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: Платформо-независимое программирование**

Работу выполнил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ К. А. Корнилов

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В. И. Шиян

**Тема:** Работа в командной строке – компиляция и запуск на выполнение.

**Цель**: Цель работы

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

**Задание:**

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;
4. Взять программу JavaApplication1.java и разделить классы так, чтобы все они находились в разных пакетах.
5. Объяснить смысл обведенных фрагментов кода (Рисунок 1).

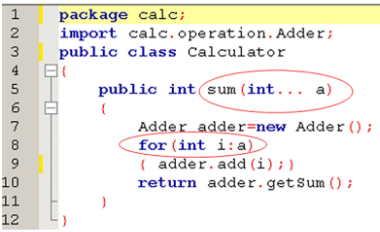


Рисунок 1 – Код с обведенными фрагментами

1. Доработать приложение “Калькулятор”, дополнив его основными математическими действиями согласно варианту. Тип переменной указан для всех реализуемых операций. При этом каждая операция должна быть описана отдельным классом и находиться в отдельном файле.
2. Выполнить компиляцию классов с размещением в указанной папке classes. Исходные файлы находятся в каталоге project\src\; запуск приложения из папки classes; создание архива jar для всего проекта, запуск приложения из jar архива.

**Ход работы:**

1. В текстовом редакторе создадим файл JavaApplication1.java

Код 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();

}

}

}

1. Выполняем компиляцию классов командой на рисунке 2:



Рисунок 2 – Команда компиляции классов

Получаем результат компиляции классов в виде появившихся файлов на рисунке 3.

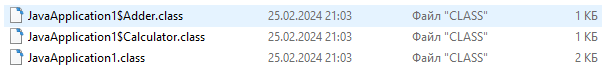


Рисунок 3 – Результат компиляции классов

1. Выполняем запуск программы и получаем ее результат работы.

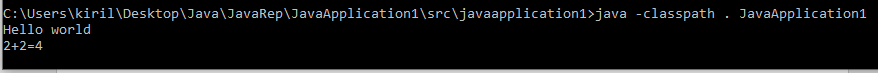


Рисунок 4 – Запуск и результат работы программы

1. Создаем класс SimpleFrame.

Листинг класса:

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.



Рисунок 5 – Результат компиляции классов



Рисунок 6 – Результат выполнения программы

1. В текстовом редакторе создан файл HelloWorldFrame.java

Листинг программы:

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, 21 century wolrd!",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) {

f.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 7.

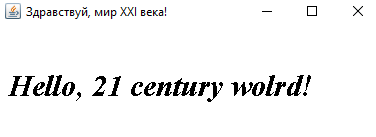


Рисунок 7 – Результат запуска программы

1. В текстовом редакторе создадим файл GraphTest01.java

Листинг программы:

class GraphTest01 extends Frame{

GraphTest01(String s){

super(s);

setBounds(0,0,500,300);

setVisible(true);

addWindowListener(new WindowAdapter(){

public void windowClosing(WindowEvent ev){

System.exit(0);

}

});

}

public void paint(Graphics g){

Dimension d = getSize();

int dx = d.width/20, dy=d.height/20;

int myWidth = 250, myHeight=250;

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

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

g.setColor(Color.red);

g.fillOval(d.width/2, d.height/2, 30, 30);

g.setColor(Color.green);

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

g.setColor(Color.yellow);

g.fillRect(50, 50, 60, 60);

g.fillRoundRect(110,100,30,30,25,25);

g.setColor(Color.black);

g.drawArc(40,40,100,1000,40,180);

setBackground(Color.blue);

}

}

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 8.

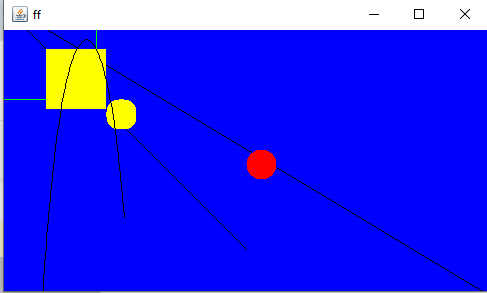


Рисунок 8 – Результат запуска программы

1. Сформулируем выводы о структуре Java-программы и методах использования классов.

Java является объектно-ориентированным языком, поэтому всю программу можно представить как набор взаимодействующих между собой классов и объектов. В первой главе при создании первого приложения программа была определена следующим образом:

public class Program{

public static void main (String args[]) {

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

}

}

То есть основу нашей программы составляет класс Program. При определении класса вначале идет модификатор доступа public, который указывает, что данный класс будет доступен всем, то есть мы сможем его запустить из командной строки. Далее идет ключевое слово class, а затем название класса. После названия класса идет блок кода, в котором расположено содержимое класса. Входной точкой в программу на языке Java является функция main, которая определена в классе Program. Именно с нее начинается выполнение программы. Он обязательно должен присутствовать в программе. При этом его заголовок может быть только таким:public static void main (String args[])

При запуске приложения виртуальная машина Java ищет в главном классе программы метод main с подобным заголовком, и после его обнаружения запускает его. Вначале заголовка метода идет модификатор public, который указывает, что метод будет доступен извне. Слово static указывает, что метод main - статический, а слово void – что он не возвращает никакого значения. Далее в скобках у нас идут параметры метода –

String args[] – это массив args, который хранит значения типа String, то есть строки. При запуске программы через этот массив мы можем передать в программу различные данные.

