ByteXL

Programming 2024

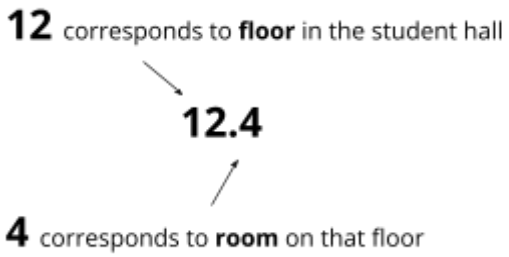
Coursework: “StudentHall App”

Background

A fictional Estonian-UK company, **Vara,** wants to develop an app that allows a university to manage its accommodation. This year the company has made the decision to build their own Student Hall app. The StudentHall app can register students in a student hall and maintain information on the number of students registered to a hall and if they are on the premises. It will also help the Student Hall cleaning team.

Overview

This coursework focuses on the classes required to support the app. It is not concerned with building a graphical front end. All interaction and testing will be carried out in a console front end. The coursework is broken down into several tasks, each with accompanying testing. The testing will be used by the assessors to evaluate functionality. The coursework must be developed as a single **project**. The core class for the app is a **Student** class. It represents a student at the university and should have three member variables: a name, a room location and an onPremises variable to indicate whether the student is in the student hall. The room location uses the following system:



The location, represented by a double, is made of two parts: the integer part (12 in the example above) and a fractional part (4 in the example above). The integer part corresponds to the **floor** in a particular student hall. The fractional part refers to the actual **room** on that floor. Each location is assumed to be unique.

The **Student** will be managed within a **StudentHall** class. The **StudentHall** class provides a number of methods to populate, display, analyse and update values of an array of Student objects. The **StudentHall** class will also allow students to be registered, and control the onPremises value for all students.

Tasks

Task 1	**Student class** (Student.java)	**10 marks**
	Tests (StudentTest.java)	**25 marks**

This Task focuses on the design, creation and testing of the **Student** class. Assessment of code quality will be based on the **Student** class design and code. Assessment of functionality will be based on execution of a separate test class (StudentTest.java).

Design and **write** the **Student** class (Student.java). This class should have three private member variables: name (String), roomLocation (double) and onPremises (boolean). All member variables should only be accessed and updated through standard get/set methods which you should also write.

Design and **write** a **constructor** which must allow all three member variables to be populated.

Design and **write exitHall()** and **enterHall()** methods. The exitHall() method sets the onPremises variable to false. The enterHall() method sets the onPremises variable to true.

Design and write a **toString()** method**.** This method should return a well-formatted String containing the variable name-value pairs stored inside the object. For example:

```
Name:              Petse Buboni
Room location:     12.1
on Premises:       false
```

Create a test script file (**StudentTest.java)** to **test** the **Student** class. All test sequences should be placed inside the main() method of this class and use the console for input and output. This class will be executed when we assess the functionality of your work.

Your testing must include all of the following tests, one after the other.

1. Create a Student object in the main() method:
    a. Using console input, allow the user to specify the name, roomLocation and onPremises status of the student. Test the data is correctly stored in the object with the toString() method to display the details of the Student.

(Example output)

```
----------
-STUDENT -
----------
Name:             Petse Buboni
Location:         12.1
on Premises:      false
----------
```

2. The Student enters the Hall:
    a. Execute an enterHall() method on the student object
    b. Verify that the procedure worked correctly by using the Student object toString() method to display the updated details of the student.
3. A fire safety alarm has been called:
    a. Execute an exitHall() method on the student object
    b. Verify that the procedure worked correctly by using the Student object toString() method to display the updated details of the student.

Task 2	**StudentHall class** (StudentHall.java)	**20 marks**
	Tests (StudentHallTest.java)	**45 marks**

This task focuses on the design, creation and testing of the **StudentHall** class. Assessment of the code quality will be based on the **StudentHall** class design and code. Assessment of functionality will be based on execution of a separate test class (StudentHallTest.java).

Design and **write** the **StudentHall** class (StudentHall.java). This class should have only one private member variable: **an array of Student objects**.

Design and **write** a constructor. The constructor should take one parameter:

a size value. This represents the number of students that must be registered in the Student Hall. The constructor should create a new instance of the array using the size parameter. The initial array should be **unpopulated** - it should contain null values at every index.

Design and **write** a **registerStudent()** method. The registerStudent() method should take three parameters (each corresponding to the parameters in the Student class constructor): name, roomLocation and onPremises. It should build a Student object and then insert the Student object at the first empty (null) array position and fail if there is no empty position in the array.

