МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**ЛАБОРАТОРНАЯ РАБОТА №1**

**Дисциплина: Программирование для мобильных платформ**

**Тема: «Создание программы на Java в текстовом редакторе.**

**Работа с пакетами и Jar-архивами Java»**

Работу выполнил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Курбатский В.А.

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

Направленность (профиль): Компьютерные технологии и прикладная математика.

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Шиян В. И.

Краснодар

2023

**Цель работы**

Освоить основы работы в командной строке Java. Изучить синтаксис простой программы. Посмотреть возможности языка. Освоить основы работы с пакетами и архивами jar.

**Практическая часть**

**Постановка задачи**

Задание 1:

1. скомпилировать и выполнить примеры 1-4 из командной строки;
2. используя справочную информацию по классу Graphics, поэкспериментировать с графическими примитивами;
3. сформулировать выводы о структуре Java-программы и методах использования классов;

Задание 2:

Взять программу JavaApplication1.java и разделить классы так, чтобы все они находились в разных пакетах.

Объяснить смысл обведенных фрагментов кода (Рисунок 1).

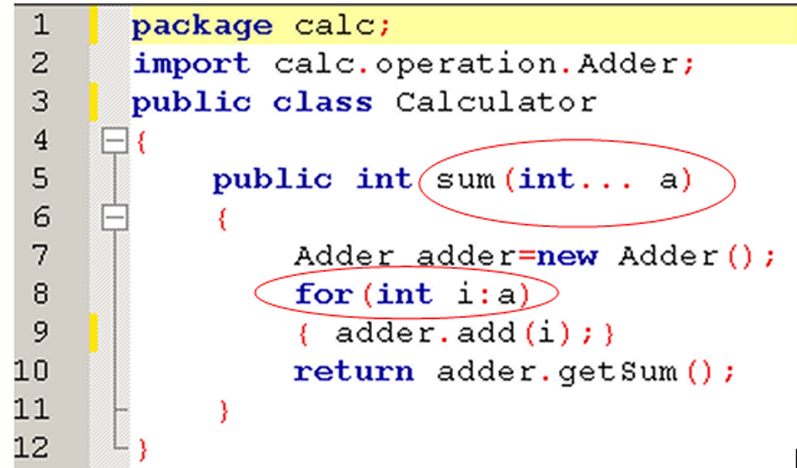
****

Рисунок 1 – Код с обведенными фрагментами

Задание 3:

Доработать пример 1 до калькулятора с основными математическими действиями: сложение, вычитание, умножение, деление, доп. операция (Таблица 1).

|  |  |  |
| --- | --- | --- |
| Вариант | Тип переменной | Доп. операция |
| 7 | Int | Инвертирование |

Таблица 1 – Доп. операция

**Шаг 1**

В текстовом редакторе создадим файл JavaApplication1.java.

**Листинг JavaApplication1.java**

public class JavaApplication1 {

public static void main(String[] args) {

System.out.println("Hello World!");

Calculator calc = new Calculator();

System.out.println("2 + 2 = " + calc.sum(2, 2));

}

public static class Adder {

private int sum;

public Adder() {

sum = 0;

}

public Adder(int a) {

this.sum = a;

}

public void add(int b) {

sum += b;

}

public int getSum() {

return sum;

}

}

public static class Calculator {

public int sum(int... a) {

Adder adder = new Adder();

for (int i: a) {

adder.add(i);

}

return adder.getSum();

}

}

}

**Шаг 2**

Заходим в директорию, где хранится файл JavaApplication1.java и выполняем компиляцию (Рисунок 4).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 2 – Выполнение компиляции в директории, где хранится файл JavaApplication1.java

В результате получаем скомпилированные файлы JavaApplication1.class, JavaApplication1$Adder.class, JavaApplication1$Calculator.class (Рисунок 5).

Изображение выглядит как снимок экрана, текст, линия

Автоматически созданное описание

Рисунок 3 – Скомпилированные файлы JavaApplication1.class, JavaApplication1$Adder.class, JavaApplication1$Calculator.class

**Шаг 3**

Запускаем из командной строки (Рисунок 6).

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 4 – Запуск из командной строки

**Шаг 4**

В текстовом редакторе создадим файл SimpleFrame.java.

**Листинг SimpleFrame.java**

import java.awt.\*;

import java.awt.event.\*;

class SimpleFrame extends Frame {

SimpleFrame(String s) {

super(s);

setSize(400, 150);

setVisible(true);

addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent ev) {

dispose();

System.exit(0);

}

});

}

public static void main(String[] args) {

new SimpleFrame("Моя программа");

}

}

Заходим в директорию, где хранится файл SimpleFrame.java и выполняем компиляцию.

