

# JAVA FUNDAMENTALS COURSE

## EXERCISE

# CONSTRUCTOR OVERLOADING

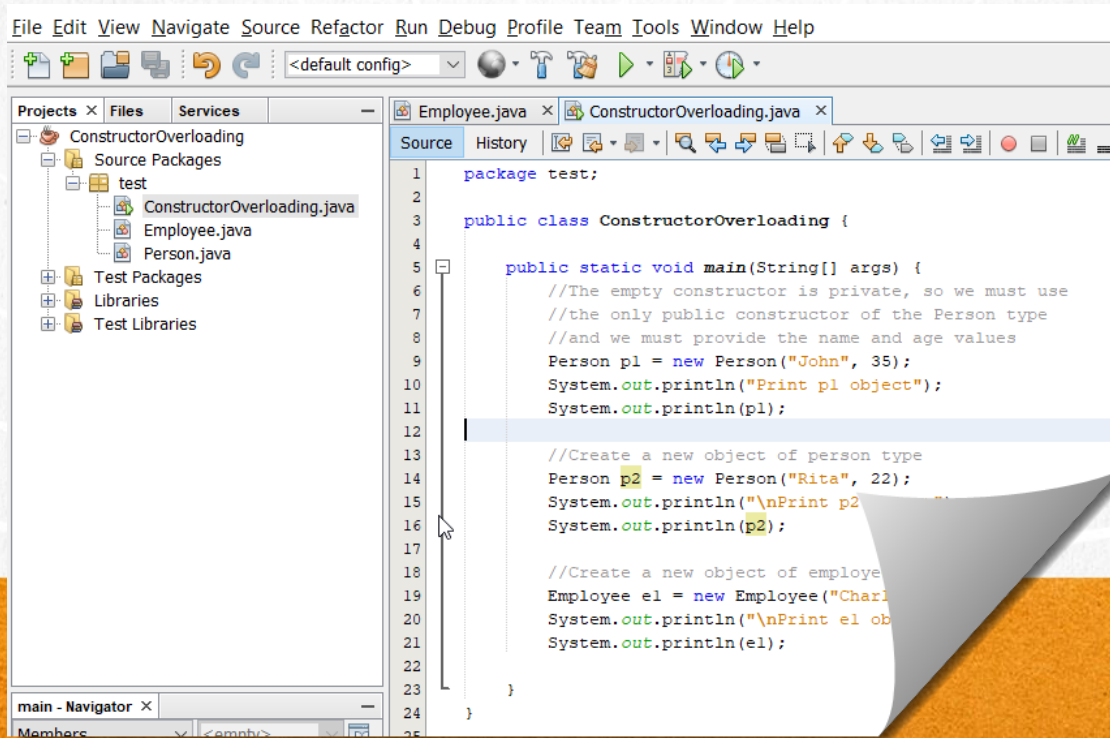


JAVA FUNDAMENTALS COURSE

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

# EXERCISE OBJECTIVE

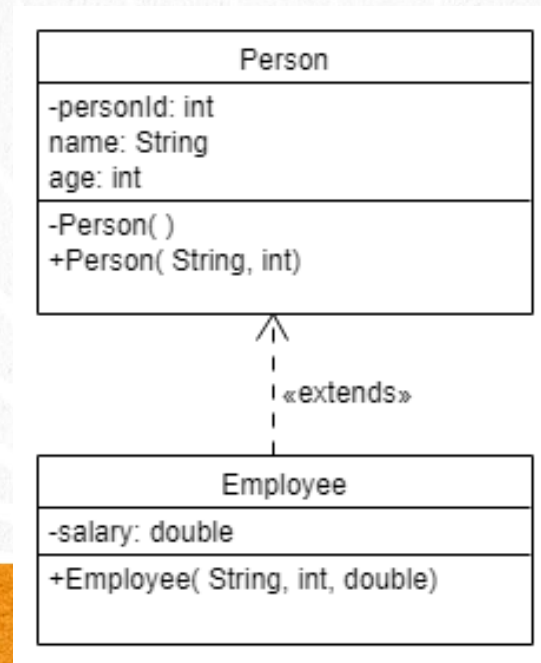
Create an exercise to apply the concept of constructor overloading in Java. At the end we should observe the following:

