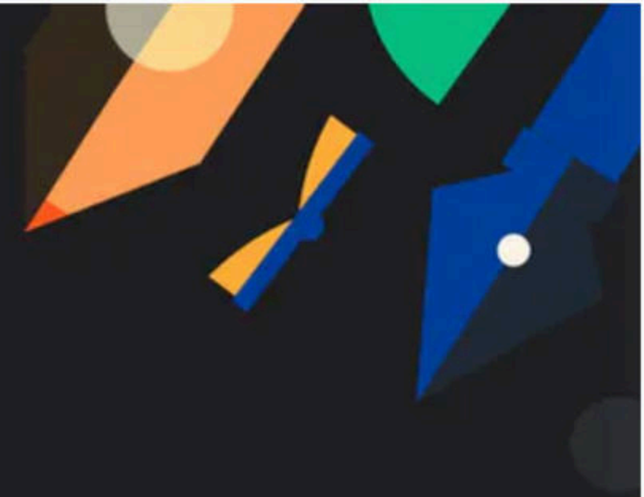


Java OOPS Concepts - Part II

Course on Programming Basics

Anuj Kumar Sharma • Lesson 4 • June 4, 2022

Java OOPS Concepts

Topics in Today's class

1. Classes & Objects
2. Methods
3. Inheritance
4. Polymorphism
5. Encapsulation
6. Abstraction (will be covered later in Collections Framework class)

Classes & Objects

Java is an object-oriented programming language. The core concept of the object-oriented approach is to break complex problems into smaller **objects**.

An **object** is any entity that has a state and behavior. For example, a bicycle is an object. It has

States: idle, first gear, etc

Behaviors: braking, accelerating, etc.

Creating a class

We can create a class in Java using the class keyword. For example,

```
class ClassName {
  // fields
  // methods
}
```

Creating an object

We have used the new keyword along with the constructor of the class to create an object. Constructors are similar to methods and have the same name as the class.

```
class Person {

        int age;

        String name;

}

Person p1 = new Person();
```

Methods Overloading

In Java, two or more methods may have the same name if they differ in parameters (different number of parameters, different types of parameters, or both). These methods are called overloaded methods and this feature is called method overloading.

```
void func() { ... }
void func(int a) { ... }
float func(double a) { ... }
float func(int a, float b) { ... }
```

Java Constructors

A constructor in Java is similar to a method that is invoked when an object of the class is created.

Unlike Java methods, a constructor has the same name as that of the class and does not have any return type. For example,

```
class Test {
  Test() {
    // constructor body
  }
}
```

Java static keyword

In Java, if we want to access class members, we must first create an instance of the class. But there will be situations where we want to access class members without creating any variables.

In those situations, we can use the **static** keyword in Java. If we want to access class members without creating an instance of the class, we need to declare the class members static.

static methods and static variables

Java this keyword

In Java, **this** keyword is used to refer to the current object inside a method or a constructor.

We can use **this** keyword for:

- Using **this** for Ambiguity Variable Names
- Using **this** in Constructor Overloading
- Passing **this** as an Argument

Methods Overloading

Method overloading is achieved by either:

- changing the number of arguments.
- or changing the data type of arguments.

It is not method overloading if we only change the return type of methods. There must be differences in the number of parameters.

Access Modifiers

In Java, access modifiers are used to set the accessibility (visibility) of classes, interfaces, variables, methods, constructors, data members, and the setter methods. For example,

```
class Animal {
    public void method1() {...}

   private void method2() {...}
}
```

Types of Access Modifiers

There are four access modifiers keywords in Java and they are:

