Week 2 Synchronisation Session – Module 2 starting on Class

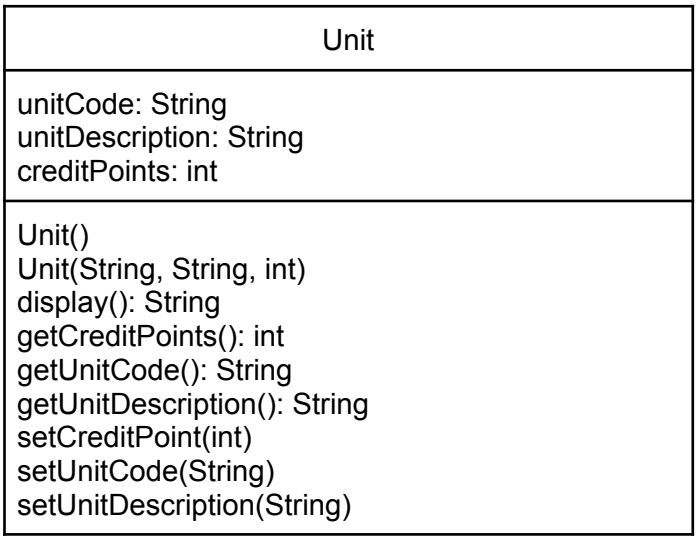
Expected runtime 60~75min

Pre-session checklist

- You have gone through Lesson 2.1-2.5 in Ed and in a process of going through the Rolling Project - Part 1 and Lesson 2.7

Rundown

1. Understanding class diagram: When working with object-oriented programming, programmers make use of a Class Diagram. A class diagram represents the various classes within a program along with the attributes (fields) and the behaviours (methods). Below is an example of a simplified class diagram.



2. Create the Java file for the above Unit class in the workspace

   Go to the workspace “ITO4131_XXXX” that is shared with you
   -> Use **terminal command "mkdir w2"** to create a folder called "w2"
   -> then use **terminal command "cd w2"** to change the working directory to "w2" (more details see appendix)
   -> create your Java file inside the "w2" folder and write your class for the above class diagram

3. OLA to demonstrate how to add a main method in the Unit class to run the display() method

4. OLA to revisit code challenge 3 in Lesson 2.6 (link https://edstem.org/au/courses/19006/lessons/59778/slides/406391) and revisit the basics of a class

5. Work on the following questions based on the code challenge 2 in Lesson 2.6 (link https://edstem.org/au/courses/19006/lessons/59778/slides/406390) and create your `Clock` Java file under "w2" folder in workspaces

   Go to the workspace "ITO4131_XXXX" that is shared with you; if you have NOT created folder w2, use **terminal command "mkdir w2"** to create a folder called "w2"

   a. Add an additional non-default Constructor
      i. `public Clock(int newHour)`: A constructor with only one parameter, it creates the Clock object by setting the hour field to the passed value.
   b. Change `setMinutes`() method:
      i. Add additional code in `setMinutes`() method to ensure that minute is within the range of 0-59 - hint you can use modulus operator .
   c. Change `main()` method:
      i. call the `setMinutes`() method for the `Clock` object and pass an argument of number 75, then display the clock again.
   d. What happens if we change the field "minute" as a public field? Is the constraint still effective?

6. Work on the Rolling Project - part 1 (link https://edstem.org/au/courses/19006/lessons/59772/slides/406350) by following the class diagram and develop the `Student` class and `Enrolment` class

   If you have not developed the code for the `Unit` class, at least create the `Unit` class file, and add fields and a default constructor in the file.Go to the workspace “RolllingProject_XXXX” that is shared with you

   -> Use **terminal command "cd w2"** to change the working directory to "w2"

   -> Use terminal commands “**touch Student.java**” and “**touch Enrolment.java**” to create your Java files inside the "w2" folder and write your class for the Rolling Project part 1 class diagram. Note: Please also download W2 folder to your own machine

Extra exercises for you to practice your programming skills. Use your workspaces to write the following code.

