException, IllegalStackValueException {

Ops convertedOp;

String[] tokens = op.split("\\s+");

if(tokens[0].isEmpty() || tokens[0].equals(Ops.comment.opName())) { context.push(Ops.comment);

return context;

}

try { convertedOp = Ops.valueOf(tokens[0].toLowerCase()); } catch(Exception e) { throw new UnknownOperationException(tokens[0]); }

if(tokens.length != convertedOp.numParams() + 1) { throw new IllegalNumberOfArgs(tokens.length - 1,

convertedOp.numParams());

}

for(int i = tokens.length - 1; i > 0; i--) context.push((String)tokens[i]);

context.push(convertedOp); return context;

}

}

public static void main(String[] args) { InputStreamReader ir;

try {

if (args.length == 0)

ir = new InputStreamReader(System.in);

else

ir = new InputStreamReader(new FileInputStream(args[0]));

} catch (IOException e) {

System.out.println("File not found. Entering interactive mode."); ir = new InputStreamReader(System.in);

}

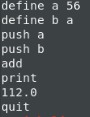
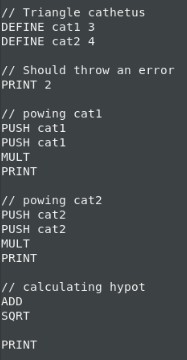
BufferedReader br = new BufferedReader(ir); Calculator calculator = new Calculator(br);

try { calculator.run(); }

catch(Exception e) { System.out.println(e.getMessage()); }

}

**Результат:**



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