EXERCISE

CONSTRUCTOR OVERLOADING

EXERCISE OBJECTIVE

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

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

<default config>

Projects | Files | Services

- ConstructorOverloading
  - Source Packages
    - test
      - ConstructorOverloading.java
      - Employee.java
      - Person.java
  - Test Packages
  - Libraries
  - Test Libraries

main - Navigator

Members <empty>

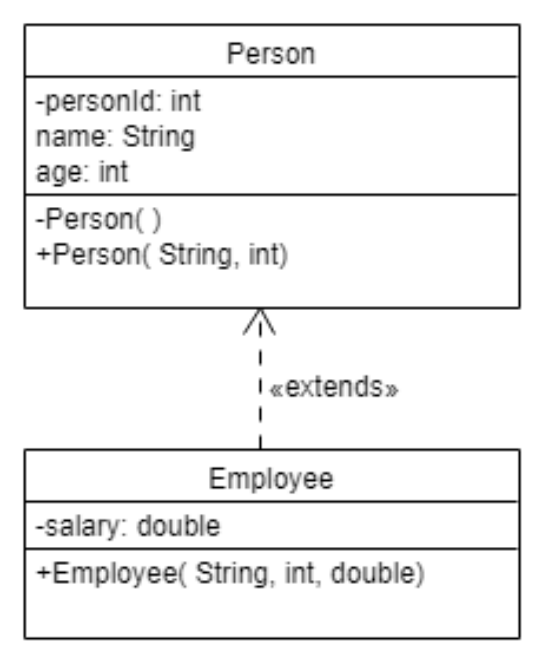
Employee.java | ConstructorOverloading.java

Source | History

```
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
16          System.out.println(p2);
17
18          //Create a new object of employe
19          Employee e1 = new Employee("Charl
20          System.out.println("\nPrint e1 ob
21          System.out.println(e1);
22
23      }
24  }
```

