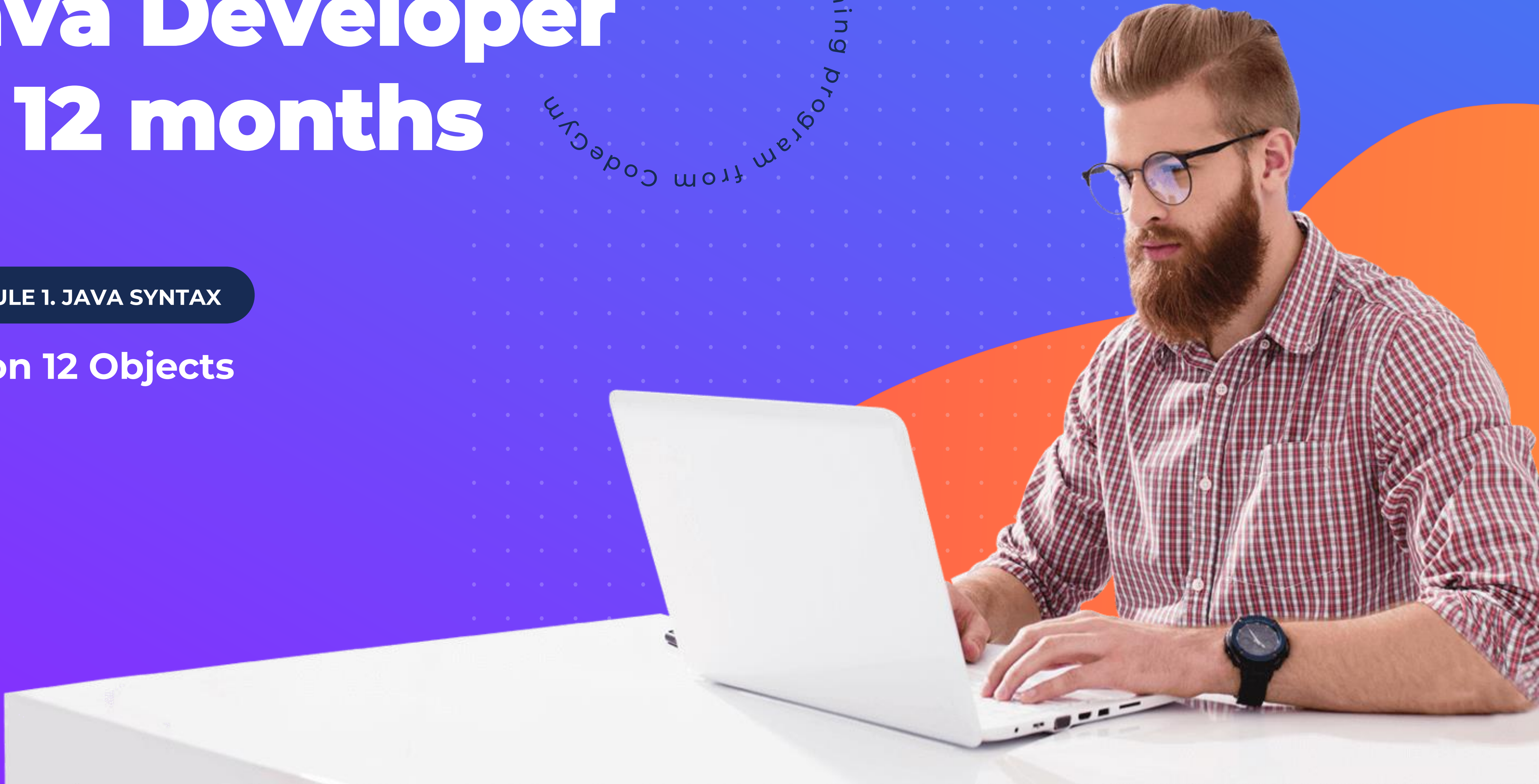
CODEGYM

Java Developer in 12 months

Mentor-supported training program from CodeGym

MODULE 1. JAVA SYNTAX

Lesson 12 Objects



CODEGYM

Lesson plan

- How Java programs are organized
- Objects and constructors
- Initializing variables
- Properties: getters and setters

CODEGYM

How Java programs are organized

Every Java program consists of classes and objects.

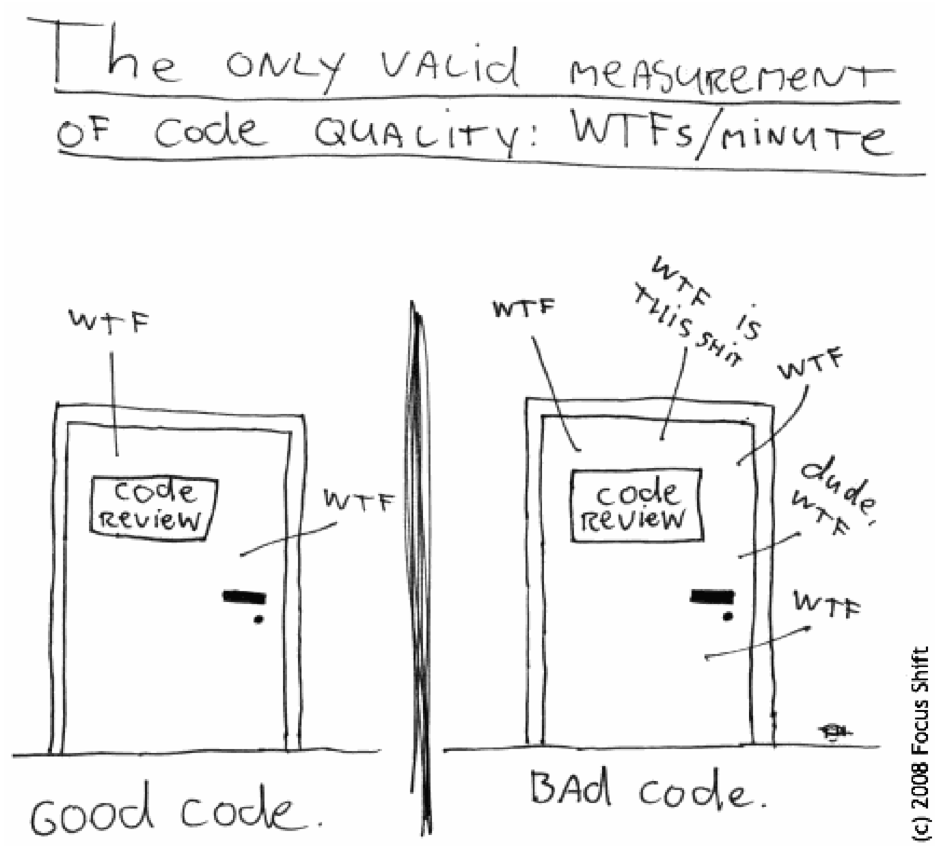
A class is essentially a template for an object. It defines what an object will look like and what functions the object will have. Every object is an object of some class.

