Problem Set 6

(10 Points) Cohort Exercise 1:

Fixed Buffer.java so that it is thread-safe and efficient

Solution: BufferFixed.java

(10 Points) Cohort Question 2:

Fix Experiment.java with volatile.

Solution: ExperimentFixed.java

(10 Points) Cohort Question 3:

Identify the pre-condition/post-condition of the methods in MyStack.java and make the class thread-safe.

Solution: MyStackThreadSafe.java

(10 Points) Cohort Question 4:

Assume a taxi tracking system which tracks taxis in Singapore. The updater threads would modify taxi locations and the view thread would fetch the locations of the taxis and display them. Examine the given Tracker.java class (shared by the updater threads and view thread) to determine if it is thread-safe and find a way to fix it if it is not thread-safe.

Solution: TrackerFixed.java

(10 Points) Cohort Question 5:

Continue cohort question 3. Examine the modified class DelegatingTracker.java and discuss whether it is thread-safe or not and fix it if it is not.

(10 Points) Cohort Question 6:

Modify Range.java so that it is thread-safe.

Solution: RangeSafe.java

(10 Points) Cohort Question 7:

Assume that multiple threads may call the static methods defined in FirstExample.java, fix FirstExample for potential problems.

Solution: FirstExample.java

(10 Points) Cohort Question 8:

Write a thread-safe class named SafeStack which extends java.util.Stack<E> with two operations pushIfNotFull(E e) and popIfNotEmpty(). Do it in different ways and design an experiment to compare the performance of the two.

Solution: SafeStack.java