

**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ  
КОММУНИКАЦИЙ**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное  
учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Лабораторная работа № 2  
«Создание иерархии классов»

Выполнила: Студентка группы БВТ2306

Максимова Мария

Москва 2024

Гитхаб: <https://github.com/Mashaaaaa7/itip-2>

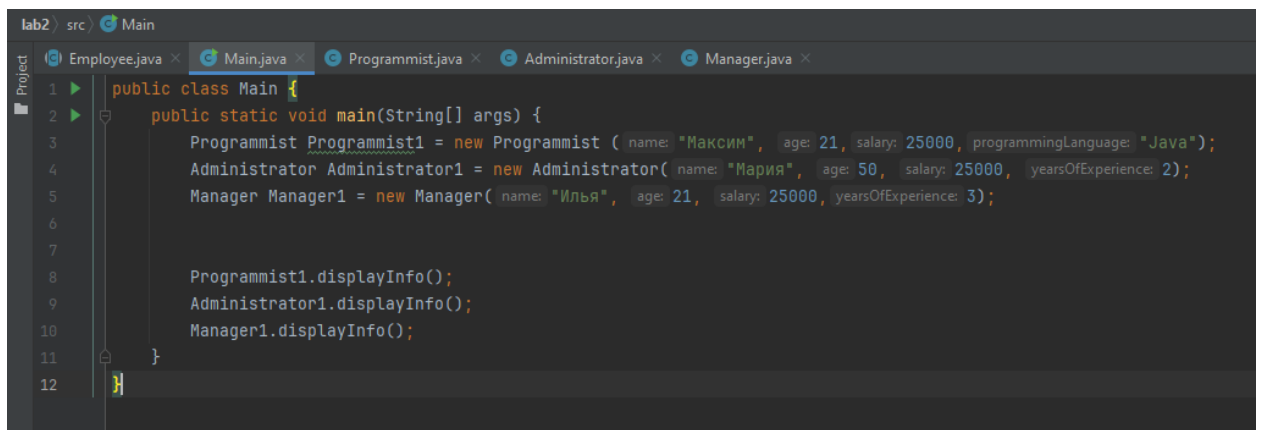
Для лабораторной работы №2 нам понадобится выполнить 1 задание:

**Задание 1 Создайте иерархию классов в соответствии с вариантом.**

Вариант для сделанной лабораторной работы представляю такой:

Базовый класс: Сотрудник Дочерние классы: Администратор, Программист, Менеджер

Класс Main.java, в котором описаны все дочерние классы с информацией об имени, возрасте, заработной плате, опыте работы и языка программирования для программиста):



```
1 public class Main {
2     public static void main(String[] args) {
3         Programmist Programmist1 = new Programmist ( name: "Максим", age: 21, salary: 25000, programmingLanguage: "Java");
4         Administrator Administrator1 = new Administrator( name: "Мария", age: 50, salary: 25000, yearsOfExperience: 2);
5         Manager Manager1 = new Manager( name: "Илья", age: 21, salary: 25000, yearsOfExperience: 3);
6
7
8         Programmist1.displayInfo();
9         Administrator1.displayInfo();
10        Manager1.displayInfo();
11    }
12 }
```

Рисунок 1

Абстрактный класс Сотрудник. Он базовый и выводит имя и возраст

```

1  public abstract class Employee {
2
3      private String name;
4      private int age;
5
6      public Employee (String name, int age) {
7          this.name = name;
8          this.age = age;
9      }
10
11     public Employee (String name, String age) { this( name, "Мария", age: 20); }
12
13     public String getName() { return name; }
14
15     public void setName(String name) { this.name=name; // Устанавливаем новое значение для имени }
16
17     public int getAge() {
18         // no usages 3 overrides
19         return age;
20     }
21
22     public void setAge(int age) { this.age = age; }
23
24     public abstract void displayInfo(); // Абстрактный метод, который должен быть реализован в подклассах
25
26     public String toString() { return "Имя: " + name + ", Возраст: " + age; }
27 }

```

Рисунок 2

Класс Программиста: включает в себя реализованный публичный класс Сотрудника. Выводит имя, возраст, зарплату и язык программирования.

```

1  public class Programmist extends Employee {
2
3      private String name; // Имя
4      private int age; //возраст
5      private double salary; // зарплата
6      private String programmingLanguage; //язык программирования
7
8      public Programmist(String name, int age, double salary, String programmingLanguage) {
9          super(name, age); //аргументы name, age
10         this.salary = salary;
11         this.programmingLanguage = programmingLanguage;
12     }
13
14     public Programmist() {
15         // no usages
16         this( name: "Максим", age: 20, salary: 25000, programmingLanguage: null);
17     }
18
19     // Геттеры и сеттеры
20     public String getName() {
21         // no usages
22         return name;
23     }
24
25     public void setName(String name) {
26         // no usages
27         this.name = name;
28     }
29 }

