МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

 «БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра ИИТ

ЛАБОРАТОРНАЯ РАБОТА №3

Выполнил:

Студент ФЭИС

3-го курса, группы ПО-5

Пищик А.В

Проверил:

Крощенко А. А.

Брест 2021

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

Задание 1

Реализовать простой класс.

Требования к выполнению

• Реализовать пользовательский класс по варианту.

• Создать другой класс с методом main, в котором будут находится примеры использования

пользовательского класса.

Для каждого класса

• Создать поля классов

• Создать методы классов

• Добавьте необходимые get и set методы (по необходимости)

• Укажите соответствующие модификаторы видимости

• Добавьте конструкторы

• Переопределить методы toString() и equals()

Равнобедренный треугольник, заданный длинами сторон – Предусмотреть возможность 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.

Задание 2

Разработать автоматизированную систему на основе некоторой структуры дан- ных,

манипулирующей объектами пользовательского класса. Реализовать требуемые функ-

ции обработки данных

Требования к выполнению

• Задание посвящено написанию классов, решающих определенную задачу автоматизации;

• Данные для программы загружаются из файла (формат произволен). Файл создать и написать

вручную.

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.

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 программы:**

public class laba3 {

public static void main(String[]args) {

try {

IsoscelesTriangle emptyTriangle = new IsoscelesTriangle();

System.out.println(emptyTriangle.toString());

IsoscelesTriangle oneTriangle = new IsoscelesTriangle(5, 4);

System.out.println(oneTriangle.toString());

IsoscelesTriangle twoTriangle = new IsoscelesTriangle(3, 4);

System.out.println(twoTriangle.toString());

System.out.println(twoTriangle.equals(oneTriangle));

IsoscelesTriangle invalidTriangle = new IsoscelesTriangle(10000, 2);

} catch(IllegalArgumentException e) {

System.out.println("Can't create!");

}

}

}

class IsoscelesTriangle {

private int base;

private int side;

public IsoscelesTriangle() {

base = side = 0;

}

public IsoscelesTriangle(int base, int side) {

this.base = base;

this.side = side;

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

}

public boolean isValid() {

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

}

public int area() {

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

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

}

public int perimeter() {

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

return base + side \* 2;

}

@Override

public boolean equals(Object obj) {

if(this == obj) return true;

if(obj == null) 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()) : "No"; String perimeter = isValid() ? String.valueOf(perimeter()) : "No";

return "Triangle: {\n" +

"\tbase:" + base + ",\n" +

"\tside:" + side + ",\n" +

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

}

}

TASK2

**Main.java**

package task2;

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

ExecContext context = new ExecContext();

Commands commands = new Commands();

File file;

Scanner scanner;

if(args.length==0)

scanner = new Scanner(System.in);

else if(args.length==1) {

try {

file = new File(args[0]);

scanner = new Scanner(file, "utf-8");

}

catch(FileNotFoundException e){

System.out.println("Error: file not found");

return;

}

}

else{

System.out.println("Error: unexpected params");

return;

}

while(scanner.hasNextLine())

{

String command = scanner.nextLine();

if(command.indexOf("#")==0)

continue;

else if(command.indexOf("POP")==0)

{

try

{

context = commands.PopCommand(context, command);

}

catch(Exception e)

{

System.out.println("Error");

}

}

else if(command.indexOf("PUSH")==0)

{

try

{

context = commands.PushCommand(context, command);

}

catch(Exception e)

{

System.out.println("Error");

}

}

else if(command.indexOf("+")==0)

{

try

{

context = commands.PlusCommand(context, command);

}

catch(Exception e)

{

System.out.println("Error");

}

}

else if(command.indexOf("-")==0)

{

try

{

context = commands.MinusCommand(context, command);

}

catch(Exception e)

{

System.out.println("Error");

}

}

else if(command.indexOf("\*")==0)

{

try

{

context = commands.MultiplyCommand(context, command);

}

catch(Exception e)

{

System.out.println("Error");

}

}

else if(command.indexOf("/")==0)

{

try

{

context = commands.DivideCommand(context, command);

}

catch(Exception e)

{

System.out.println("Error");

}

}

else if(command.indexOf("SQRT")==0)

{

try

{

context = commands.SqrtCommand(context, command);

}

catch(Exception e)

{

System.out.println("Error");

}

}

else if(command.indexOf("PRINT")==0)

{

try

{

commands.PrintCommand(context, command);

}

catch(Exception e)

{

System.out.println("Error");

}

}

else if(command.indexOf("DEFINE")==0)

{

try

{

commands.DefineCommand(context, command);

}

catch(Exception e)

{

System.out.println("Error");

