Practical 2 – Week 2
Classes, Objects & Methods

1. The following pseudocode describes how to turn a string containing a ten-digit phone number (such as "+6565501645") into a more readable string with spaces, like this: "(+65) 6550 1645".

   ```
   Take the substring consisting of the first three characters and surround it with
   "(" and ") ". This is the country code.
   Concatenate the country code, the substring consisting of the next four
   characters, a space, and the substring consisting of the last four characters.
   This is the formatted number.
   ```

   Translate this pseudocode into a Java program that reads a telephone number into a string variable, computes the formatted number, and prints it.

2. Java API has the `GregorianCalendar` class in the java.util package, which you can use to obtain the year, month, day of date. The default constructor creates an instance for the current date. The following instance methods return the year, month and day of the `GregorianCalendar` object respectively:
   `get(GregorianCalendar.YEAR), get(GregorianCalendar.MONTH),`
   `get(GregorianCalendar.DAY_OF_MONTH)`
   For the month value, January is 0, February is 1, and so on.

   Write a program, `CalendarApp.java`, to perform 2 tasks:
   - Display the current date in the format: DD-MM-YYYY
   - The `GregorianCalendar` class a constructor that takes in year, month day. Create a `GregorianCalendar` object with your birthdate.

3. Download the `Student.zip` from Blackboard. The zip contains the `Student.java` and `StudentApp.java.` Run the `StudentApp.java` and make sure that you understand the codes.

   a) Include another attribute, enrolment date, to the `Student` class. The enrolment date is of type `GregorianCalendar` and will be set to the current date when the student object is instantiated.

   b) Amend the `view` method to display the enrolment date. Run the `StudentApp.java` to test the `Student` class.

4. Write a `Customer` class that contains:
   - String data fields named `custId`, `name`.
   - A default constructor and a constructor that takes parameters to set the attributes.
   - Accessor and mutator methods.
   - A user defined method, `print()`, to display the customer id and name on the screen.

   Write a test program (`CustomerApp.java`) to test the `Customer` class.

   Sample output:

```
== Customer object instantiated using default constructor ==
Customer Id: null
Customer name: null

== Customer object created using non-default constructor ==
Customer Id: 1234567
Customer name: Ms Phoon
```

<u>Optional</u>

5. Given the class diagram, write the `Stock` class.

```
:Stock
------------------------------
symbol : String
name : String
openingPrice : double
currentPrice : double
------------------------------
//accessor & mutator methods
calPercentchange(): double
```

   The `calPercentChange()` returns the percentage changed from the opening price to the current price of the stock. Include in the class, a constructor that creates a stock with the specified symbol and name.

   Write a test program (`StockApp.java`) that creates a stock with the stock symbol ORCL, the name ORACLE CORPORATION, and the opening price of $34.50. Set the market price to $34.35 and display the price change percentage (to 3 decimal places).

   Sample output: `A drop of 0.435 percent`

-- End --

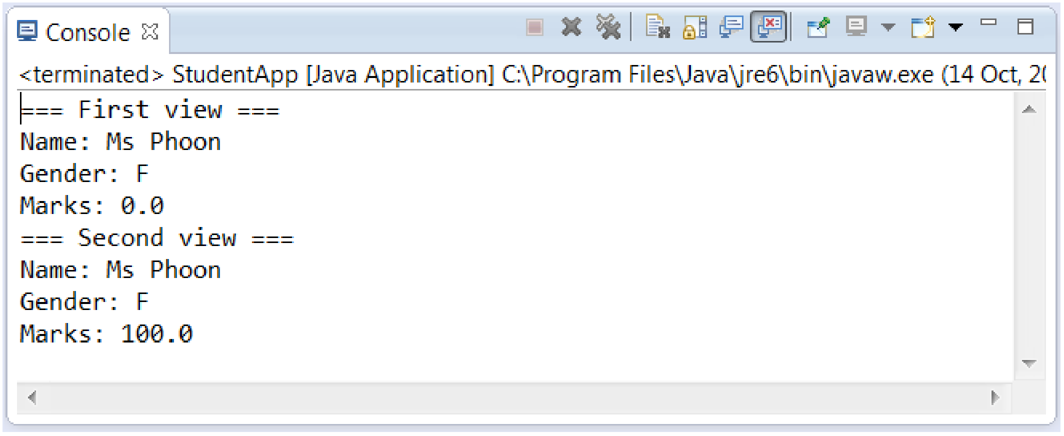
Practical 2 EP – Week 2

Debugging Exercise

Instructions

In order to complete this exercise, you must watch the video to learn the basic features of the Eclipse Debugger (http://vimeo.com/4110420). Then, download the Java project **Student.zip** (the same zip file for Practical 2 Question 3) from the Blackboard. Unzip/extract all the files to your local drive, and import the project to a workspace.

1. When you run the `StudentApp` program, the output is as shown:



```
<terminated> StudentApp [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (14 Oct, 2(
=== First view ===
Name: Ms Phoon
Gender: F
Marks: 0.0
=== Second view ===
Name: Ms Phoon
Gender: F
Marks: 100.0
```

2. Go to Window->Preferences->General->Editors->Text Editors to click the “Show line numbers checkbox.

3. Set breakpoints at: line #6 and line #9.

4. Run `StudentApp` in debug mode (Run->Debug As). A dialogue box should appear to prompt for confirmation to switch perspective. Click “Yes”. What do you notice?

5. Press F8. What does F8 do?

6. Click the red icon to terminate the program.

7. Run `StudentApp` in debug mode again. This time, you can try clicking the "bug" icon. Notice that the program stops at line #6.

8. Examine the variable, `args`, in the variables window and write down its content.

9. Press "Step Into" or F5. What happens? If you encounter `ClassNotFoundException`, don't panic. Press "Step Return" or F7 until the program returns to `StudentApp.`

10. Press F5. What do you notice?

11. Press F5 until the program execution returns to `StudentApp`. Notice that the program executes line by line.

12. Look at the variables window now. What do you notice?

13. Click on the variable, `s`, in the variables window and write down its value.

14. Expand `s` to check that the attributes have been updated correctly.

15. Press F5 twice. What happens?

16. Examine the variable in the variables window and write down its content.

17. Explain why the variables `args` and `s` disappear from the variables window.

18. Press F5 until the program execution returns to `StudentApp` and stops at line #9 or at `s.setMark(100)`.
19. Press F6. What happens?

20. What is the difference between "Step Into" (F5) and "Step Over" (F6)?

21. Expand `s` again. What do you notice?

22. Press F8 to complete the program execution.

23. Check the output in the console window.

24. Repeat the exercise a few times to get a better understanding of the debugger.

<u>Self-Check:</u>

- ☐ Name the 4 function keys in Eclipse Debugger and explain what they do?
- ☐ How would you rectify the problem of stepping into a Java API with no source code?

END OF EXERCISE