Constructing an Object, Calling a Constructor from a Constructor

José Paumard

PHD, Java Champion, JavaOne RockStar

@JosePaumard https://github.com/JosePaumard

Agenda

How object are constructed

How constructors are called

How they call each other implicitly

How you can call a constructor from another constructor explicitly

Constructing an Object

```
public class City {

}

City city = new City();
```

Calling new invokes a constructor

Where is this constructor?

Every class has at least one constructor.

If no constructor is declared
then the compiler adds
an empty, no-args constructor.

```java
public class City {
    public City() {
    }
}

City city = new City();
```

The compiler creates this empty no-arg constructor

If no other constructor is declared

So you do not need to add it

Rules for constructors:

1) no constructors = the default empty constructor is added by the compiler

2) explicit constructor = no default empty constructor

You may create several constructors

No constructor =
a default constructor is added

An explicit constructor =
no default constructor is added

```
public class City {

    public City(String name) {

    }

    public City(String name, int population) {
    }
}
```

A constructor can call another constructor from the same class

```
public class City {

    public City(String name) {
        this(name, 0);
    }

    public City(String name, int population) {
    }
}
```

A constructor can call another constructor from the same class

This call must be the first line in the constructor

Rules for constructors:

3) a constructor must call a constructor from its superclass. If there is no explicit call, then a call to the empty no-arg constructor is added by the compiler.

A constructor must call
a constructor from its class
or from its super class

```
public class City {

    public City(String name) {

        this.name = name;
    }
}
```

When you write this code

```java
public class City extends Object {

    public City(String name) {
        super();
        this.name = name;
    }
}
```

The compiler compiles this code

super() calls the empty constructor of the Object class

```java
public class City {

    public City(String name) {
        this.name = name;
    }
}
```

```java
public class Capital
extends City {

}
```

The Capital class does not compile!

```java
public class City {

    public City(String name) {
        this.name = name;
    }
}
```

```java
public class Capital
extends City {

    public Capital() {
        super();
    }
}
```

Because the City class has no empty constructor

```
public class City {

    public City() {
    }

    public City(String name) {
        this.name = name;
    }
}
```

```
public class Capital
extends City {

    public Capital() {
        super();
    }
}
```

Adding this empty constructor explicitly fixes this code

Demo

Writing classes and playing with constructors

Module Wrap Up

What did you learn?

How constructors are working

1) Every class has a constructor

2) Every constructor calls a constructor

- from this class with this()

- from its super class with super()

Course Wrap Up

What did you learn?

The implementation of the four principles of object-oriented programming in Java:

- abstraction

- encapsulation

- inheritance

- polymorphism

Up Next: Pass the Certification!