A programmer is like a designer. A designer creates blueprints, and a Java programmer writes classes. Parts are created based on the blueprints, and objects are created based on classes.

Each class has data (fields) and methods for working with this data.

CODEGYM

Designing a program



What matters most for code is that it must be understandable to other programmers.

Incorrect code that is understandable can be corrected. Correct but incomprehensible code cannot be improved. All you can do is discard it and rewrite the code.

Important:
- To write good and understandable code inside methods
- Decide which entities should be included in the program
- Split the program into logical parts correctly

Objects and constructors

Each class has a special method (or methods) that are responsible for handling arguments passed when creating an object. These methods are called **constructors**. Or when we're talking about just one: the constructor.

Distinctive features of constructors:

- The name of a constructor is the same as the name of its class (and starts with a capital letter)
- A constructor has no return type.

```
modifiers Class(parameters) {
   Code
}
```

A constructor is similar to a regular method, but it does not have a return type. What's more, the name of the constructor is the name of the class, with the same capitalization.

```
public Car(String model, int maxSpeed) {
   this.model = model;
   this.maxSpeed = maxSpeed;
}
```

Calling a constructor

When creating a new object using the **new** operator and a command like new Class(arguments), two things happen:

- The Java machine creates an object whose type is Class
- The Java machine calls the object's constructor and passes in your arguments

CODEGYM

Multiple constructors

You can add multiple constructors to a class. There is no limit on the number of constructors or their parameters. When you create an object, the compiler automatically selects the constructor that matches the parameters

Default constructor

If a class declares not one constructor, the compiler will add a default constructor, which is a no-argument constructor with the public modifier.

But if a class declares even one constructor, then no default constructor will be added and you'll need to add it yourself if you want it.

CODEGYM

Initializing variables

Variables can be initialized in a constructor as well.

Here's what will happen:

- A Type object is created.
- All instance variables are initialized with their initial values.
- The constructor is called and its code is executed.

The variables first get their initial values, and only then is the code of the constructor executed.

Order of initialization of variables in a class

Variables are not merely initialized before the constructor runs — they are initialized in a well-defined order: the order in which they are declared in the class.

```java
public class Payment {
   int price = 100;
   double tax = 0.2;
   double totalPrice = Math.ceil(price * (1 + tax));
}
```

Code in a constructor

In theory, you can write code of any complexity in a constructor.
But don't do this.

Complex logic implies a high probability of errors and that means the code must handle exceptions correctly.

Base class constructor

Variables of a class are initialized before the constructor is called. A base class gets initialized fully before the initialization of the inherited class.

Properties: getters and setters

It is customary to make all class fields private in Java. Only the class's methods can modify the variables of the class. No methods from other classes can directly access the variables.

If you want other classes to be able to get or change the data inside objects of your class, you need to add two methods to your class — a get method and a set method.

The name comes from the English words "get", which means "to get" (i.e. "a method for getting the field value") and "set", which means "to set" (i.e. "a method for setting the field value").

```
private int price = 100;

public int getPrice() {
   return price;
}

public void setPrice(int price) {
   this.price = price;
}
```

Comparing objects in Java

In Java, objects can be compared both by reference and by value.

Object class
All classes in Java inherit the Object class. And if a class inherits the Object class, then it gains all the methods of the Object class. And this a major consequence of inheritance. These inherited methods include methods related to object comparison. These are the **equals**() and **hashCode**() methods.

```
class Person {
   String name;
   int age;

   public boolean equals(Object obj) {
      return this == obj;
   }

   public int hashCode() {
      // This is the default implementation, but a different implementation is possible
      return address_of_object_in_memory;
   }
}
```

hashCode() method

The hashCode() method returns a fixed-length numerical value for any object. In Java, the hashCode() method returns a 32-bit number (int) for any object.

Comparing two numbers is much faster than comparing two objects using the equals() method, especially if that method considers many fields.

If you call it on an object, it returns some number — analogous to the first letter in a word. This number has the following properties:

- Identical objects always have the same hashcode
- Different objects can have the same hashcode, or their hashcodes can be different
- If objects have different hashcodes, then the objects are definitely different

Java programmers have a universal agreement that if they write their own implementation of the equals() method and thereby override the standard implementation (in the Object class), they must also write their own implementation of the hashCode() method in such a way that the aforementioned rules are satisfied.

This arrangement is called a contract.

Homework

MODULE 1. JAVA SYNTAX

Complete Level 13

MODULE 1. JAVA SYNTAX

CODEGYM

Answers to questions

MODULE 1. JAVA SYNTAX