Практика 9

1. Переопределить метод `getChild` в классах **Cat**(**кот**) и **Dog**(**собака**), чтобы **кот** порождал **кота**, а **собака - собаку**.

```
public class Solution {
   public static void main(String[] args) {
      Pet pet1 = new Cat();
      Pet cat = pet1.getChild();
      Pet pet2 = new Dog();
      Pet dog = pet2.getChild();
   }

   public static class Pet {
      public Pet getChild() {
         return new Pet();
      }
   }

   public static class Cat extends Pet {

   }

   public static class Dog extends Pet {

   }
}
```

2. Написать метод, который определяет, объект какого класса ему передали, и выводит на экран одну из надписей: **Кошка**, **Собака**, **Птица**, **Лампа**.

```
public class Solution {
   public static void main(String[] args) {
      printObjectType(new Cat());
      printObjectType(new Bird());
      printObjectType(new Lamp());
      printObjectType(new Cat());
      printObjectType(new Dog());
   }

   public static void printObjectType(Object o) {
      //Напишите тут ваше решение
   }

   public static class Cat  {
   }

   public static class Dog {
   }

   public static class Bird {
   }

   public static class Lamp {
   }
}
```

3. Написать два метода: `print(int)` и `print(String)`.

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

    }
    //Напишите тут ваши методы
}
```

4. Написать два метода: `print(int)` и `print(Integer)`.
   Написать такой код в методе `main`, чтобы вызвались они оба.

5. Написать пять методов `print` с разными *параметрами*.

6. Написать public static методы: `int min(int, int)`, `long min(long, long)`, `double min(double, double)`. Каждый метод должен возвращать минимальное из двух переданных в него чисел.

7. Унаследуйте класс **Cow** от **Animal**. Реализуйте все недостающие методы в классе Cow.

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

    public static abstract class Animal {
        public abstract String getName();
    }

    public static class Cow {
    }
}
```

8. Унаследуйте классы **Cat** и **Dog** от **Pet**. Реализуйте недостающие методы. Классы **Cat** и **Dog** не должны быть абстрактными.

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

    }

    public static abstract class Pet {
        public abstract String getName();

        public abstract Pet getChild();
    }

    public static class Cat {

    }

    public static class Dog {

    }

}
```

9. Напишите свой `public` интерфейс `CanFly(летать)`. Добавьте в него **два метода**.

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

    }
    //add an interface here - добавь интерфейс тут
}
```

10. Напишите public интерфейсы `CanFly(летать)`, `CanRun(бежать/ездить)`, `CanSwim(плавать)`. Добавить в каждый интерфейс по **одному методу**.

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

    }
    //add interfaces here - добавь интерфейсы тут
}
```

11. Есть public интерфейсы `CanFly(летать)`, `CanMove(передвигаться)`, `CanEat(есть)`. Добавьте эти интерфейсы классам **Dog**(**собака**), **Car**(**автомобиль**), **Duck**(**утка**), **Airplane**(**самолет**).

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

    public interface CanFly {
        public void fly();
    }

    public interface CanMove {
        public void move();
    }

    public interface CanEat {
        public void eat();
    }

    public class Dog {
    }

    public class Duck {
    }

    public class Car {
    }

    public class Airplane {
    }
}
```

12. Есть public интерфейсы CanFly(летать), CanRun(бежать/ездить), CanSwim(плавать). Добавьте эти интерфейсы классам **Human**(**человек**), **Duck**(**утка**), **Penguin**(**пингвин**), **Airplane**(**самолет**).

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

    }

    public interface CanFly {
        public void fly();
    }

    public interface CanRun {
        public void run();
    }

    public interface CanSwim {
        public void swim();
    }

    public class Human {
    }

    public class Duck {
    }

    public class Penguin {
    }

    public class Airplane {
    }
}
```

13. Напишите public класс Human(человек) и public интерфейсы CanRun(бежать/ездить), CanSwim(плавать). Добавьте в каждый интерфейс по **одному методу**. Добавьте эти интерфейсы классу **Human**, но не реализуйте методы. Объявите класс **Human** абстрактным.

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

    }
    //add public interfaces and a public class here - добавьте public интерфейсы и public класс тут
}
```

14. Переопределите метод `getName` в классе **Cat** так, чтобы программа выдавала на экран надпись «***Я - кот***».

```
public class Solution {
   public static void main(String[] args) {
      Pet pet = new Cat();
      System.out.println(pet.getName());
   }

   public static class Pet {
      public String getName() {
         return "Я - пушистик";
      }
   }

   public static class Cat extends Pet {
   }
}
```

15. Переопределите метод `setName` в классе **Cat** так, чтобы программа выдавала на экран надпись «Я - кот»

```
public class Solution {
   public static void main(String[] args) {
      Pet pet = new Cat();
      pet.setName("Я - пушистик");
      System.out.println(pet.getName());
   }

   public static class Pet {
      protected String name;

      public Pet() {
      }

      public String getName() {
         return name;
      }

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

   }

   public static class Cat extends Pet {
   }
}
```

16. Добавьте один метод в класс **Cat** так, чтобы программа ничего не выводила на экран.

```
public class Solution {
    public static void main(String[] args) {
        Pet pet = new Cat();
        pet.setName("Я - пушистик");
        System.out.println(pet.getName());
    }

    public static class Pet {
        protected String name;

        public Pet() {
        }

        public final String getName() {
            return name;
        }

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

    public static class Cat extends Pet {
    }
}
```

