Практика 8

1. Написать четыре класса: Employee (сотрудник), Manager (управляющий), Chief (директор) и Secretary (секретарь). Унаследовать управляющего, директора и секретаря от сотрудника.

```
public class Solution {
   public static void main(String[] args) {
   }

   public class Manager {
   }

   public class Chief {
   }

   public class Employee {
   }

   public class Secretary {
   }
}
```

2. Изменить два класса **Adam** (**Адам**) и **Eve** (**Ева**). Унаследовать **Еву** от **Адама**.

```
public class Solution {
   public static void main(String[] args) {
   }

   //Адам
   public class Adam {
   }

   //Ева
   public class Eve {
   }
}
```

3. Исправить девять классов: Worker (сотрудник), Clerk (клерк), IT (ИТ-специалист), Programmer (программист), ProjectManager (менеджер проекта), CTO (технический директор), HR (рекрутер), OfficeManager (офис-менеджер), Cleaner (уборщик). Унаследовать программиста, менеджера проекта и технического директора от ИТ-специалиста. Унаследовать рекрутера, уборщика и офис-менеджера от клерка. Унаследовать клерка и ИТ-специалиста от сотрудника.

```
public class Solution {
    public static void main(String[] args) {
    }

    public class Worker {
    }

    public class Clerk {
    }

    public class IT {
    }

    public class Programmer {
    }

    public class ProjectManager {
    }

    public class CTO {
    }

    public class OfficeManager {
    }

    public class HR {
    }

    public class Cleaner {
    }
}
```

4. Скрыть внутренние переменные **класса Cat**, к которым есть доступ с помощью методов.

```
public class Solution {
   public static void main(String[] args) {
   }

   public class Cat {
      public String name;
      public int age;
      public int weight;

      public Cat(String name, int age, int weight) {
         this.name = name;
         this.age = age;
         this.weight = weight;
      }

      public String getName() {
         return name;
      }

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

      public int getAge() {
         return age;
      }

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

5. Скрыть все внутренние переменные **класса Cat**, а также методы, позволяющие менять внутреннее состояние объектов **класса Cat**.

```
public class Solution {
   public static void main(String[] args) {
   }

   public class Cat {
      public String name;
      public int age;
      public int weight;

      public Cat(String name, int age, int weight) {
         this.name = name;
         this.age = age;
         this.weight = weight;
      }

      public String getName() {
         return name;
      }

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

      public int getAge() {
         return age;
      }

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

6. Скрыть все внутренние переменные класса **Cat** и **Dog**. Также скрыть все методы, кроме тех, с помощью которых эти **классы** взаимодействуют **друг с другом**.

```
public class Solution {
   public static void main(String[] args) {
      Cat cat = new Cat("Vaska", 5);
      Dog dog = new Dog("Sharik", 4);
      cat.isDogNear(dog);
      dog.isCatNear(cat);
   }
}

class Cat {
   public String name;
   public int speed;

   public Cat(String name, int speed) {
      this.name = name;
      this.speed = speed;
   }

   public String getName() {
      return name;
   }

   public int getSpeed() {
      return speed;
   }

   public boolean isDogNear(Dog dog) {
      return this.speed > dog.getSpeed();
   }
}

class Dog {
   public String name;
   public int speed;

   public Dog(String name, int speed) {
      this.name = name;
      this.speed = speed;
   }

   public String getName() {
      return name;
   }

   public int getSpeed() {
      return speed;
   }

   public boolean isCatNear(Cat cat) {
      return this.speed > cat.getSpeed();
   }
}
```

7. Посмотрите внимательно на **методы** и добавьте недостающие **поля**.

```
public class Solution {
   public static void main(String[] args) {
   }

   public class Cat {

      public Cat(String name, int age, int weight) {
      }

      public String getName() {
         return null;
      }

      public int getAge() {
         return 0;
      }

      public void setWeight(int weight) {
      }

      public void setSpeed(int speed) {
      }
   }
}
```

8. Изменить два класса **AppleIPhone** и **SamsungGalaxyS2**. Унаследовать **SamsungGalaxyS2** от **AppleIPhone**.

```
public class Solution {
   public static void main(String[] args) {
   }

   public class AppleIPhone {
   }

   public class SamsungGalaxyS2 {
   }
}
```

9. Изменить четыре класса: Fish (Рыба), Animal (Животное), Ape (Обезьяна), Human (Человек). Унаследовать **животное** от **рыбы**, **обезьяну** от **животного** и **человека** от **обезьяны**.

```
public class Solution {
   public static void main(String[] args) {
   }

   public class Fish {
   }

   public class Animal {
   }

   public class Ape {
   }

   public class Human {
   }
}
```

