**ИТМО Кафедра Вычислительной техники**

# Отчет по лабораторной работе №2
# Вариант 1718

**Выполнил: студент группы P3217**
**Плюхин Дмитрий**
**Преподаватель: Гаврилов А. В.**

**2016 год**

## 1. Задание к лабораторной работе

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.

## 2. Исходный код

```
public class Lab2 {
    public static void main(String[] args) {
        Gastly sister = new Haunter();
        Haunter daughter = new Haunter();
        Gastly mother = new Gastly();

        // Вызов public метода из Haunter
        daughter.workUp();

        // Вызов public метода из Gastly
        mother.growth();
        sister.sharpen();

        // Вызов статического метода через экземпляр
        // Вызывается метод, определенный в типе данных объекта
        // (т.е. метод из Haunter)
        daughter.batonPass();

        // Выбор метода из двух возможных происходит по типу аргумента sister
        // Поскольку sister была объявлена с типом Gastly, то
        // вызывается первая версия
        mother.knockOff(sister);

        // Вызов public метода из Haunter
        daughter.growth();

        // Такого метода нет в Haunter, но зато такой метод был
        // получен в результате наследования Haunter от Gastly, поскольку
        // в Gastly метод объявлен как public
        daughter.defenseCurl();

        // Вызов public метода из Haunter
        daughter.conversion();

        // Вызов метода focusEnergy(), соответствующий типу Haunter, а не Gastly
        // что достигается использованием приведения типов
        // в данном случае ошибки нет потому что фактически
        // sister указывает на объект
        // типа Haunter и приведением мы восстанавливаем тип ссылки
        ((Haunter)sister).focusEnergy();

        // Без приведения к типу Haunter из sister можем вызывать только
        // те методы, которые были объявлены в Gastly, причем если в
        // Haunter метод переопределяется,
        // то вызывается его новая версия
        sister.growth();

        // Выбор метода из двух возможных происходит по типу аргумента daugher
        // Поскольку daugher была объявлена с типом Haunter,
        // то вызывается вторая версия
        mother.knockOff(daughter);
```

```
        // Вызов public метода из Gastly
        mother.batonPass();

        // Выбор метода из двух возможных происходит по типу аргумента mother
        // Поскольку mother была объявлена с типом Gastly,
        // то вызывается вторая версия
        daughter.knockOff(mother);

        // Выбор метода из двух возможных происходит по типу аргумента sister
        // Поскольку sister была объявлена с типом Gastly,
        // то вызывается вторая версия
        daughter.knockOff(sister);

        // Метод batonPass - статический, поэтому не переопределяется
        // Здесь вызывается его начальная версия, записанная в типе Gastly
        sister.batonPass();

        // Вызов public метода из Gastly
        mother.howl();
    }
}

class Gastly {
    protected String flyingFairy = "FlyingFairy";
    protected String flying = "Flying";
    protected static int troublesome = 95;
    protected String fairy = "Fairy";

    int burrow;
    float speed = 9.4f; //float literal

    public int glow;

    public Gastly() {
        burrow = 95;
        glow = 064; // 64 oct = 52 dec
    }

    // блок инициализации, выполнится в начале конструктора
    {
        glow = 13;
    }

    public void defenseCurl() {
        System.out.println(burrow - glow); // 95 - 52
        System.out.println(glow - troublesome); // 52 - 95
        System.out.println(troublesome + burrow); // 95 + 95
    }

    public void sharpen() {
        // false потому что сравнение разных ссылок на объекты типа String
        System.out.println(flyingFairy == "Flying"+fairy);
        System.out.println(flyingFairy == flying+fairy);

        // true потому что сравнение содержимого объекта типа String
        // и эквивалентного строкового литерала
        System.out.println(flyingFairy == "Flying"+"Fairy");

        // true потому что сравнение содержимого объекта типа String
        // и содержимого эквивалентного строкового литерала
        System.out.println(flyingFairy.equals("Flying"+"Fairy"));

        // true потому что сравнение эквивалентного содержимого объектов типа String
        System.out.println(flyingFairy.equals(flying+fairy));
        System.out.println(flyingFairy.equals(flying+"Fairy"));
    }
```

```
    public void howl() {