17. Напиши метод, который определяет, какой объект передали в него. Программа должна выводить на экран одну из надписей: "Кот", "Тигр", "Лев", "Бык", "Животное".

```
public class Solution {
    public static void main(String[] args) {
        System.out.println(getObjectType(new Cat()));
        System.out.println(getObjectType(new Tiger()));
        System.out.println(getObjectType(new Lion()));
        System.out.println(getObjectType(new Bull()));
        System.out.println(getObjectType(new Pig()));
    }

    public static String getObjectType(Object o) {
        //напишите тут ваш код
        return "Животное";
    }

    public static class Cat {
    }

    public static class Tiger {
    }

    public static class Lion {
    }

    public static class Bull {
    }

    public static class Pig {
    }
}
```

18. Напишите метод, который определяет, какой объект передали в него. Программа должна выводить на экран одну из надписей: "Кот", "Тигр", "Лев", "Бык", "Корова", "Животное". Замечание: постарайся определять тип животного как можно более точно.

```
public class Solution {
   public static void main(String[] args) {
      System.out.println(getObjectType(new Cat()));
      System.out.println(getObjectType(new Tiger()));
      System.out.println(getObjectType(new Lion()));
      System.out.println(getObjectType(new Bull()));
      System.out.println(getObjectType(new Cow()));
      System.out.println(getObjectType(new Animal()));
   }

   public static String getObjectType(Object o) {
      //напишите тут ваш код
      return «Животное";
   }

   public static class Cat extends Animal   //<--Классы наследуются!!
   {
   }

   public static class Tiger extends Cat {
   }

   public static class Lion extends Cat {
   }

   public static class Bull extends Animal {
   }

   public static class Cow extends Animal {
   }

   public static class Animal {
   }
}
```

19. 1. Внутри класса **Solution** создайте интерфейс public interface CanFly(**летать**) с методом void fly().
2. Внутри класса **Solution** создайте интерфейс public interface CanClimb(**лазить по деревьям**) с методом void climb().
3. Внутри класса **Solution** создайте интерфейс public interface CanRun(**бегать**) с методом void run().
4. Подумайте логически, какие именно интерфейсы нужно добавить для каждого класса.
5. Добавьте интерфейсы классам **Cat**(**кот**), **Dog**(**собака**), **Tiger**(**тигр**), **Duck**(**Утка**).

```
public class Solution {

    public static void main(String[] args) {

    }

    public class Cat {
    }

    public class Dog {
    }

    public class Tiger extends Cat {
    }

    public class Duck {
    }
}
```

20. Есть интерфейсы `CanFly(летать)`, `CanSwim(плавать)`, `CanRun(бегать)`. Добавьте эти интерфейсы классам **Duck**(**утка**), **Penguin**(**пингвин**), **Toad**(**жаба**)

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

    public interface CanFly {
        public void fly();
    }

    public interface CanRun {
        public void run();
    }

    public interface CanSwim {
        public void swim();
    }

    public class Duck {
    }

    public class Penguin {
    }

    public class Toad {
    }
}
```

21. Добавьте как можно больше интерфейсов к классу **Human**, но чтобы он не стал абстрактным классом. Добавлять методы в класс **Human** запрещается.

```
public class Solution {
    public static void main(String[] args) {
        Human human = new Human();
        System.out.println(human);
    }

    public static interface Worker {
        public void workLazy();
    }

    public static interface Businessman {
        public void workHard();
    }

    public static interface Secretary {
        public void workLazy();
    }

    public static interface Miner {
        public void workVeryHard();
    }

    public static class Human {
        public void workHard() {
        }
        public void workLazy() {
        }
    }
}
```

22. Добавьте такой класс-родитель к классу **CTO**(**технический директор**), чтобы класс перестал быть ***абстрактным***. Добавлять/реализовывать методы в классе **CTO** запрещается.

```
public class Solution {

    public static void main(String[] args) {
        CTO cto = new CTO();
        System.out.println(cto);
    }

    public static interface Businessman {
        public void workHard();
    }

    public static class CTO implements Businessman {

    }
}
```

23. Добавьте еще один метод, чтобы программа выводила на экран число **10**. Подсказка: используйте перегрузку методов.

```
public class Solution {
    public static void main(String[] args) {
        Integer i = 5;
        int x = transformValue(i);

        System.out.println(x);
    }

    public static int transformValue(int i) {
        return i * i;
    }
}
```

24. Необходимо расставить правильно ключевые слова ***abstract***, чтобы программа компилировалась. Добавьте там где надо и удалите там, где они не нужны.

```
public class Solution {

    public static void main(String[] args) {
        Horse horse = new Pegasus();
        horse.run();
    }

    public static interface CanFly {
        public abstract void fly();
    }

    public static abstract class Horse {
        public void run() {
        }
    }

    public static class Pegasus extends Horse implements CanFly {
        public abstract void fly() {
        }
    }

    public static class SwimmingPegasus extends Pegasus {
        public void swim();
    }
}
```

25. Сделать класс **Pegasus**(**пегас**) на основе класса **Horse**(**лошадь**) и интерфейса `CanFly(летать)`.

```
public class Solution {
    public static void main(String[] args) {
        Pegasus horse = new Pegasus();
    }

    public static interface CanFly {
        public void fly();
    }

    public static class Horse {
        public void run() {
        }
    }

    public static class Pegasus {
    }
}
```

26. Написать метод, который возвращает **минимальное** число в **массиве** и его позицию (**индекс**).

```
public class Solution {
    public static void main(String[] args) throws Exception {
        int[] data = new int[]{1, 2, 3, 5, -2, -8, 0, 77, 5, 5};

        Pair<Integer, Integer> result = getMinimumAndIndex(data);

        System.out.println("The minimum is " + result.x);
        System.out.println("The index of the minimum element is " + result.y);
    }

    public static Pair<Integer, Integer> getMinimumAndIndex(int[] array) {
        if (array == null || array.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;
        }
    }
}
```