```

Рисунок 3

```

public int getAge() { no usages
    return age;
}

public void setAge(int age) { no usages
    this.age = age;
}

public double getSalary() { no usages
    return salary;
}

public void setSalary(double salary) { no usages
    this.salary = salary;
}

public String getProgrammingLanguage() { no usages
    return programmingLanguage;
}

public void setProgrammingLanguage(String programmingLanguage) { no usag
    this.programmingLanguage = programmingLanguage;
}

```

Рисунок 4

```

@Override
public String toString() {
    return "Программист: " + "Максим" + ", возраст:" + 21 + ", заработанная плата: " + salary + ", язык программирования: " + programmingLanguage;
}

public void displayInfo() { 3 usages
    System.out.println("Программист:" + "Максим" + ", возраст:" + 21 + ", Зарплата: " + salary + ", язык программирования: " + programmingLanguage);
}

```

Рисунок 5

Класс администратора (имя, возраст, зарплата, опыт работы):

```

public class Administrator extends Employee {
    private String name; // имя 2 usages
    private int age; // возраст 2 usages
    private double salary; // зарплата 3 usages
    private int yearsOfExperience; // опыт работы 3 usages

    public Administrator(String name, int age, double salary, int yearsOfExperience) {
        super(name, age);
        this.salary = salary;
        this.yearsOfExperience = yearsOfExperience;
    }

    public Administrator() {
        this("Мария", age: 35, salary: 0, yearsOfExperience: 0);
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }
}

```

Рисунок 6

```

public void setAge(int age) {
    this.age = age;
}

public double getSalary() {
    return salary;
}

public void setSalary(double salary) {
    this.salary = salary;
}

public int getYearsOfExperience() {
    return yearsOfExperience;
}

public void setYearsOfExperience(int yearsOfExperience) {
    this.yearsOfExperience = yearsOfExperience;
}

```

Рисунок 7

```

public String toString() {
    return "Администратор: " + "Мария" + ", возраст: " + 20 + ", Зарплата: " + getSalary() + ", Опыт работы: " + getYearsOfExperience() + " лет";
}

public void displayInfo() {
    System.out.println("Администратор: " + "Мария" + ", возраст: " + 20 + ", Зарплата: " + getSalary() + ", Опыт работы: " + getYearsOfExperience() + " лет");
}

```

Рисунок 8

Класс Manager(имя, возраст, зарплата, опыт работы):

```
Employee.java Main.java Programmer.java Administrator.java Manager.java
public class Manager extends Employee { 2 usages

    private String name; // имя 2 usages
    private int age; // возраст 2 usages
    private double salary; // зарплата 3 usages
    private int yearsOfExperience; // опыт работы (int вместо String) 3 usages

    public Manager(String name, int age, double salary, int yearsOfExperience) { 2 usages
        super(name, age);
        this.salary = salary;
        this.yearsOfExperience = yearsOfExperience;
    }

    public Manager() { no usages
        this( name: "известен", age: 20, salary: 25000, yearsOfExperience: 0); // Передайте значения в конструктор с параметрами
    }
```

Рисунок 9

```
public String getName() { no usages
    return name;
}

public void setName(String name) { no usages
    this.name = name;
}

public int getAge() { no usages
    return age;
}

public void setAge(int age) { no usages
    this.age = age;
}
```

Рисунок 10

```
public double getSalary() { 2 usages
    return salary;
}

public void setSalary(double salary) { no usages
    this.salary = salary;
}

public int getYearsOfExperience() { 2 usages
    return yearsOfExperience;
}

public void setYearsOfExperience(int yearsOfExperience) { no usages
    this.yearsOfExperience = yearsOfExperience;
}

@Override
public String toString() {
    return "Менеджер: " + "Илья" + ", возраст: " + 21 + ", Зарплата: " + getSalary() + ", Опыт работы: " + getYearsOfExperience() + " лет";
}

public void displayInfo() { 3 usages
    System.out.println("Менеджер: " + "Илья" + ", возраст: " + 21 + ", Зарплата: " + getSalary() + ", Опыт работы: " + getYearsOfExperience() + " лет");
}
```

Рисунок 11

