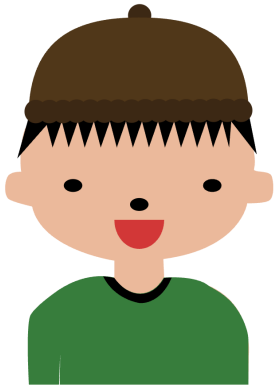
Observer Pattern



Scenario

A system is composed of:

- a GradeBook object stores a list of student's subjects and grades (GradeData objects).
- an email notification system that should automatically email a student when grade data is added or updated.
- a dashboard which is automatically updated when grades are added or updated.

The second 2 items on the above list should be updated automatically when the GradeBook changes.

How can we implement this? One option is to use the Observer design pattern.

Task 1

If you haven't already watched the video and completed the practical activity, it is recommended that you do so. Having a practical lab in front of you will help you to answer your own questions as you wrestle with the concept eg "what would happen if I removed this code?" or "what would happen if I changed this to that?"

Task 2

Answer the following questions.

1. In this scenario, what is being "observed"?

2. What datatype was added to the GradeBook class to store the observers?

3. It is not desirable to construct generic Observer objects. We only want to construct subclasses that are Observers. What kind of class should `Observer` be?

4. The update method in the Observer class has not been implemented. It is an abstract method. What implications does this have in the program design?

5. How did the EmailNotification class implement the update method?

6. How did the StudentDashboard class implement the update method?

7. Back in the GradeBook class, a notifyObservers method was designed. Explain how this method works.

8. How confident are you that for another scenario, you would be able to use the Observer pattern?

    - Key points in this scenario:
    - Identify what is being observed. In this case it was a GradeBook object.
    - In the design of the Gradebook class, Observers were also present (in this case stored in an ArrayList but could easily be an array or other data structure.
    - Examples of Observers were EmailNotification and StudentDashboard. These became subclasses of an abstract class, Observer.
    - Each subclass was forced to implement an update method.
    - In the GradeBook, every time a change happened, each Observer's update method was called.