7. Write a Java class Author with following features. After finishing your code, **post your code** under the question “W2-S2 - task 6”
    a. Instance variables :
        i. `firstName` for the author’s first name of type String.
        ii. `lastName` for the author’s last name of type String.
    b. Constructor:
        i. `public Author()`: A default constructor, it creates the Author object by setting the values - ”unknown”.
        ii. `public Author(String firstName, String lastName)`: A constructor with parameters, it creates the Author object by setting the two fields to the passed values.
    c. Instance methods:
        i. `public void setFirstName(String firstName)`: Used to set the first name of the author.
        ii. `public void setLastName(String lastName)`: Used to set the last name of the author.
        iii. `public String getFirstName()`: This method returns the first name of the author.
        iv. `public String getLastName()`: This method returns the last name of the author.
        v. `public void display()`: This method printed out author’s first and last name to the screen

8. Write a Java class Book with following features. After finishing your code, **post your code** under the question “W2-S2 - task 7”
    a. Instance variables :
        i. `title` for the title of book of type `String`.
        ii. `author` for the author’s name of type `Author`.
        iii. `price` for the book price of type `double`.
    b. Constructor:
        i. `public Book(String title, Author name, double price)`: A constructor with parameters, its purpose is to create a Book object by initialising the fields to the passed values.
    c. Instance methods:
        i. `public void setTitle(String title)`: Used to set the title of a book.
        ii. `public void setAuthor(String firstName, String lastName)`: Used to set the author name of a book.
        iii. `public void setPrice(double price)`: Used to set the price of a book.
        iv. `public String getTitle()`: This method returns the title of the book.
        v. `public String getAuthor()`: This method returns the author’s name of the book.
        vi. `public void display()`: This method prints out book’s details to the screen

9. Writing a separate class `BookDemo` with a `main()` method. After finishing your code, **post your output** as a screenshot under the question “W2-S2 - task 8”

    Your `main()` method should

    a. Create a `Book` object titled “Developing Java Software” with the author being “Russel Winderand” and the price of 79.75
    b. Print the book’s string representation

10. To further build on top of the code challenge in Lesson 2.6 (link https://edstem.org/au/courses/19006/lessons/59778/slides/406391), add the following features in your `ClockDisplay` / `NumberDisplay` class.
    a. Constructor:
        i. Add the 3rd constructor with one `int` parameter: the value of time in terms of seconds since midnight (it should be converted into the time value in hours, minutes, and seconds)
    b. Instance methods:
        i. Add a new set method `setClock()` with one parameter seconds since midnight (to be converted into the time value in hours, minutes, and seconds as above)
            1. Hint - to get the hour from seconds, you can use the integer division like seconds / 3600. For example, if the number of seconds is 10000, then the hour is 10000 / 3600 = 2
            2. Hint - to get the minute from seconds, you can combine integer division and modulus like seconds / 60 % 60. For example, if the number of seconds is 10000, then the minute is 10000 / 60 % 60 = 46
            3. Hint - to get the remaining seconds, you can use modulus like seconds % 60. For example, if the number of seconds is 10000, then the second is 1000 % 60 = 40
        ii. Add an instance method `tickDown()` which decreases the seconds by 1 and then updates display
        iii. [Challenge task] Add an instance method `subtractClock()` that takes one `ClockDisplay` parameter and returns the difference between the time represented in the current `ClockDisplay` object and the one represented by the `ClockDisplay` parameter. Difference of time should be returned as a `ClockDisplay` object.

11. Write a separate class `ClockDemo` with a `main()` method. After finishing your code, **post your output** as a screenshot under the question “W2-S2 - task 10”. Your `main()` method should:

    a. Instantiate a `ClockDisplay` object `firstClock` using 10000 seconds since midnight.
    b. Tick the clock twice by applying its `timeTick()` method and print out the time after each tick.
    c. Instantiate a `ClockDisplay` object `secondClock` by using three integers (hours, minutes, seconds).
    d. Then tick the clock ten times, printing the time after each tick.
    e. Print both clock objects calling `getTime()` method
    f. [Challenge task] Create an object `thirdClock` that should reference the object of difference of `firstClock` and `secondClock` by calling the method `subtractClock()`.

Appendix

Class template

```
class Classname
{
        visibility_modifier data_type instance-variable1;
        visibility_modifier data_type instance-variable2;
        // ...
        visibility_modifier data_type instance-variableN;

        // Default Constructor must appear here
        visibility_modifier Classname()
        {
                // body of default constructor
        }

        // Non-Default Constructor(s) must appear here, if applicable
        visibility_modifier Classname(parameter-list)
        {
                // body of non-default constructor
        }

        // Other methods appear here.
        //      This should include the get methods (accessors)
        //      This should include the set methods (mutators)
        //      This should include the display() method
        //      This should include the main() method
        visibility_modifier data_type methodName1(parameter-list)
        {
                // body of method
        }
        visibility_modifier data_type methodName2(parameter-list)
        {
                // body of method
        }

        // ...
        visibility_modifier data_type methodNameN(parameter-list)
        {
                // body of method
        }
}
```