Exercise

Alexandre Bergel
Nancy Hitschfeld

30/09/2020

Preamble

Java offers numerous way to define collections

The most used and easiest to use is `ArrayList<type>`

Preamble

For example:

```
ArrayList<String> aCollection = new ArrayList<>();
aCollection.add(“Hello ”);
aCollection.add(“C3002”);
for(String s : aCollection) {
        System.out.print(s);
}
```

Preamble exercise

Write a short Java program that manipulate collections

The class `C` has three methods:

- `fill(ArrayList<Integer>)` to add numbers in the collection
- `print(ArrayList<Integer>)` to print the collection
- A `main` method to run an example

You can use random numbers in fill:

```
new Random().nextInt(42)
```

Here is a possible solution

```
package smallexercise;

import java.util.ArrayList;
import java.util.Random;

public class C {
    public void fill(ArrayList<Integer> aList) {
        Random r = new Random(2020);
        for(int nb = 0; nb < 10; nb++) {
            aList.add(r.nextInt(42));
        }
    }

    public void print(ArrayList<Integer> aList) {
        for(int i : aList) {
            System.out.println(i);
        }
    }

    public static void main(String[] args) {
        ArrayList<Integer> aCollection = new ArrayList<>();
        C c = new C();
        c.fill(aCollection);
        c.print(aCollection);
    }
}
```

Here is a unit test that test the class C

```java
import org.junit.Before;
import org.junit.Test;
import java.lang.reflect.Array;
import java.util.ArrayList;
import java.util.Arrays;
import static org.junit.Assert.assertEquals;

public class CTest {
    private C c;
    private ArrayList<Integer> aCollection;

    @Before
    public void setUp() {
        c = new C();
        aCollection = new ArrayList<>();
    }

    @Test
    public void testFill() {
        assertEquals(0, aCollection.size());
        c.fill(aCollection);
        assertEquals(10, aCollection.size());

        // We verify some numbers are in the collection
        // since we have a seed for the random number generator
        // we know it will always produce the same sequence of numbers
        // But this method is not really elegant, and it is painful to write
        assertEquals((long) 22, (long) aCollection.get(0));
        assertEquals((long) 26, (long) aCollection.get(1));
        assertEquals((long) 18, (long) aCollection.get(2));

        // Here is the list of number we expected
        Integer[] expected = {22, 26, 18, 10, 37, 18, 35, 16, 25, 40};

        // We are here testing that the collection is filled with the
        // expected sequence of numbers
        assertEquals(Arrays.asList(expected), aCollection);
    }
}
```

Exercise

The University decided as ask *you* to design a new computational system to handle students, auxiliares, professors and ramos

We will go through *two different steps*:

1 - Design ramos

2 - Design students, auxiliares, professors

Designing ramos

A ramo has *one professor* in charge of the lecture, some *auxiliares*, and some *students*

How would you *design* ramos?

What would be the *motivations* of your design?

What are the *positive* and *negative* aspects of your design?

What would be the *responsibilities* of a ramo object?

Designing People

A student can be part of several ramos

An auxiliar can be a student

How would you design your system?

```java
package department;

import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.*;

public class DepartmentTest {
    private Ramo cc3002;
    private Professor prof;
    private Student uwu, owo, awa;
    private Auxiliare tomas, slater;

    @Before
    public void setUp() {
        prof = new Professor("Michael");
        cc3002 = new Ramo("CC3002", prof);
        uwu = new Student("Uwu");
        owo = new Student("Owo");
        awa = new Student("AwA");

        tomas = new Auxiliare("Tomas");
        slater = new Auxiliare("Slater");
    }

    @Test
    public void testEmpty() {
        assertTrue(cc3002.isEmpty());
        assertEquals(0, cc3002.numberOfStudents());
        assertEquals(0, cc3002.numberOfAuxiliares());
    }

    @Test
    public void testProfessor() {
        assertEquals(prof, cc3002.getProfessor());
    }

    @Test
    public void testWithStudent() {
        cc3002.addStudent(uwu);
        cc3002.addStudent(awa);
        cc3002.addStudent(owo);

        assertFalse(cc3002.isEmpty());
        assertEquals(3, cc3002.numberOfStudents());
        assertEquals(0, cc3002.numberOfAuxiliares());
    }

      public void testAuxiliaresAndStudents() {
          cc3002.addAuxiliare(slater);
          cc3002.addAuxiliare(tomas);
          assertEquals(2, cc3002.numberOfAuxiliares()
      }
}
```

```
package department;

import java.util.ArrayList;

class Ramo {
    protected Professor professor;
    protected ArrayList<Student> students;
    protected ArrayList<Auxiliare> auxiliares;
    private final String code;

    public Ramo(String aCode, Professor aProfessor) {
        code = aCode;
        professor = aProfessor;
        students = new ArrayList<>();
        auxiliares = new ArrayList<>();
    }
   public boolean isEmpty() {
        return students.isEmpty() && auxiliares.isEmpty();
    }
    public int numberOfStudents() { return students.size(); }
    public int numberOfAuxiliares() { return auxiliares.size(); }
    public Professor getProfessor() { return professor; }

    public void addStudent(Student student) {
        students.add(student);
    }
    public void addAuxiliare(Auxiliare auxiliare) {
        auxiliares.add(auxiliare);
    }
}
```

```
package department;

public class Auxiliare extends Person {
    public Auxiliare(String name) {
        super(name);
    }
}
```

```
package department;

public class Person {
    protected String name;

    public Person(String aName) {
        name = aName;
    }
}
```

```
package department;

public class Student extends Person {

    public Student(String aName) {
        super(aName);
    }
}
```

```
package department;

public class Professor extends Person {

    public Professor(String aName) {
        super(aName);
    }
}
```

License

dcc

CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE CHILE

www.dcc.uchile.cl

/ DCCUCHILE