A screenshot of an IDE window showing a Java project named 'ConstructorOverloading'. The project structure on the left includes 'Source Packages' with a 'test' package containing 'ConstructorOverloading.java', 'Employee.java', and 'Person.java'. The main editor displays the source code of 'ConstructorOverloading.java'. The code defines a 'test' package and a 'ConstructorOverloading' class. It includes a 'main' method that creates and prints objects of 'Person' and 'Employee' classes. The 'Person' class has a private empty constructor and a public constructor taking 'name' and 'age' parameters. The 'Employee' class has a public constructor taking 'name' and 'age' parameters. The 'main' method creates a 'Person' object 'p1' with name 'John' and age 35, prints it, creates a 'Person' object 'p2' with name 'Rita' and age 22, prints it, and creates an 'Employee' object 'e1' with name 'Charles' and age 25, prints it.

```
1 package test;
2
3 public class ConstructorOverloading {
4
5     public static void main(String[] args) {
6         //The empty constructor is private, so we must use
7         //the only public constructor of the Person type
8         //and we must provide the name and age values
9         Person p1 = new Person("John", 35);
10        System.out.println("Print p1 object");
11        System.out.println(p1);
12
13        //Create a new object of person type
14        Person p2 = new Person("Rita", 22);
15        System.out.println("\nPrint p2 object");
16        System.out.println(p2);
17
18        //Create a new object of employee type
19        Employee e1 = new Employee("Charles", 25);
20        System.out.println("\nPrint e1 object");
21        System.out.println(e1);
22    }
23 }
24
25 }
```

# CLASS DIAGRAM

The following is a class diagram of the exercise, created with the tool <http://www.umlet.com/umletino/umletino.html>





# 1. CREATE THE PROJECT

Create a new Project:

**New Java Application**

**Steps**

1. Choose Project
2. **Name and Location**

**Name and Location**

Project Name: ConstructorOverloading

Project Location: C:\Courses\JavaFundamentals\Lesson19 **Browse...**

Project Folder: C:\Courses\JavaFundamentals\Lesson19\ConstructorOverloading

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder: **Browse...**

Different users and projects can share the same compilation libraries (see Help for details).

☐ Create Main Class constructoroverloading.ConstructorOverloading

**< Back** **Next >** **Finish** **Cancel** **H**elp

**JAVA FUNDAMENTALS COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

## 2. CREATE A NEW CLASS

**New Java Class**

**Steps**

1. Choose File Type
2. **Name and Location**

**Name and Location**

Class Name:

Project:

Location:

Package:

Created File:

< Back   Next >   **Finish**   Cancel   Help

**JAVA FUNDAMENTALS COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

# 3. MODIFY THE CODE

## Person.java:

```
package test;

public class Person {

    private int personId;
    private String name;
    private int age;
    private static int peopleCounter;

    //Constructor with no arguments and private
    //Assigns the personId value
    private Person() {
        this.personId = ++peopleCounter;
    }

    //Full constructor overloaded
    public Person(String name, int age) {
        //The empty constructor is called
        this();//must be the first line in the constructor if used
        this.name = name;
        this.age = age;
    }

    @Override
    public String toString() {
        return "Person{" + " personId=" + personId + ", name=" + name + ", age=" + age + '}';
    }
}
```

## 4. CREATE THE EMPLOYEE CLASS

**New Java Class**

**Steps**

1. Choose File Type
2. **Name and Location**

**Name and Location**

Class Name:

Project:

Location:

Package:

Created File:

< Back   Next >   **Finish**   Cancel   Help

# 5. MODIFY THE CODE

## Employee.java:

```
package test;

public class Employee extends Person {

    private double salary;

    public Employee(String nombre, int edad, double sueldo) {
        super(nombre, edad); //Super must be the first line if used
        this.salary = sueldo;
    }

    public double getSalary() {
        return salary;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }

    @Override
    public String toString() {
        //First we call the toString method of the parent class
        //after that we concatenate the attributes of the child class
        return super.toString() + " Employee{salary=" + salary + "}";
    }
}
```



## 6. CREATE THE NEW CLASS

**New Java Class**

**Steps**

1. Choose File Type
2. **Name and Location**

**Name and Location**

Class Name:

Project:

Location:

Package:

Created File:

