CS1020: Data Structures and Algorithms I

Tutorial 2 – Advanced OO Concepts

(For week 4, starting 3 February 2014)

1. **[Programming Model]**
   The following statements compare procedural programming with OOP. For each of them, indicate whether it is ***true*** or ***false*** and state your reason.
   a) It is easier to add or delete functions in OOP compared to procedural programming.
   b) If I want to write a simple code (e.g. less than 100 lines), using OOP will be less tedious and faster (performance wise).
   c) If I want to build a huge project with many functions and involving a lot of people, I should use procedural programming instead of OOP.
   d) If I want to disallow user to alter some key attributes in my program, I should use procedural programming instead of OOP.

2. **[Object-oriented Design Principles]**
   A `Student` class is used to keep track of students within a school. Each student may take a maximum of 7 modules in a semester.

   The code snippet below contains multiple violations of object-oriented design principles, as well as other design flaws.

```
class Student {
   public String studentName; // The student's name
   private String[] modules; // What a student is studying in the semester
   private int numOfModules = 0; // No. of modules a student is studying
   private int numOfStudents; // Track total no. of students in school
   private static double cumulativeGPA; // Track student's cumulative GPA

   public Student(String studentName) {
      studentName = studentName;
      cumulativeGPA = 0.0;
      numOfStudents++;
   }

   // Accessor for student name
   private String getStudentName(String studentName) {
      studentName = studentName;
      return studentName;
   }

   private static int getNumOfStudents() {
      return numOfStudents;
   }

   private static String[] getModules() {
      return modules;
   }

   // Method to add modules
   private boolean addModule(String newModule) {
      if (numOfModules < 7) {
         modules[numOfModules] = newModule;
```

```
            numOfModules++;
            return true;
        } else {
            System.out.println("Module workload exceeded!");
            return false;
        }
    }

    private int getNumOfModules() {
        return numOfModules;
    }

    private static double getCumulativeGPA() {
        return cumulativeGPA;
    }
}
```

Highlight all the violations. For each violation, suggest a modification to resolve it.

3. **[UML Design and modeling; Overriding]**
   A small school wants to develop an information system to keep track of all its assets (which are things such as TV projectors, overhead projectors, printers and desktop PCs) in its rooms. The school's estate department has specified the following:

   - There are only 3 rooms in the school, namely:
     1. LT 1
     2. LT 2
     3. Tutorial Room 1
   - Each room in the school can have one or more assets.
   - The estate department wants to keep track of the asset name, the type of item, as well as a numeric serial number for each asset. Each room should have a room name.
   - Room names will never be changed. The same goes for all information about assets.
   - It should be possible for new assets to be added to a room.
   - It should be possible to search for an asset within a room.

   To apply what you have learnt on overriding `equals()` and `toString()`, you have vowed to utilize method overriding as much as possible in your design. (Hint: refer to the sample interactions to decide what to do with `toString()`).

   a) Draw a UML diagram that models the school's requirements. Include suitable constructors and accessors/mutators to provide access control.
   b) Write the complete set of Java codes to satisfy the UML model.
   c) Design a client program to allow the estate department to use the system. Assume that all inputs to the client will be valid inputs.

   The following client stub has been provided (You are free to add new class/instance attributes, as well as to modify the main method as necessary):

```
import java.util.Scanner;

public class AssetClient {

  private static Scanner sc = null;

  public static void main(String[] args) {
    sc = new Scanner(System.in);
    int menuInput = 0;

    while (menuInput != 4) {
      System.out.println("Welcome to Asset Tracking Management System:\n");
      System.out.println("1. View all assets by room");
      System.out.println("2. Add asset to room");
      System.out.println("3. Locate asset");
      System.out.println("4. Exit");
      System.out.println("");
      System.out.print("Please select your choice: ");

      menuInput = sc.nextInt();
      sc.nextLine();

      System.out.println("");

      switch (menuInput) {
        case 1: viewAllAssetsByRoom(); break;
        case 2: addAssetToRoom(); break;
        case 3: checkAssetInRoom(); break;
        case 4: System.out.println("Thank you for using the Asset
Tracking Management System!");
      }

      System.out.println("");
    }
  }
}
```

Sample interactions are shown below (user inputs are underlined):

Main screen

```
Welcome to Asset Tracking Management System:

1. View all assets by room
2. Add asset to room
3. Locate asset
4. Exit

Please select your choice:
```

Viewing all assets by room

```
Welcome to Asset Tracking Management System:

1. View all assets by room
2. Add asset to room
3. Locate asset
4. Exit

Please select your choice: 1

Room: LT 1
- Printer: HP LaserJet 301A (Serial: 89909)
- Projector: Epson EX5210 (Serial: 89910)

Room: LT 2
- Printer: HP LaserJet 555B (Serial: 89911)
- Projector: Epson EX5210 (Serial: 89912)

Room: Tutorial Room 1
- Projector: Epson EX5210 (Serial: 89913)

Welcome to Asset Tracking Management System:

1. View all assets by room
2. Add asset to room
3. Locate asset
4. Exit

Please select your choice:
```

Adding a new asset to the room

```
Welcome to Asset Tracking Management System:

1. View all assets by room
2. Add asset to room
3. Locate asset
4. Exit

Please select your choice: 2

Available Rooms:
1. LT 1
2. LT 2
3. Tutorial Room 1

Please enter a room to add asset to: 1

Room LT 1 has been selected
Please enter asset name: HP LaserJet 301A
Please enter type of asset: Printer
Please enter asset number: 89909

A new Printer: HP LaserJet 301A (Serial: 8809) has been added
successfully!

Welcome to Asset Tracking Management System:

1. View all assets by room
2. Add asset to room
3. Locate asset
4. Exit

Please select your choice:
```

Checking to see if an asset is in the room (successful)

```
Welcome to Asset Tracking Management System:

1. View all assets by room
2. Add asset to room
3. Locate asset
4. Exit

Please select your choice: 3

Please enter asset name: HP LaserJet 301A
Please enter type of asset: Printer
Please enter asset number: 89909

Asset found in room: LT 1

Welcome to Asset Tracking Management System:

1. View all assets by room
2. Add asset to room
3. Locate asset
4. Exit

Please select your choice:
```

Checking to see if an asset is in the room (unsuccessful)

```
Welcome to Asset Tracking Management System:

1. View all assets by room
2. Add asset to room
3. Locate asset
4. Exit

Please select your choice: 3

Please enter asset name: ViewSonic PJD5123
Please enter type of asset: Projector
Please enter asset number: 89910

Asset was not found!

Welcome to Asset Tracking Management System:

1. View all assets by room
2. Add asset to room
3. Locate asset
4. Exit

Please select your choice:
```

Exiting the client

```
Welcome to Asset Tracking Management System:

1. View all assets by room
2. Add asset to room
3. Locate asset
4. Exit

Please select your choice: 4

Thank you for using the Asset Tracking Management System!
```

- End of Tutorial 2 -