В результате получаем скомпилированные файлы SimpleFrame.class, SimpleFrame$1.class.

Запускаем из командной строки.

Результат запуска изображен на рисунке 7.



Рисунок 7 – Примитивное окно приложений с обработкой события Close

**Шаг 5**

В текстовом редакторе создадим файл HelloWorldFrame.java.

**Листинг HelloWorldFrame.java**

import java.awt.\*;

import java.awt.event.\*;

class HelloWorldFrame extends Frame {

HelloWorldFrame(String s) {

super(s);

}

public void paint(Graphics g) {

g.setFont(new Font("Serif", Font.ITALIC | Font.BOLD, 30));

g.drawString("Hello, XXI century World!", 20, 100);

}

public static void main(String[] args) {

Frame f = new HelloWorldFrame("Здравствуй, мир XXI века!");

f.setSize(400, 150);

f.setVisible(true);

f.addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent ev) {

//dispose();

System.exit(0);

}

});

}

}

Заходим в директорию, где хранится файл HelloWorldFrame.java и выполняем компиляцию.

В результате получаем скомпилированные файлы HelloWorldFrame.class, HelloWorldFrame$1.class.

Запускаем из командной строки.

Результат запуска изображен на рисунке 8.

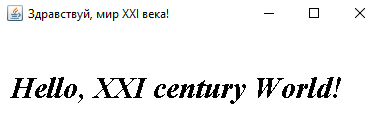


Рисунок 5 – Окно приложений с выводом текста

**Шаг 6**

В текстовом редакторе создадим файл GraphTest01.java.

**Листинг GraphTest01.java**

import java.awt.\*;

import java.awt.event.\*;

class GraphTest01 extends Frame {

GraphTest01(String s) {

super(s);

setBounds(0, 0, 500, 300);

setVisible(true);

}

public void paint(Graphics g) {

Dimension d = getSize();

int myWidth = 250, myHeight = 250;

g.drawLine(0, 0, myWidth, myHeight);

g.drawLine(0, 0, d.width, d.height);

setBackground(Color.blue);

setForeground(Color.red);

}

public static void main(String[] args) {

GraphTest01 f = new GraphTest01("Пример рисования");

f.addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent ev) {

System.exit(0);

}

});

}

}

Заходим в директорию, где хранится файл GraphTest01.java и выполняем компиляцию.

В результате получаем скомпилированные файлы GraphTest01.class, GraphTest01$1.class.

Запускаем из командной строки.

Результат запуска изображен на рисунке 9.

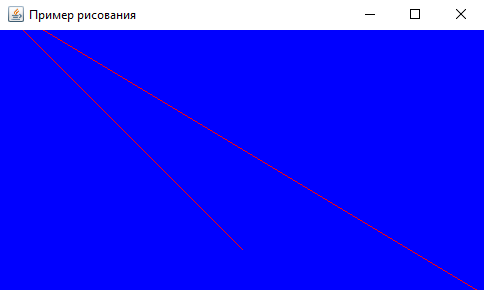


Рисунок 6 – Окно приложений c графическим примитивом линии

**Шаг 7**

Используя справочную информацию по классу Graphics, поэкспериментируем с графическими примитивами (Рисунок 10).

**Листинг Task1.java**

import java.awt.\*;

import java.awt.event.\*;

class Task1 extends Frame{

Task1(String s){

super(s);

setBounds(0, 0, 305, 305);

setVisible(true);

}

public void paint(Graphics g){

Dimension d = getSize();

g.fillRect(0, 0, 100,100);

g.fillRect(200, 0, 100, 100);

g.fillRect(50, 200, 200, 100);

setBackground(Color.WHITE);

setForeground(Color.BLACK);

}

public static void main(String[] args){

Frame f = new Task1("Image");

f.addWindowListener(new WindowAdapter(){

public void windowClosing(WindowEvent ev){

System.exit(0);

}

});

}

}

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок 7 – Личико

**Шаг 8**

Возьмем программу JavaApplication1.java и разделим классы так, чтобы все они находились в разных пакетах.

Создаем структуру папок (Рисунок 8).

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 8 – Созданная структура папок

**Шаг 9**

Внутрь директории calc записываем классы: Calc (который назывался JavaApplication1) и класс Calculator.

**Листинг Calc.java**

package calc;

public class Calc {

public static void main(String[] args) {

System.out.println("Privet");

Calculator calc = new Calculator();

System.out.println("2 + 2 = " + calc.sum(2, 2));

}

}

**Листинг Calculator.java**

package calc;

import calc.operation.Adder;

public class Calculator {

public int sum(int... a) {

Adder adder = new Adder();

for (int i: a) {

adder.add(i);

}

return adder.getSum();

}

}

**Шаг 10**

В директорию operation определяем класс Adder.