        float length = 3.2f; // float literal

        System.out.println((speed + length) == 12.6f);
        // Выводит false потому что операции с числами
        // с плавающей точкой имеют погрешность и не всегда точны
        // в отличие от операций с целыми числами
        // это связано с тем, что не все десятичные дроби имеют точное
        // двоичное представление
    }

    public static void batonPass() {
        System.out.println("Gastly casts Baton Pass");
    }

    public void growth() {
        System.out.println("Gastly casts Growth");
    }

    public void knockOff(Gastly p) {
        System.out.println("Gastly attacks Gastly with Knock Off");
    }

    public void knockOff(Haunter p) {
        System.out.println("Gastly attacks Haunter with Knock Off");
    }
}
class Haunter extends Gastly {
    double defense = 6.7;

    // При создании экземпляра Haunter вызывается конструктор по умолчанию, в начале
    // которого вызывается конструктор Gastly, т.е. конструктор родительского класса

    private String groundFairy = "GroundFairy";
    private byte overland;
    private String ground = "Ground";

    // блок инициализации, выполнится в начале конструктора
    {
        overland = (byte) 0x95; // 95 hex = 149 dec
                                // при преобразовании к byte старший бит
                                // интерпретируется как знаковый разряд, а само число
                                // воспринимается как отрицательное, записанное в доп. коде
                                // то есть происходит переполнение и вместо 149 получаем -107
    }

    public void conversion() {
        // burrow, glow, troublesome получены от Gastly
        System.out.println(overland - burrow); // -107 - 95
        System.out.println(troublesome - overland); // 95 + 107
        System.out.println(overland + glow); // -107 + 52
    }

    public void focusEnergy() {
        // Метод intern() выполняет интернирование строки, то есть, динамически
        // Из хеш-таблицы получается хеш-код строки, содержимое которой полностью соответствует
        // Содержимому того экземпляра String, на котором метод был вызван
        // В данном случае строка (ground+fairy) уже есть в хеш - таблице строк, причем её
        // хеш-код соответствует хеш-коду groundFairy
        // Поэтому в результате получаем true
        System.out.println(groundFairy == (ground+fairy).intern());

        // Сравнение ссылок на разные объекты типа String, получаем false
        System.out.println(groundFairy == new String("Ground"+"Fairy"));
        System.out.println(groundFairy == new String("GroundFairy"));
        System.out.println(groundFairy == ground+fairy);
    }
```

```
    public void workUp() {
        double attack = 8.1;

        System.out.println((attack - defense) == 1.4);
        // Выводит false потому что операции с числами
        // с плавающей точкой имеют погрешность и не всегда точны
        // в отличие от операций с целыми числами
        // это связано с тем, что не все десятичные дроби имеют точное двоичное представление
    }

    public static void batonPass() {
        System.out.println("Haunter casts Baton Pass");
    }

    public void growth() {
        System.out.println("Haunter casts Growth");
    }

    public void knockOff(Haunter p) {
        System.out.println("Haunter attacks Haunter with Knock Off");
    }

    public void knockOff(Gastly p) {
        System.out.println("Haunter attacks Gastly with Knock Off");
    }
}
```

## 3. Результаты выполнения

```
false
Gastly casts Growth
false
false
true
true
true
true
Haunter casts Baton Pass
Gastly attacks Gastly with Knock Off
Haunter casts Growth
43
-43
190
-202
202
-55
true
false
false
false
Haunter casts Growth
Gastly attacks Haunter with Knock Off
Gastly casts Baton Pass
Haunter attacks Gastly with Knock Off
Haunter attacks Gastly with Knock Off
Gastly casts Baton Pass
false
```

## 4. Вывод

Так, в результате лабораторной работы были рассмотрены такие особенности языка Java, как использование механизма наследования, возможные способы сравнения строк, особенности операций с числами с плавающей точкой, использование механизма динамического связывания, использование операции приведения примитивных типов, использование префиксов и суффиксов в литералах, использование механизма полиморфизма, использование нестатических блоков инициализации. Все полученные знания достаточно полезны и с большой долей вероятности пригодятся при дальнейшем изучении языка Java.