После заголовка метода идет его блок, который содержит набор выполняемых инструкций.

1. 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("Hello World!");

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();

}

}

1. 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;

}

}

1. Компилируем полученный код

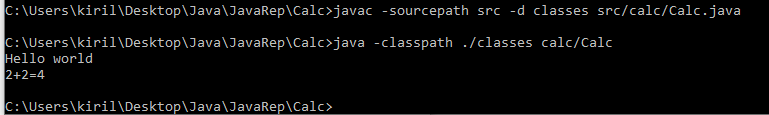


Рисунок 9 – Результат компиляции классов

1. 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(inti = 0; i < a.length; ++i) { ... }.

1. Были добавлены операции согласно варианту.

Листинг Calc.java:

package calc;

import calc.Calculator;

public class Calc {

public static void main(String[] args) {

System.out.println("Hello world");

Calculator calc = new Calculator();

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

System.out.println("2&2&3="+calc.LogMultiply(2,2,3));

System.out.println("2-2-3="+calc.sub(2,2,3));

System.out.println("2\*2\*3="+calc.Multiply(2,2,3));

System.out.println("2/2/3="+calc.Divide(10,2,3));

}

}

Листинг Calculator.java

package calc;

import calc.operation.\*;

public class Calculator {

public int sum(int...a){

Adder adder = new Adder();

for(int i:a){

adder.add(i);

}

return adder.getSum();

}

public int sub(int...a){

Substracter substracter = new Substracter((short)a[0]);

for(int i=1;i<a.length;i++){

substracter.substract((short)a[i]);

}

return substracter.getSub();

}

public short LogMultiply(int ...a){

LogMult multiplyer = new LogMult((short)(a[0]));

for(int i:a){

multiplyer.mult((short)i);

}

return multiplyer.getMult();

}

public short Multiply(int ...a){

Multiplyer multiplyer = new Multiplyer((short)(a[0]));

for(int i=1;i<a.length;i++){

multiplyer.mult((short)a[i]);

}

return multiplyer.getMult();

}

public short Divide(int ...a){

Divider div = new Divider((short)(a[0]));

for(int i=1;i<a.length;i++){

div.divide((short)a[i]);

}

return div.getDivide();

}

}

Листинг LogMult.java:

package calc.operation;

public class LogMult {

private short TotalMult;

public LogMult(){

this.TotalMult = 0;

}

public LogMult(short a){

this.TotalMult = a;

}

public void mult(short a){

this.TotalMult = (short) (this.TotalMult & a);

}

public short getMult(){

return this.TotalMult;

}

}

Листинг Multiplyer:

package calc.operation;

public class Multiplyer {

private short TotalMult;

public Multiplyer(){

this.TotalMult = 0;

}

public Multiplyer(short a){

this.TotalMult = a;

}

public void mult(short a){

this.TotalMult = (short) (this.TotalMult \* a);

}

public short getMult(){

return this.TotalMult;

}

}

Листинг Divider.java:

package calc.operation;

public class Divider {

private short Result;

public Divider(){

this.Result = 0;

}

public Divider(short a){

this.Result = a;

}

public void divide(short a){

this.Result = (short) (this.Result / a);

}

public short getDivide(){

return this.Result;

}

}

Листинг Substracter.java:

public class Substracter {

private short sub;

public Substracter(){

this.sub = 0;

}

public Substracter(short a){

this.sub = a;

}

public void substract(short b){

this.sub-=b;

}

public short getSub(){

return this.sub;

}

}

1. Была проведена компиляция и запуск программы:

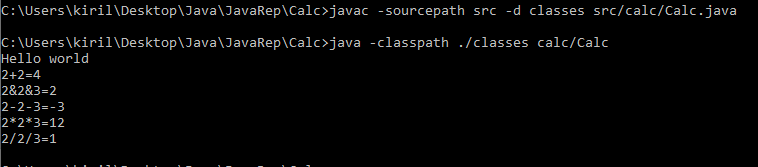


Рисунок 10 – Результат компиляции классов

1. Был создан и запущен jar архив полученной программы.

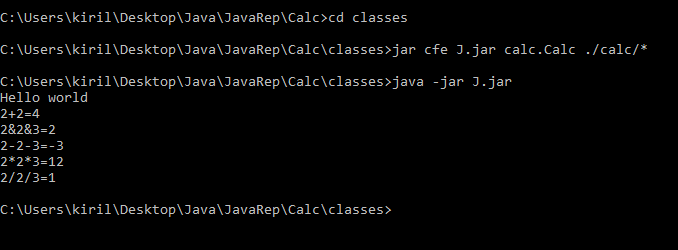


Рисунок 11 – Запуск jar архива полученной программы

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