CODEGYM

Java Developer in 12 months

Mentor-supported training program from CodeGym

MODULE 1. JAVA SYNTAX

Lesson 13 Classes and the static keyword

CODEGYM

Lesson plan

- Static variables
- Static methods
- Static classes
- Order of initialization

CODEGYM

Static variables

When a class is loaded into memory a **static object** is created immediately. This object stores static class variables (static class fields).

The static object exists even if no ordinary (non-static) objects of the class have been created.

To create a static variable in a class, you need to write the **static** keyword before its name. The general format for declaring a static variable is:

```
static Type name = value;
```

Inside of a class, you can refer to its static variables simply by using their names. But to access them from another class, you need to write the name of the class before the name of the static variable.

```
ClassName.variable
```

CODEGYM

Difference between static and non-static variables

Ordinary variables of a class are bound to objects of the class (instances of the class), while static variables are bound to the class's static object.

If there are multiple instances of a class, each of them has its own copy of the non-static (ordinary) variables.

Static variables of a class are always stored in its static object and only a single instance of them exists.

CODEGYM

Static methods

Static methods differ from ordinary methods in that they are bound to a class, not to an object.

An important property of a static method is that it can only access static variables/methods.

To call an ordinary (non-static) method on a class, you must first create an object of the class and then call the method on the object.

You cannot call an ordinary method on the class rather than an object.

CODEGYM

Static vs ordinary (non-static) methods

The differences between the two types of methods are expressed in the following table:

Ability/property	Ordinary method	Static method
Bound to an instance of the class	Yes	No
Can call ordinary methods of the class	Yes	No
Can call static methods of the class	Yes	Yes
Can access ordinary variables of the class	Yes	No
Can access static variables of the class	Yes	Yes
Can be called on an object	Yes	Yes
Can be called on the class	No	Yes

CODEGYM

Inner classes

In Java, you are allowed to declare classes within classes.
And even classes within classes that are within classes within classes.

```
class OuterClass {          variables of the class
                    methods of the class

  class InnerClass {        variables of the class
                    methods of the class
  }
}
```

Nested classes can be **static** or **non-static**.

Static nested classes are simply called static nested classes. Non-static nested classes are called **inner** classes (inner classes).

CODEGYM

Static classes

Static nested classes can be used outside of their outer class. If such a class has the public access modifier, then it can be used anywhere in the program. Accessing a static nested class from the class other than the one in which it is declared:

```
OuterClass.InnerClass
```

Creating an object of a static nested class:

```
OuterClass.NestedClass name = new OuterClass.NestedClass();
```

Calling a static method of a nested class:

```
OuterClass.NestedClass.staticMethod();
```

Accessing the public static variables of a nested class:

```
OuterParent.NestedClass.nameOfStaticVariable
```

CODEGYM

Features of static classes

Static nested classes have the least reason to be called static. They behave just like regular classes. There are no restrictions on accessing them from non-static methods.

If you're working with a static nested class inside its outer class, you won't notice any difference from the most ordinary (not nested and not static) class.

If you take some static nested class and move it out of its outer class, the only thing that will change is that the new class will no longer be able to access the private static variables and methods of its former outer class.

CODEGYM

Order of initialization

1. Static blocks and fields of the parent class in the order in which they appear in the class.
2. Static blocks and fields of the child class in the order in which they appear in the class.
3. Initialization blocks and non-static fields of the parent class in the order in which they appear in the class.
4. Parent class constructor.
5. Initialization blocks and non-static fields of the child class in the order in which they appear in the class.
6. Child class constructor.

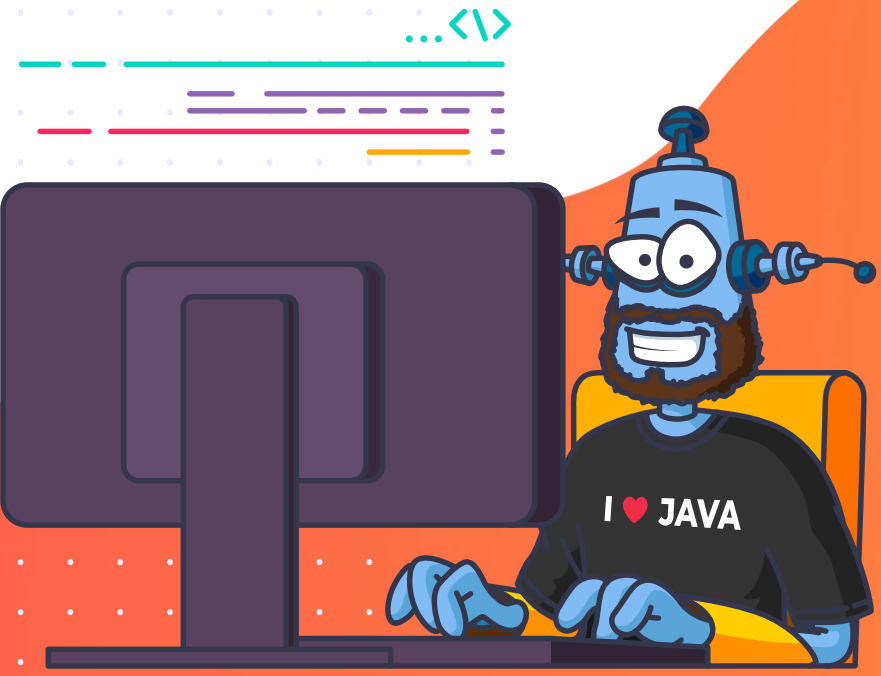
The static blocks and variables of the parent class are processed before the static blocks and variables of the child class. The same is true of non-static blocks and variables and constructors: first the parent class, then the child

CODEGYM

Homework

MODULE 1. JAVA SYNTAX

Complete Level 14



CODEGYM

Answers to questions

MODULE 1. JAVA SYNTAX