}

}

else if(command.indexOf("EXIT")==0)

break;

else

{

System.out.println("Error: unknown command");

}

}

}

}

**ExecContext.java**

package task2;

import java.util.HashMap;

import java.util.Stack;

public class ExecContext {

private Stack<Double> stack;

private HashMap<String, Double> params;

public ExecContext() {

stack = new Stack<Double>();

params = new HashMap<String,Double>();

}

public void pushInStack(double value){

stack.push(value);

}

public double popFromStack(){

return stack.pop();

}

public double peekFromStack(){

return stack.peek();

}

public boolean stackIsEmpty(){

return stack.empty();

}

public void setParam(String name, double value){

params.put(name, value);

}

public Double getParam(String name){

return params.get(name);

}

}

**Commands.java**

package task2;

public class Commands {

public ExecContext PopCommand(ExecContext context, String command) throws Exception

{

if(!command.equals("POP"))

throw new Exception();

if(context.stackIsEmpty())

throw new Exception();

context.popFromStack();

return context;

}

public ExecContext PushCommand(ExecContext context, String command) throws Exception

{

String[] args = command.split(" ");

if(args.length!=2)

throw new Exception();

if(!args[0].equals("PUSH"))

throw new Exception();

Double value = null;

try{

value = Double.parseDouble(args[1]);

}

catch(NumberFormatException e)

{

value = context.getParam(args[1]);

}

if(value==null)

throw new Exception();

context.pushInStack(value);

return context;

}

public ExecContext PlusCommand(ExecContext context, String command) throws Exception

{

if(!command.equals("+"))

throw new Exception();

double value1, value2;

if(context.stackIsEmpty())

throw new Exception();

value2 = context.popFromStack();

if(context.stackIsEmpty()) {

context.pushInStack(value2);

throw new Exception();

}

value1 = context.popFromStack();

context.pushInStack(value1+value2);

return context;

}

public ExecContext MinusCommand(ExecContext context, String command) throws Exception

{

if(!command.equals("-"))

throw new Exception();

double value1, value2;

if(context.stackIsEmpty())

throw new Exception();

value2 = context.popFromStack();

if(context.stackIsEmpty()) {

context.pushInStack(value2);

throw new Exception();

}

value1 = context.popFromStack();

context.pushInStack(value1-value2);

return context;

}

public ExecContext MultiplyCommand(ExecContext context, String command) throws Exception

{

if(!command.equals("\*"))

throw new Exception();

double value1, value2;

if(context.stackIsEmpty())

throw new Exception();

value2 = context.popFromStack();

if(context.stackIsEmpty()) {

context.pushInStack(value2);

throw new Exception();

}

value1 = context.popFromStack();

context.pushInStack(value1\*value2);

return context;

}

public ExecContext DivideCommand(ExecContext context, String command) throws Exception

{

if(!command.equals("/"))

throw new Exception();

double value1, value2;

if(context.stackIsEmpty())

throw new Exception();

value2 = context.popFromStack();

if(value2==0) {

context.pushInStack(value2);

throw new Exception();

}

if(context.stackIsEmpty()) {

context.pushInStack(value2);

throw new Exception();

}

value1 = context.popFromStack();

context.pushInStack(value1/value2);

return context;

}

public ExecContext SqrtCommand(ExecContext context, String command) throws Exception

{

if(!command.equals("SQRT"))

throw new Exception();

double value;

if(context.stackIsEmpty())

throw new Exception();

value = context.popFromStack();

if(value<0) {

context.pushInStack(value);

throw new Exception();

}

context.pushInStack(Math.sqrt(value));

return context;

}

public void PrintCommand(ExecContext context, String command) throws Exception

{

if(!command.equals("PRINT"))

throw new Exception();

if(context.stackIsEmpty())

throw new Exception();

System.out.println(context.peekFromStack());

}

public ExecContext DefineCommand(ExecContext context, String command) throws Exception

{

String[] args = command.split(" ");

if(args.length!=3)

throw new Exception();

if(!args[0].equals("DEFINE"))

throw new Exception();

Double value = null;

try{

value = Double.parseDouble(args[2]);

}

catch(NumberFormatException e)

{

throw new Exception();

}

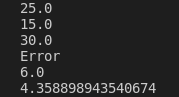
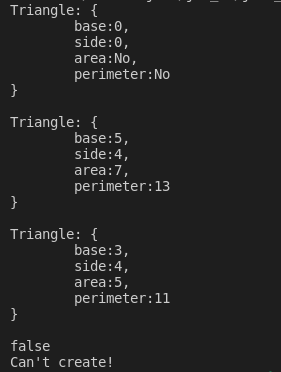
context.setParam(args[1], value);

return context;

}

}

**Вывод работы:**



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