4. Ключевое слово static

Нередко возникает потребность создать такой член класса, который бы не зависел от конкретных объектов. Такими членами могут быть переменные, методы или вложенные классы.

Для того чтобы они имели независимость от объектов, необходимо при их создании указать ключевое слово `static` .

Статические переменные

В языке Java, чтобы переменные были статическими, необходимо объявлять их с ключевым словом `static` .

> Переменная **static** — такая переменная, которая принадлежит не какому-то конкретному объекту, а всему классу.

Давайте разберем, в чём отличия между статическими переменными и обычными, на примере:

```
public class SomeStaticClass {
   int number1;
   static int number2;

   public static void main(String[] args) {
      SomeStaticClass someStaticClass = new SomeStaticClass();
      System.out.println(someStaticClass.number1);
      System.out.println(number2);
   }
}
```

Для того чтобы обратиться к переменной `number1` , которая является нестатической, нам нужен объект класса `SomeStaticClass` . А статическая переменная `number2` этого не требует, так как принадлежит классу, в котором мы работаем.

Однако если мы хотим обратиться к статической переменной из другого класса, нам придется это сделать через имя класса, которому она принадлежит, — `SomeStaticClass.number2` — или через любой объект того же класса.

> Обращаться к члену другого класса через объект **не рекомендуется** — можно запутаться в том, какие поля являются статическими, а какие — нет.

Обратимся к переменной через имя ее класса:

```
public class SomeStaticClassDemo {
   public static void main(String[] args) {
      SomeStaticClass someStaticClass1 = new SomeStaticClass();
      SomeStaticClass someStaticClass2 = new SomeStaticClass();

      System.out.println(SomeStaticClass.number2);
      // Обращение к статической переменной через имя класса
      System.out.println(someStaticClass1.number2);
      // Обращение к статической переменной через конкретный объект

      someStaticClass1.number2 = 3;
      someStaticClass2.number2 = 4;

      System.out.println(someStaticClass1.number2);
      System.out.println(someStaticClass2.number2);
    }
}
```

Получим следующий результат:

```
0
0
4
4
```

Приведем стандартный пример использования статической переменной, которая подсчитывает, сколько объектов класса было создано. Определяем и инициализируем в классе статическую переменную `counter`, которая будет являться счетчиком и увеличиваться на единицу при создании каждого нового объекта. Делать это мы будем в конструкторе:

```
public class SomeClass {
   static int counter = 0;

   public SomeClass() {
      counter++;
   }
}

public class SomeClassTest {
   public static void main(String[] args) {
      SomeClass object1 = new SomeClass();
      SomeClass object2 = new SomeClass();
      System.out.println("Созданных объектов:" + SomeClass.counter);
   }
}
```

В итоге получаем такой результат:

```
Созданных объектов: 2
```

Статические методы

Статические методы, так же как и статические переменные, создаются с использованием ключевого слова `static`. Для вызова таких методов не нужен конкретный объект класса, в котором создан сам метод.

Такие методы имеют ряд ограничений:

- Они имеют доступ **только** к статическим переменным.
- Из такого метода можно вызывать **только** статические методы.
- Эти методы **не могут** использовать ссылки на this или super.

Приведем пример использования статических методов:

```
public class SomeStaticClass2 {
   static int number1 = 3;
   int number2;

   public void someMethod() {
      System.out.println("Нестатический метод");
   }

   static void someStaticMethod(int localVar) {
      System.out.println("localVar = " + localVar);
      System.out.println("staticVar = " + number1);
      // Нельзя обратиться к не static-переменной из static-метода
      // System.out.println("nonStaticVar = " + number2);
   }

   public static void main(String[] args) {
      someStaticMethod(42);
      //Нельзя обратиться к не static-методу без указания объекта
      //someMethod();
      SomeStaticClass2 someStaticObj = new SomeStaticClass2();
      someStaticObj.someMethod();
      someStaticObj.someStaticMethod(67);
   }
}

public class SomeStaticClass2Test {
   public static void main(String[] args) {
      SomeStaticClass2.someStaticMethod(42);
   }
}
```

Статический блок

Еще в Java есть такая сущность, как статический блок.

> **Статический блок** — это часть кода, которая выполняется только один раз — в момент загрузки класса.

Внутри этого блока производятся те операции, которые необходимо произвести единожды при загрузке программы.

Для инициализации такого блока необходимо написать ключевое слово `static` , а далее в фигурных скобках обозначить все необходимые вычисления.

```
import java.util.Scanner;

public class SomeStaticClass3 {
   static String string;

   static {
      System.out.println("Инициализация статического блока");
      Scanner scanner = new Scanner(System.in);
      string = scanner.nextLine();
   }

   public static void main(String[] args) {
      System.out.println("String = " + string);
   }
}
```

Java static import

Чтобы воспользоваться статическими членами классов, необходимо указать ссылку на класс, которому они принадлежат.

Например, для того чтобы воспользоваться статическими методами класса `Math` или его константами, мы должны прописать класс `Math` :

```
public class SomeStaticClass4 {
   public static void main(String[] args) {
      double value = Math.cos(Math.PI * 4);
      System.out.println(value);
   }
}
```

Мы также можем импортировать статические члены класса и использовать их, не прописывая имя класса, которому они принадлежат. Для импорта таких членов мы используем конструкцию `import static` , а далее прописываем полное имя класса и название константы или метода.

```java
package oop;

import static java.lang.Math.PI;
import static java.lang.Math.cos;

public class SomeStaticClass5 {
   public static void main(String[] args) {
      double value = cos(PI * 4);
      System.out.println(value);
   }
}
```

Статический импорт Java-языка располагается после указания пакета перед объявлением класса.