**Листинг Adder.java**

package calc.operation;

public class Adder {

private int sum;

public Adder() {

sum = 0;

}

public Adder(int a) {

this.sum = a;

}

public void add(int b) {

sum += b;

}

public int getSum() {

return sum;

}

}

**Шаг 11**

Компилируем (Рисунок 9).

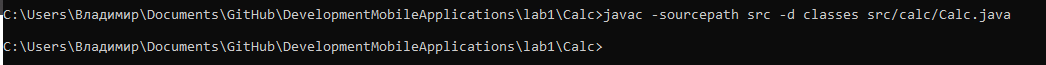


Рисунок 9 – Компиляция

**Шаг 12**

Запускаем на выполнение (Рисунок 10).

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 10 – Запуск на выполнение

**Шаг 13**

Объясним смысл обведенных фрагментов кода, изображенную на рисунке 1.

Конструкция … указывает, что метод sum класса Calculator будет использовать переменное количество аргументов. Эти элементы будут храниться в массиве, на который ссылается переменная a.

Конструкция for (int i: a) { ... } означает, что при прохождении цикла переменной i автоматически присваивается значение, равное значению следующего элемента массива a. Кроме того, при таком способе исключается возможность ошибок выхода за пределы массива. Эту синтаксическую конструкцию можно переписать следующим образом: for (int i = 0; i < a.length; ++i) { ... }.

**Шаг 14**

Доработаем пример 1 до калькулятора с основными математическими действиями: сложение (уже есть), вычитание, умножение, деление, доп. операция согласно варианту (Таблица 1).

**Листинг Calc.java**

package calc;

public class Calc {

public static void main(String[] args){

System.out.println("Privet");

Calculator calc = new Calculator();

System.out.println("2+2= " + calc.sum(2,2));

System.out.println("2-2= " + calc.minus(2,2));

System.out.println("2\*2= " + calc.mult(2,2));

System.out.println("2/2= " + calc.del(2,2));

System.out.println("not 2= " + calc.inv(2));

}

}

**Листинг Calculator.java**

package calc;

import calc.operation.Adder;

import calc.operation.Minus;

import calc.operation.Multiple;

import calc.operation.Delenie;

import calc.operation.Invert;

public class Calculator

{

public int sum(int... a)

{

Adder adder = new Adder();

for(int i:a)

{

adder.add(i);

}

return adder.getSum();

}

public int minus(int a, int... b)

{

Minus minusOb = new Minus(a);

for(int i:b)

{

minusOb.add(i);

}

return minusOb.getMinus();

}

public int mult(int a, int... b)

{

Multiple multOb = new Multiple(a);

for(int i:b)

{

multOb.add(i);

}

return multOb.getMult();

}

public int del(int a, int... b)

{

Delenie delOb = new Delenie(a);

for(int i:b)

{

delOb.add(i);

}

return delOb.getDel();

}

public int inv(int a)

{

Invert invertOb = new Invert(a);

return invertOb.getInvert();

}

}

**Листинг Adder.java**

package calc.operation;

public class Adder

{

private int sum;

public Adder()

{

sum = 0;

}

public Adder(int a)

{

this.sum = a;

}

public void add(int b)

{

sum+=b;

}

public int getSum()

{

return sum;

}

}

**Листинг Delenie.java**

package calc.operation;

public class Delenie

{

int del;

public Delenie(int a)

{

this.del = a;

}

public void add(int b)

{

del/=b;

}

public int getDel() {

return del;

}

}

**Листинг Invert.java**

package calc.operation;

public class Invert

{

int digit;

public Invert(int a)

{

digit = ~a;

}

public int getInvert()

{

return digit;

}

}

**Листинг Multiple.java**

package calc.operation;

public class Multiple

{

int mult;

public Multiple(int a)

{

mult = a;

}

public void add(int b)

{

mult\*=b;

}

public int getMult()

{

return mult;

}

}

**Шаг 15**

Выполним:

1) компиляцию классов с размещением в указанной папке classes; исходные файлы находятся в каталоге Calculator\src\

2) создание jar архива для всего проекта;

3) запуск приложения из jar архива (Рисунок 11).

Изображение выглядит как снимок экрана, текст

Автоматически созданное описание

Рисунок 11 – Компиляция классов, запуск приложения, создание jar архива для всего проекта и запуск приложения из jar архива

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рисунок 12 – Иерархия приложения

**Выводы по проделанной работе**

Освоили основы работы в командной строке Java. Изучили синтаксис простой программы. Посмотрели возможности языка. Освоили основы работы с пакетами и архивами jar.