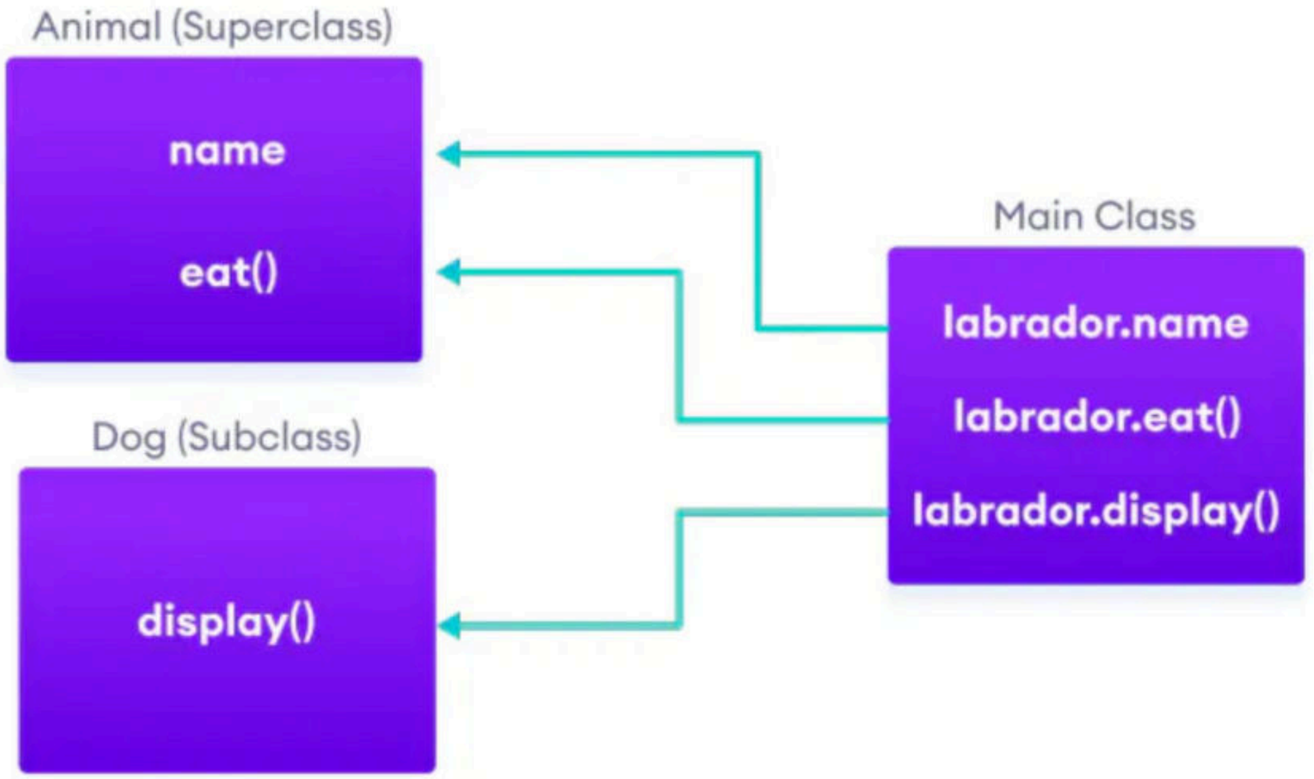
Modifier	Description
Default	declarations are visible only within the package (package private)
Private	declarations are visible within the class only
Protected	declarations are visible within the package or all subclasses
Public	declarations are visible everywhere

Java Inheritance

Inheritance is one of the key features of OOP that allows us to create a new class from an existing class.

The new class that is created is known as subclass (child or derived class) and the existing class from where the child class is derived is known as superclass (parent or base class).

The `extends` keyword is used to perform inheritance in Java.


Animal (Superclass)
name
eat()
Dog (Subclass)
display()
Main Class
labrador.name
labrador.eat()
labrador.display()

Java Polymorphism

Polymorphism is an important concept of object-oriented programming. It simply means more than one form.

That is, the same entity (method or operator or object) can perform different operations in different scenarios.

We can achieve polymorphism in Java using the following ways:

- Method Overriding
- Method Overloading

Java Encapsulation

Encapsulation refers to the bundling of fields and methods inside a single class.

It prevents outer classes from accessing and changing fields and methods of a class. This also helps to achieve data hiding.

Encapsulation refers to the bundling of related fields and methods together. This can be used to achieve data hiding. **Encapsulation in itself is not data hiding.**

Java Encapsulation

Encapsulation refers to the bundling of fields and methods inside a single class.

It prevents outer classes from accessing and changing fields and methods of a class. This also helps to achieve data hiding.

Encapsulation refers to the bundling of related fields and methods together. This can be used to achieve data hiding. **Encapsulation in itself is not data hiding.**