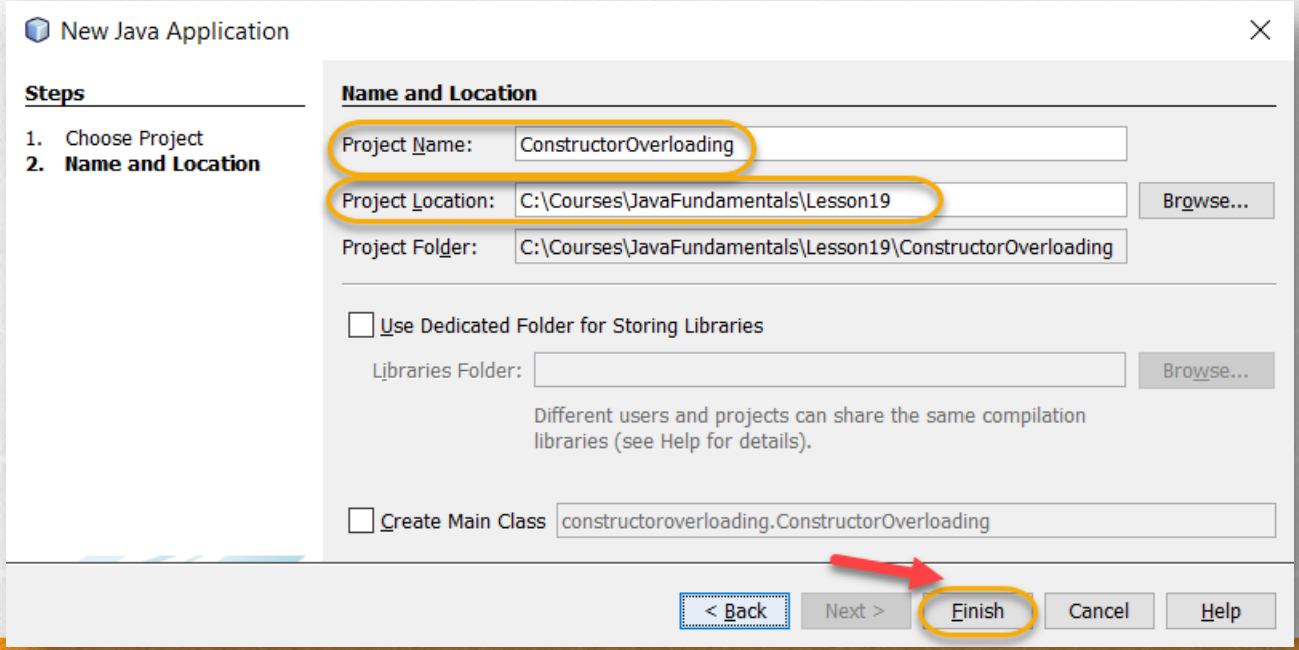
CLASS DIAGRAM

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

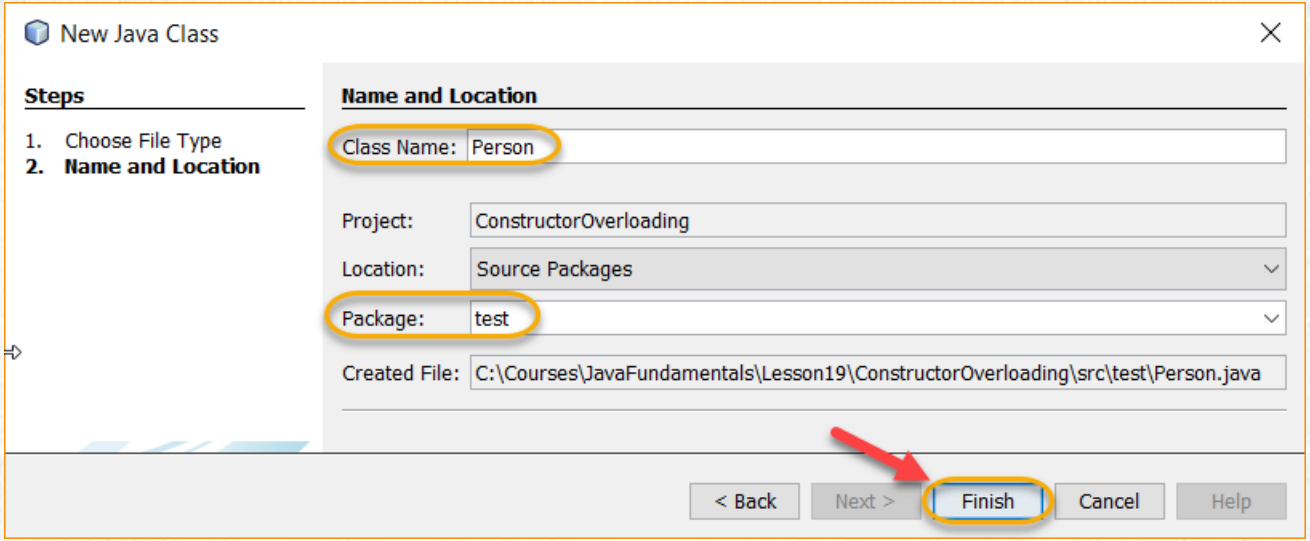


1. CREATE THE PROJECT

Create a new Project:



