**Centennial College**

**COMP 228: Java Programming**

**Lab Assignment 6: Developing Multithreaded Applications using Java Multithreading API and Collections API.**

**Student:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Due Date: Week 13

Purpose: The puarpose of this Lab assignment is to:

1. Practice multithreading in Java Applications
2. Practice Collections API in Java Applications
3. Develop a Java multithreaded application

References: Read the course’s text, ppt slides and class examples.

This material provides the necessary information you need to complete the exercises.

Instructions: Be sure to read the following general instructions carefully:

* **This is an in-class assignment**. You will have to finish the assignment and demonstrate your solution in **Week 13 scheduled lab session**.
* Submit the project through the **dropbox link on eCentennial**.
* You must name your Eclipse project according to the following rule:

**YourFullName\_COMP228Labnumber**

Example: **JohnSmith\_COMP228Lab6**

Each exercise should be placed in a separate package named *exercise1*, *exercise2*, etc.

Submit your assignment in a **zip file** that is named according to the following rule:

**YourLastName\_COMP228Labnumber.zip**

Example: **JohnSmith\_COMP228Lab6.zip**

**For a pair submission include both full names. Example: JohnSmith\_JaneSmith\_COMP228Lab6**

Apply the naming conventions for variables, methods, classes, and packages:

- *variable names* start with a *lowercase* character

- *classes* start with an *uppercase* character

- **packages** use only *lowercase* characters

- *methods* start with a *lowercase* character

### **Exercise 1:**

This exercise is similar to PrintTask example from Week 12.

Write a Java application that handles multiple ATM transactions (withdraw, deposit) at the same time. Create an **Account** class and implement both **deposit** and **withdraw** operations. Synchronize the operations to allow thread synchronization. Use Java Runnable interface to implement a **Transaction** class. Perform **withdraw** and deposit **operations** in **run** method.

Create an **AccountTest** class to test multiple transactions (threads). Use an ArrayList to create a list of three or more Transaction objects. Use method **execute** of ExecutorService to execute the threads. Display the results.

(10 marks)

**Evaluation:**

|  |  |
| --- | --- |
| **Functionality** |  |
| Correct implementation of Multithreading | 50% |
| Correct implementation of Collections API | 30% |
| Comments, correct naming of variables, methods, classes, etc. | 5% |
| **Friendly input/output** | 15% |
| **Total** | 100% |