< Back   Next >   **Finish**   Cancel   Help

# 6. MODIFY THE CODE

## ConstructorOverloading.java:

```
package test;

public class ConstructorOverloading {

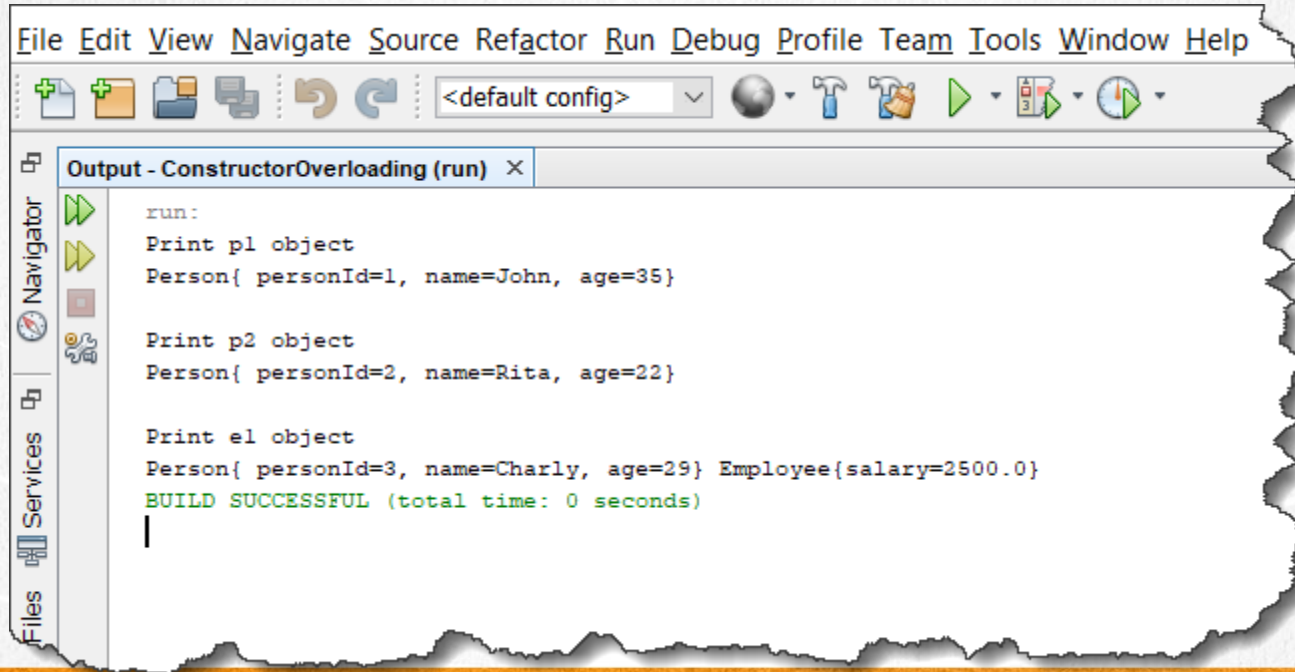
    public static void main(String[] args) {
        //The empty constructor is private, so we must use
        //the only public constructor of the Person type
        //and we must provide the name and age values
        Person p1 = new Person("John", 35);
        System.out.println("Print p1 object");
        System.out.println(p1);

        //Create a new object of person type
        Person p2 = new Person("Rita", 22);
        System.out.println("\nPrint p2 object");
        System.out.println(p2);

        //Create a new object of employee type
        Employee e1 = new Employee("Charly", 29, 2500);
        System.out.println("\nPrint e1 object");
        System.out.println(e1);
    }
}
```

# 7. EXECUTE THE PROJECT

Execute the Project and the result is as follows:

The image shows a screenshot of an IDE's 'Run Output' window. The window title is 'Output - ConstructorOverloading (run)'. The output text is as follows:

```
run:
Print p1 object
Person{ personId=1, name=John, age=35}

Print p2 object
Person{ personId=2, name=Rita, age=22}

Print e1 object
Person{ personId=3, name=Charly, age=29} Employee{salary=2500.0}
BUILD SUCCESSFUL (total time: 0 seconds)
```

The IDE interface includes a menu bar (File, Edit, View, Navigate, Source, Refactor, Run, Debug, Profile, Team, Tools, Window, Help) and a toolbar with icons for file operations and running. On the left, there is a sidebar with 'Files', 'Services', 'Navigator', and 'Run and Debug' views.

**JAVA FUNDAMENTALS COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)

# EXERCISE CONCLUSION

- With this exercise we have put into practice the concept of Constructors Overloading.
- We also saw several related topics, which have to do with the good design of our classes.
- We are already beginning to apply several issues that we have been working on, so it is important that they remain clear, as we will continue using them in the following lessons.



**ONLINE ONLINE**

# **JAVA FUNDAMENTALS**

---

By: Eng. Ubaldo Acosta



**JAVA FUNDAMENTALS COURSE**

[www.globalmentoring.com.mx](http://www.globalmentoring.com.mx)