Module 2
Java Development

Assignment 8- MADP202- July 5, 2017

Due: Friday July 7, 6:00 pm:

Submission: Please send it to me on Slack as a direct message. Zip all files in a folder with the following format: Yourname_Assignment#_DueDate: for example: AlirezaDavoodi_Assignment8_July7.

Late submission Penalty Policy:

Between 6:01pm and 11:59pm: 25% deduction
Before 6 pm the day after the due date: 50% deduction
Before 6 pm 2 days after the due date: 75% deduction
After 48 hours late: 100% penalty but still you need to submit the assignment.
Lack of submission of 3 or more assignments: 50% of your entire assignments (7.5% of your course mark).
Lack of submission of 5 or more assignments: 100% of your entire assignments (15% of your course mark).
Remember you need to get at least 70% to pass the course.

Topics:

Iterator/Iterable and Observer/Observable Design Patterns

Problem 1: Iterator/Iterable

The *Book* class consists of the following properties:

- A *title* (String)
- A *category* (Enum BookCategory: [Science, History, Food, IT, Engineering, Novel])
- An *array of pages*

The Page class consists of the following properties:

- A *pageNumber* (int)
- *hasImage* (boolean)

1. Create the classes Book and Page.
2. Add the mentioned properties to the Book and Page classes. Remember to follow Encapsulation. (properties defined private)
3. Defined a getter and setter method for each property of the class book except for *array of pages*.
4. Define a getter and setter method for each property of the class *Page*.
5. Add a constructor for the Book and Page class.
6. Write a class called *SearchBook* with an instance method called *search*. This method is given an object from type Book and return the number of pages that have images and their page number is an even number.
7. Write an ApplicationDrive class to the *search* method you defined above properly.
8. What is the time and space complexity of the *search* method? Explain why?

Remember you need to use the *Iterable/Iterator* design pattern for this problem

Problem 2: Iterator/Iterable + Generic Type

Solve the problem 1 with this differences:

- Create a class called *GenericSearch* with an instance method called *search*. The search method will receive an *iterable* generic object instead of a book object and a predicate (condition) and return number of elements in the *iterable* object which holds the condition.
- Test this generic function using the Book and Page classes you defined in problem one.

Remember in addition to the *Iterable/Iterator* design pattern you used for Problem 1, you need to use Generic types to solve this problem.

Problem 3: Iterator/Iterable + Generic Type

Solve the problem 2 with the following differences:

- Create a class called *GenericSearchList* with an instance method called *search*. The search method will receive an *iterable* generic object (similar to problem 2) and a predicate (condition) and return a list of all elements in the list of all elements in the list, which hold the condition.
- Test this generic function using the Book and Page classes you defined in problem one.

Problem 4: Observer/Observable

Define a class called *Building*. The building has the following properties:

- Name (String)
- yearBuild (integer)
- A collection (list) of rooms.

The Room class has the following properties:

- size (integer)
- locked (boolean)

Define a class called *Manager*. The job of the manager is to call the police if a room is unlocked by an intruder. (You don't need to implement a police class).

Define a class called *Intruder*. The intruder has a static method called *unauthorizedAccess* which has two inputs: 1- A *Building* object and a room number. The *Building* class has two methods called *open* and *close*. Both method will receive a room number and the open method unlocks a room if it is locked and the close method will lock the room if it is unlocked.

1. Create the classes Building and Room and Manager and Intruder.
2. Add the mentioned properties to the classes. Remember to follow Encapsulation. (The properties are defined private)
3. Defined a getter and setter method for each property of the class *Building*.
4. Define a getter and setter method for each property of the class *Room*.
5. Define a proper constructor for the classes if needed.
6. At the beginning all rooms of the building is locked.
7. Write an *ApplicationDriver* class with a main method. Create a sample building with a room and manager and intruder.
8. Use Scanner to prompt the user to enter a room number and intruder will lock that room if it is unlocked. Repeat this until all rooms are unlocked and then terminate the application with a proper message like: "All rooms are unlocked. Not safe anymore".

9. Once a room is unlocked a message should be shown on the consule saying that the “The room x is locked” or “The room x is unlocked”.

Remember you should use Observer/Observable pattern to solve this problem

Problem 5: Observer/Observable + Iterator/Iterable

Think about three following classes:

- TemperatureSensor
    - Has one property: *temperature* (int)
- HumiditySensor
    - Has one property: *humidity* (int)
- PressureSensor
    - Has one property: *pressure* (int)

Think about another class called *WeatherStation*. This class’s job is to monitor any change in the value of *temperature*, *humidity* and *pressure*. In other word, if any change happens in the value of temperature, humidity and pressure, the class *WeatherStation* is notified.

The class *WeatherStation* keeps the average of temperature, humidity and pressure and updates the average when a new value is received.
Also when the a change happens in the average values (averageTemprature, averageHumidity and averagePressure), a news station is notified automatically and the news station will report the new average value. (“reporting” here means, printing the average value on the consule). Implement a proper class for news station.

There is a *Database* of *DataPoint*. The Database is updated by some satellites. (You don’t need to implement Satellite or the update mechanism). The *Database* contains some *DataPoints*.

The class *DataPoint* has two properties:

- *dataCategory* : Enum (*Temperature*, *Humidity*, *Pressure*)
- *newValue*: int

Create a proper constructor for this class and remember to apply the encapsulation principle for the properties.

1. Use the Observer/Observable pattern to connect the Sensors to WeatherStation and WeatherStation to NewsStation.
2. Use the Iterator/Iterable pattern to connect *Database* and *DataPoint* classes.

Now create the ApplicationDriver class.

- Create an object from *TemperatureSensor* and *HumiditySensor* and *PressureSensor* class.

Inside the ApplicationDriver class, create 10 instances of datapoints with arbitrary values. Then iterate over all datapoints and update the value of the corresponding object. For instance if the category of the datapoint is *Temprature* and its new value is 10, then you need to set these values in the instance of the TemporatureSensor you have already created. Repeat this for all datapoints in the database.

Remember when a value changes in any instances of *TemperatureSensor or HumiditySensor or PressureSensor, they notify the weatherStation instance and the weatherStation updates the average values and notifies the news station. Therefore, every time, a datapoint is applied, your program needs to print in console the new average values of temperature, pressure and humidity.*

<u>Problem 6: Observer/Observable + Generic Type</u>

Write a generic method called notify which will notify if size of a list of generic objects changes (an element is removed or being added) to the list.