**Tutorial 1 (Week 2)**

The aims of this tutorial are

1. to ensure you can recall how to write, compile, debug, execute and test a Java program;
2. to revise how to create objects and call methods in them;
3. to revise the basics of a console user interface application;
4. to revise how to create classes to represent real world objects.

**Question 1**

Jimmy works part time in a company. On each working day the company requires him to do street survey. He is paid daily depending on how many surveys he has done. For every 5 surveys he has done, he is paid RM70.00, otherwise it is RM10.00 per survey. The table below shows some examples how he is paid:

|  |  |
| --- | --- |
| **Survey done** | **Paid amount, in RM** |
| 3 | 3 x 10.00 = 30.00 |
| 5 | 1 x 70.00 = 70.00 |
| 7 | 1 x 70.00 + 2 x 10.00 = 90.00 |
| 10 | 2 x 70.00 = 140.00 |
| 11 | 2 x 70.00 + 1 x 10.00 = 150.00 |
| 14 | 2 x 70.00 + 4 x 10.00 = 180.00 |
| 15 | 3 x 70.00 = 210.00 |

Write a Java program that can be used by the company’s account assistant to compute Jimmy’s daily pay. [**Hint**: %, /]

**Question 2**

Read the documentation for the TuteGroup class. This is available from the web page called TuteGroup.html. From the documentation, find out the purpose of a TuteGroup object, how to create one and the signatures (names and arguments) of any public methods.

**Part (A)**

When you are quite familiar with the TuteGroup class, in a new file called 'TestTG.java' write a Java application which creates a static TuteGroup object (in class scope) and then performs each of the following tasks:

* allow students to sign up their preferred tutorial group – with success and failed examples;
* print on the console screen the number of students who have signed up their tutorial group;
* print on the console screen the number of students who have signed up for a particular tutorial group;
* print on the console screen the names of all students who have already signed up their tutorial group;
* print on the console screen the names of student who have signed up for a particular tutorial group;
* search for a specific student given his/her name, and display his/her tutorial group if success, otherwise display an error message.

**Sample output:**

Tutorial grouping for ITC237

Total number of students signed up: 3

0 for Group 1, and 3 for Group 2

Student who have signed up:

Tan Chong

Amy

Kim Chong

Students signed up for Group 1

Jimmy

Enter name to search: Amy

Amy has signed up for Group 2

Student who have signed up:

Jimmy

Tan Chong

Amy

Kim Chong

**Alternative for searching:**

Enter name to search: Jane

Error. No student with name "Jane"

**Part (B)**

Write a Java application which provides a **menu** allowing the user to perform any of the tasks listed in (A) above. As each task is performed, the application menu should reappear.

The menu should have a 'Quit' function added to it so that the user can terminate the application.

**Question 3**

Define two Java classes:

Point, with the x and y coordinates, both of which are double values, as the fields.

Circle, with the centre, an instance of Point, and radius, a double value, of the circle as the fields.

Write two methods of the Circle class to do the following:

(a) Given a point p and a circle centered at c with radius r, determine whether p is **inside** the circle.

(b) Given the centre and radius of two circles, determines whether the two circles touch or overlap.

Write a main() method of the Circle class to test each of these methods.

The different diagrams shown below should give you some hints on how to find the conditions (or formulae) for determining whether a point is inside/outside a circle, and whether two circles touch/overlap each other or not.

