Университет ИТМО

Кафедра вычислительной техники

**Отчёт по лабораторной работе №2  
по предмету «Программирование интернет-приложений»**

Выполнил:  
Студент гр. P3201  
Петров Д. С.

Принял:  
Гаврилов А. В.

**Номер варианта**

Вариант выданный преподавателем: **20121**

**Цель работы**

Изучить способы записи чисел на языке Java в различных системах счисления. Ознакомиться с понятиями pool данных, интернированием строк. На практике опробовать полиморфизмы языка java.

**Задание**

Скомпилировать и запустить исходный код на языке программирования Java, выданный в соответствии с вариантом. Разобраться том, как реализуются принципы объектно-ориентированного программирования в получившейся программе, и том, почему она выдаёт такой результат. Добавить комментарии в ключевые фрагменты программы, поясняющие её поведение.

**Исходный код получившейся программы**

// Вариант 20121

public **class** Lab2 {

public static void main(String[] args) {

Metang son = new Metang();

Beldum mother = new Beldum();

Beldum sister = new Metang(); //Liskov Substitution Principle

mother.camouflage(); //Вызов метода класса Beldum

son.batonPass(); //Вызов метода класса Metang, НО данный метод определен только в Beldum

son.workUp(); //Вызов метода класса Metang

sister.workUp(); //Вызов метода класса Beldum

mother.mindMold(); //Вызов метода класса Beldum

son.heavyMetal(); //Вызов метода класса Metang

son.snore(mother); //Вызов метода класса Metang с сигнатурой: public void snore(Beldum p)

sister.lightningrod(); //Вызов метода класса Beldum

mother.snore(sister); //Вызов метода класса Beldum с сигнатурой: public void snore(Beldum p)

mother.workUp(); //Вызов метода класса Beldum

son.defenseCurl(); //Вызов метода класса Metang

((Metang)sister).sharpen(); //Вызов метода класса Metang

son.snore(sister); //Вызов метода класса Metang с сигнатурой: public void snore(Beldum p)

son.camouflage(); //Вызов метода класса Metang

sister.camouflage(); //Вызов метода класса Beldum

mother.snore(son); //Вызов метода класса Beldum с сигнатурой: public void snore(Metang p)

}

}

**class** Beldum {

protected String darkWater = "DarkWater";

public byte burrow;

protected String dark = "Dark";

float accuracy = 5.7f;

protected String water = "Water";

protected int weathershape;

static int inflatable;

public Beldum() {

burrow = (byte) 0x88; //16-ти ричная СС

weathershape = 071; //8-ми раичная СС

}

{

//Блок инициализации (выполняется перед вызовом любого конструктора данного класса)

weathershape = 8;

}

static {

//Статический блок инициализации (выполняется при первом запуске класса)

inflatable = 88;

}

public void batonPass() {

//Здесь все понятно

System.out.println(weathershape + inflatable); //Складываем 2 числа

System.out.println(burrow - weathershape); //burrow неявно воспринимается как int. Старшие биты заполняются знаковым разрядом

System.out.println(inflatable - burrow);

}

public void snore(Beldum p) {

System.out.println("Beldum attacks Beldum with Snore");

}

public void camouflage() {

System.out.println("Beldum casts Camouflage");

}

public void mindMold() {

float height = 4.1f;

System.out.println((accuracy + height) == 9.8f); //false - из-за неточности представления float

}

public void snore(Metang p) {

System.out.println("Beldum attacks Metang with Snore");

}

public void lightningrod() {

System.out.println(darkWater == "Dark"+"Water"); //true: и левая, и правая части условия лежат в pool

System.out.println(darkWater.equals("Dark"+"Water")); //true: equals сравнивает по значению

System.out.println(darkWater == dark+water); //false: левая часть условия лежит в pool, правая часть - не лежит

System.out.println(darkWater.equals(dark+water)); //true: equals сравнивает по значению

System.out.println(darkWater.equals(dark+"Water")); //true: equals сравнивает по значению

System.out.println(darkWater == "Dark"+water); //false: левая часть условия лежит в pool, правая часть - не лежит

}

public static void workUp() {

System.out.println("Beldum casts Work Up");

}

}

**class** Metang extends Beldum {

private String fightingWater = "FightingWater";

private int sky = 88;

float weight = 5.5f;

private String fighting = "Fighting";

public void snore(Beldum p) {

System.out.println("Metang attacks Beldum with Snore");

}

public void camouflage() {

System.out.println("Metang casts Camouflage");

}

public void defenseCurl() {

System.out.println(sky + burrow); // 88 + (-120) = - 32

System.out.println(weathershape - sky); // 071- 88 = -31

System.out.println(sky - inflatable); // 88 - 88 = 0

}

public void snore(Metang p) {

System.out.println("Metang attacks Metang with Snore");

}

public void heavyMetal() {

float depth = 6.3f;

System.out.println((depth - weight) == 0.8f); //false - из-за неточности представления float

}

public void sharpen() {

System.out.println(fightingWater == new String("Fighting"+"Water")); //false: левая часть условия лежит в pool, правая часть - не лежит

System.out.println(fightingWater == fighting+water); //false: левая часть условия лежит в pool, правая часть - не лежит

System.out.println(fightingWater == (fighting+water).intern()); //true: intern - кладет правую часть условия в pool

String int\_string = (fighting+water).intern();

System.out.println(fightingWater == int\_string); //true: и левая, и правая части условия лежат в pool

System.out.println(fightingWater == new String("FightingWater")); //false: левая часть условия лежит в pool, правая часть - не лежит

}

public static void workUp() {

System.out.println("Metang casts Work Up");

}

}

**Вывод**

В языке java имеют место полиморфизмы, которые позволяют повысить возможности по организации кода и читабельность исходных текстов, так же, как создание расширяемых программ, которые могут расти не только во время их создания, но и после, когда к ним нужно добавить новую требуемую возможность.

Все строки в java, записанные "в кавычках" на самом деле хранятся в пуле строк, т.е. по умолчанию являются интернированными. Когда вызывается метод  intern(), рантайм-библиотека просматривает пул интернированных объектов в поисках данного или равного ему. Если такой объект находится, возвращается указатель на него, если нет, данный объект интернируется и возвращается указатель на новый объект из пула.