Design and **write** an **update()** method. This method takes two parameters: the Student name (a search parameter) and a new onPremises value. The method attempts to locate the Student using the student name parameter. If it finds a Student with that name then it updates the onPremises value with the corresponding new onPremises value.

Design and **write** a **studentsInHall**() method. This method should count the number of students currently in the student hall. It should return the count.

Design and **write** a **studentsOnFloor()** method. This method takes one parameter: the floor value. This method should count the number of students currently on the floor specified by the parameter. It should return the count.

Design a **closeFloorForCleaning()** method. This method takes one parameter: the floor value. This method should remove all students currently on that floor and place them in a room Location 0.0

Design a **changeRoom()** method. This method takes two parameters: a student's name and new roomLocation. The method should attempt to locate a student through the student's name. If it finds the student it should change the roomLocation value. Only change rooms if no other student currently occupies the room.

Design and **write** a **fireAlarm**() method. This method should exit all students from the hall..

Design and write two **toString()** methods**.** The first **toString()** method should return a well-formatted String containing all student data. It should loop through the array and build a string using the Student object toString() method. Once this is built it then returns this. It assumes the array is fully populated. For example:

```
----------
-STUDENT 1-
----------
Name:              Petse Buboni
Location:          12.1
on Premises:       false
----------
----------
-STUDENT 2-
----------
Name:              Jonty Joe King
Location:          5.3
on Premises:       true
----------
----------
-STUDENT 3-
----------
Name:              Franklin Ruzell
Location:          3.4
on Premises:       false
---------
```

The second **toString()** method should take one parameter: an index value. The method attempts to locate a student using the index value and return the data of that student. If no student is found at the given index, then it should return “No student found”.

Advanced:
Design a **selectFloorForCleaning()** method. This method should work out the floor with the fewest students on the premises. It should return this floor number. If a floor has already been cleaned then it must return the floor with the next fewest students on the premises.

Create a test script file (**StudentHallTest.java)** to **test** the **StudentHall** class. All test sequences should be placed inside the main() method and use the console for input and output. This class will be executed when we assess the functionality of your work.

Your testing must include all of the following tests, one after the other.

1. Create the StudentHall object. You might want to assume the Student Hall has 3 floors, with 2 rooms on each floor for purposes of testing.
    a. Using console input, ask the user to specify the number of students (size) to be registered in a Student Hall.
    b. Create a StudentHall object using the size value given by the user.
    c. Populate the StudentHall object with students. This method should use a loop. Inside the loop it should (i) ask the user for a student

name, a roomLocation and an onPremises value, and then (ii) pass these three values as parameters to the registerStudent() method of the StudentHall object.
    d. The loop should stop once the array is fully populated. *This does not require all rooms to be populated.*
    e. Verify that the StudentHall is correctly populated using the StudentHall object toString() method.
2. A student leaves the Hall.
    a. Ask the user for the student name and a new onPremises value, and using the update() method, then update the onPremises value.
    b. Verify the value is updated using the StudentHall object's toString() method.
3. Execute a countOnPremises() on the StudentHall object
    a. Display the count value in main()
4. Execute a studentsOnFloor() and closeFloorForCleaning() on the StudentHall object.
    a. Ask the user for a floor value and using the studentsOnFloor() method, return the number of students in rooms on that floor.
    b. Execute the closeFloorForCleaning() method on that floor
    c. Verify the values are updated using the StudentHall object's toString() method.
5. Execute a changeRoom() on a particular student.
    a. Ask the user for the student name and a new free unoccupied roomLocation. Locate the student in the StudentHall object and then change the roomLocation using the roomLocation parameter.
    b. Verify the values are updated using the StudentHall object's toString() method.
6. Execute a fireAlarm() operation on the StudentHall object
    a. Execute the fireAlarm() method on the StudentHall.
    b. Verify the onPremises values are all false using the StudentHall object's toString() method.
7. Execute a selectFloorForCleaning() operation on the StudentHall object
    a. Execute the closeFloorForCleaning() method with the floor from the selectFloorForCleaning() method.
    b. Verify the students have all been removed from that floor using the StudentHall object's toString() method.
    c. Repeat 7a and verify a different floor is cleaned.

Java Libraries Permitted

This coursework should only use:

- java.util.Scanner;
- an agreed library

You should not use:

- java.util.Arrays - especially for toString()
- arraylists - as an alternative to arrays

Assessment

This coursework will be assessed in the following way:

1. The coursework will be unzipped and the .java files will be opened in Nimbus
2. Each **Task** test script will be executed, eg. StudentTest.java, StudentHallTest.java**.**
3. **Functionality** will be judged by script execution and input and output at the console.
4. **Code quality** will be judged on the underlying classes.