CSC2044 Concurrent Programming

Exercise 1

Create three threads that can satisfy the criteria listed below by using Java.

- Name the three threads as “Thread A”, “Thread B”, and “Thread C” respectively.
- Each thread will print the name assigned to it to the console, when it is selected by the CPU scheduler to run.
- Ensure that “Thread B” can only start after “Thread A” and “Thread C” have completed.
- The main thread (the thread that runs the main() method) will print an appropriate message to the console to indicate that all the three threads have completed.

Exercise 2

You have opened two bank accounts (one main account and one sub account), each with initial saving of RM100. Your parents (father and mother) have decided that each will deposit RM1000 and RM500 into your main account and sub account respectively. But due to the transfer limit of the system, they are asked to split the amount into 10 transactions. Assuming that your parents will perform the transactions simultaneously, write a Java program to simulate the scenario. At the end, the program will print the total amount of saving in the main account and the sub account for verification purpose. Moreover, proper synchronization must be implemented to prevent race condition

Exercise 3

With the help of the class given in Appendix A, write a Java program to simulate the following scenario by using the await() method and the signal() or signalAll() method.

- Two chefs will be in the waiting state when there is no order.
- There is a waiter/waitress that will notify the chefs when there is an order, but only one chef is allowed to cook the dish.
- After sending an order to the chefs, the waiter/waitress will be in the waiting state when one of the chefs is cooking the dish.
- After finished cooking, the chef will notify the waiter/waitress to deliver the order.

Exercise 4

Implement a simple bulk purchase system for the RTX-series or RX-series graphics card that can satisfy the criteria listed below by using Java.

- The system consists of a buffer with 100 slots, and every single slot is enough to store all the units that a graphics card manufacturer (or the seller) would like to sell.
- A local distributor (or buyer) can only choose to buy all the units available in a slot. For example, if a slot consists of 50 units, the buyer can only choose to buy all the 50 units.
- There are 100 sellers. Each seller will randomly sell 1 to 100 units each round for 10 rounds. The sellers will also print to the console the number of units that they are trying to sell each round.
- There are 100 buyers, and each will buy all the units available in a slot for 10 rounds. At the end, each buyer should be able to know the total number of units bought from the system (but the buyer will not be printing this to the console).
- Proper synchronization must be implemented.

Appendix A

```
class Order {
        // To store the name of the dish that one of the chefs will be cooking.
          private String text;
        // To reflect whether there is any order that needs to be processed.
          private boolean orderStatus;

          public Order(String text, boolean orderStatus) {
                    this.text = text;
                    this.orderStatus = orderStatus;
          }

          public String getText() {
                    return text;
          }

          public void setText(String text) {
                    this.text = text;
          }

          public boolean getOrderStatus() {
                    return orderStatus;
          }

          public void setOrderStatus(boolean orderStatus) {
                    this.orderStatus = orderStatus;
          }
}
```