10. Изменить три класса: **Judaism** (**Иудаизм**), **Christianity** (**Христианство**), **Islam** (**Мусульманство**). Унаследовать **христианство** от **иудаизма** и **мусульманство** от **христианства**.

```
public class Solution {
    public static void main(String[] args) {
    }

    public class Judaism {
    }

    public class Christianity {
    }

    public class Islam {
    }
}
```

11. Изменить четыре класса: **Schoolboy** (**школьник**), **Student** (**студент**), **Worker** (**Сотрудник**), **Slave** (**Раб**). Унаследовать **студента** от **школьника**, **сотрудника** от **студента**, **раба** от **сотрудника**.

```
public class Solution {
    public static void main(String[] args) {
    }

    public class Schoolboy {
    }

    public class Student {
    }

    public class Worker {
    }

    public class Slave {
    }
}
```

12. Расставьте правильно "**цепочку наследования**" в классах: **Pet** (**домашнее животное**), **Cat** (**кот**), **Dog** (**собака**).

```
public class Solution {
   public static void main(String[] args) {
   }

   public class Pet {
   }

   public class Cat {
   }

   public class Dog {
   }
}
```

13. Расставьте правильно "**цепочку наследования**" в классах: Carnivore (**плотоядное животное**), Cow (**корова**), Dog (**собака**), Pig (**свинья**), Animal (**животное**).

```
public class Solution {
   public static void main(String[] args) {
   }

   public class Carnivore {
   }

   public class Cow {
   }

   public class Dog {
   }

   public class Pig {
   }

   public class Animal {
   }
}
```

14. Расставьте правильно "**цепочку наследования**" в классах: **Pet** (**домашнее животное**), **Cat** (**кот**), **Dog** (**собака**), **Car** (**машина**).

```
public class Solution {
   public static void main(String[] args) {
   }

   public class Pet {
   }

   public class Cat {
   }

   public class Car {
   }

   public class Dog {
   }
}
```

15. Расставьте правильно "**цепочку наследования**" в классах: **House** (**дом**), **Cat** (**кот**), **Dog** (**собака**), **Car** (**машина**).

```
public class Solution {

   public static void main(String[] args) {
   }

   public class House {
   }

   public class Cat {
   }

   public class Car {
   }

   public class Dog {
   }
}
```

16. Расставьте правильно "**цепочку наследования**" в классах: **House** (**дом**), **Cat** (**кот**), **Dog** (**собака**), **Car** (**машина**), **Animal** (**животное**), **Asset** (**имущество**).

```
public class Solution {
   public static void main(String[] args) {
   }

   public class House {
   }

   public class Cat {
   }

   public class Car {
   }

   public class Dog {
   }

   public class Animal {
   }

   public class Asset {
   }
}
```

17. Исправьте наследование в классах: (**классы Cat**, **Dog**, **Pet**, **House**, **Airplane**).

```
public class Solution {
   public static void main(String[] args) {
   }

   public class Pet extends House {
   }

   public class Cat extends Airplane {
   }

   public class Dog extends Cat {
   }

   public class House extends Dog {
   }

   public class Airplane {
   }
}
```

18. Добавьте общий базовый класс **ChessFigure** к **классам-фигур**: (фигуры из шахмат).

```
public class Solution {
    public static void main(String[] args) {
    }

    public class King {
    }

    public class Queen {
    }

    public class Rook {
    }

    public class Knight {
    }

    public class Bishop {
    }

    public class Pawn {
    }
}
```

19. Написать метод, который возвращает **минимальное** и **максимальное числа** в **массиве**.

```
public class Solution {
    public static void main(String[] args) {
        int[] data = new int[]{1, 2, 3, 5, -2, -8, 0, 77, 5, 5};
        Pair<Integer, Integer> result = getMinimumAndMaximum(data);
        System.out.println("The minimum is " + result.x);
        System.out.println("The maximum is " + result.y);
    }

    public static Pair<Integer, Integer> getMinimumAndMaximum(int[] inputArray) {
        if (inputArray == null || inputArray.length == 0) {
            return new Pair<Integer, Integer>(null, null);
        }
        // напишите тут ваш код

        return new Pair<Integer, Integer>(0, 0);
    }

    public static class Pair<X, Y> {
        public X x;
        public Y y;

        public Pair(X x, Y y) {
            this.x = x;
            this.y = y;
        }
    }
}
```