CODEGYM

Mentor-supported training program from Codegym

# Java developer in 12 months

MODULE 2. JAVA CORE

## Lesson #4

## OOP interfaces

CODEGYM

# Lesson plan

- Interfaces
- Default methods in interfaces
- Multiple inheritance of interfaces
- Abstract class vs interface

CODEGYM

# Interfaces

Interface is a child of Abstraction and Polymorphism.
It has a strong resemblance to an abstract class, in which all methods are abstract.

It is declared in the same way as a class, but instead of the word class we specify the interface keyword.

```
public interface Swimmable {
    public void swim();
}
```

We created the Swimmable interface — i.e. “can swim” – with a single swim() method.

# Advantages over classes

### 1. Separating “method declarations” from their implementation

We hide not only various implementations, but even the class itself that contains it (everywhere in the code only an interface can appear). This allows you to very flexibly, right in the process of program execution, replace one object with another, changing the object's behavior invisibly from all classes that use it.

### 2. Multiple inheritance

Java banned multiple inheritance of classes, but allowed multiple inheritance of interfaces. An interface can have multiple parent interfaces. A class can have multiple parent interfaces and only one parent class.

CODEGYM

# A class is usually a model of a specific object

An interface, on the other hand, corresponds to an object's abilities or roles.

For example, things like cars, bikes, motorcycles, and wheels are best represented as classes and objects. And their abilities such as "I can ride", "I can transport people", "I can stand" - are best presented in the form of interfaces.

Interfaces greatly simplify the life of a programmer. There is no need to write code to interact with all classes. it is enough to interact with their roles (interfaces).

CODEGYM

# Supporting declared behavior

An interface is a standardized way for two things to interact, and the standard is known to the two parties.

When a person tells the dog to "sit", he gives a command according to the "dog voice control interface", and if the dog executes this command, then we say that the dog supports this interface.

Methods are actions on an object, on its data. And if the class implements certain methods, then it "supports the execution" of certain commands.

CODEGYM

# Benefits of combining methods into an interface

1. Each interface, same as a class, has a unique name
Both parties can be 100% certain that the other party supports the required (known to them) interface, rather than a similar one.

2) Each interface imposes certain restrictions on the class that is going to support it.
The class itself decides (its developer) what it will do in case of a call to its methods that it inherited from the interface, but the result must be within expectations.

CODEGYM

**To support the implementation of some interface (or a group of interfaces) in your class, you need to:**

1. Inherit from them.
2. Implement the methods declared in them.
3. Methods should do what they are intended for.

CODEGYM

# Default methods in interfaces

An interface does not implement a behavior. Its task is to describe what behavior all objects that implement it should have.

Since Java 8, it has become possible to define default methods and implement them right inside an interface!

```
//Default interface methods can be overridden.
public class UnusualCar implements Car {
    @Override
    public void gas() {
        System.out.println("This car accelerates differently!");
    }
}
```

CODEGYM

# Multiple inheritance of interfaces

Java allows multiple implementations of interfaces.

If a class cannot inherit from multiple parents, implementing multiple interfaces is easy! Our duck can both fly and float :)

```
public class Duck implements FlyingBird, Waterfowl {
    // methods of both interfaces easily combined in a single class
    @Override
    public void fly() {
        System.out.println("Flying!");
    }
    @Override
    public void swim() {
        System.out.println("Swimming!");
    }
}
```

CODEGYM

# Abstract class vs interface

| Abstract class | Interface |
|---|---|
| **Inheritance** | |
| An abstract class can only inherit a single class and any number of interfaces. | An interface can not inherit from classes, but can inherit from any number of interfaces. |
| **Abstract methods** | |
| An abstract class may contain abstract methods. But it may not contain them at all. | All non-static and non-default interface methods are abstract (without implementation). An interface may not contain any methods at all. |
| **Methods with implementation** | |
| An abstract class may contain methods with implementation. | An interface may contain default methods. |
| **Data** | |
| No limits. | An interface only contains public final static data. |
| **Object creation** | |
| An object of an abstract class can not be created. | An object of an interface can not be created. |

CODEGYM

# Homework

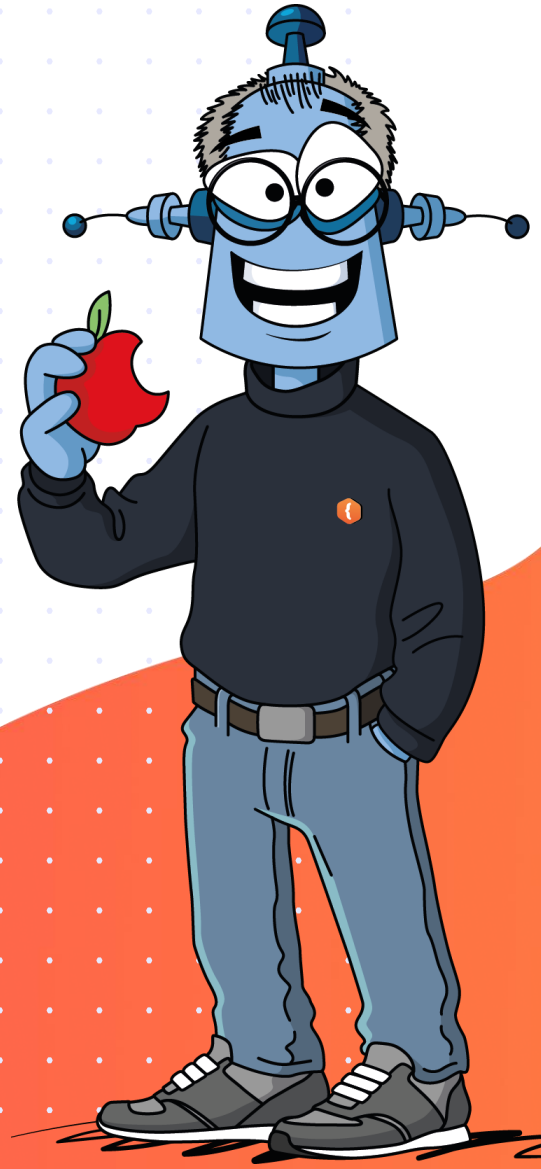
## Module 2

Level 4. OOP interfaces


MODULE 2. JAVA CORE

CODEGYM

# Questions and Answers



MODULE 2. JAVA CORE