2. CREATE A NEW CLASS

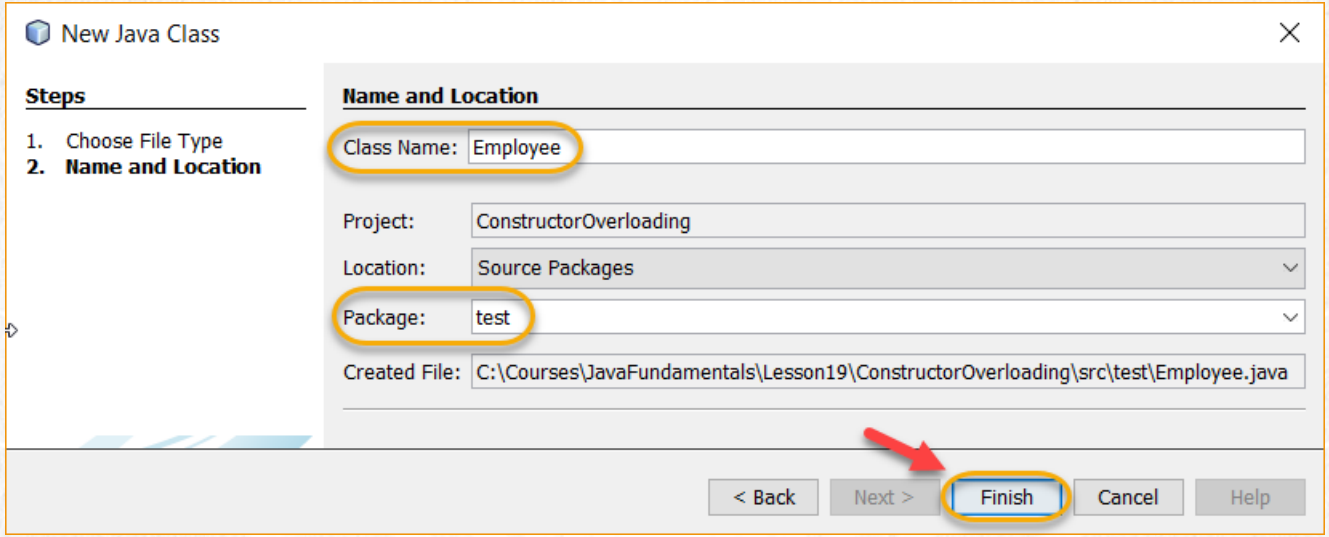


3. MODIFY THE CODE

Person.java:

```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

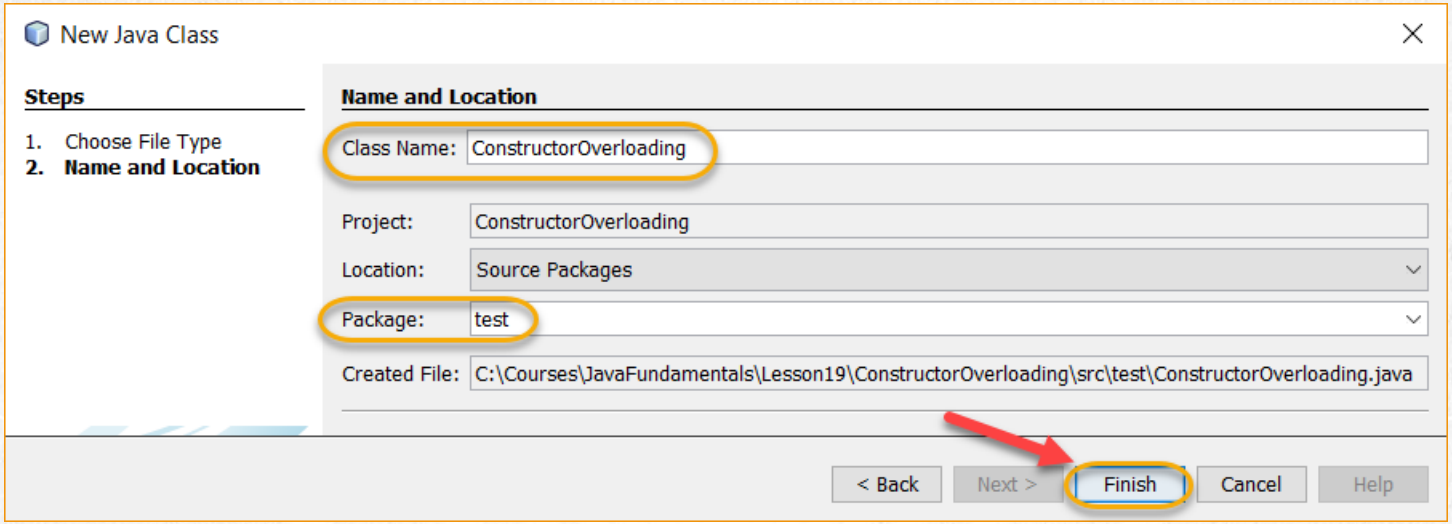


5. MODIFY THE CODE

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



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:

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

<default config>

Output - ConstructorOverloading (run)

```
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)
```

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