COMP201P Written Java Concurrency Coursework 2017

Background

The computer science lifts are broken again! Also the department has finally fallen out with the contractors! They have asked you to write a new lift controller in Java!

They have provided you with 'lift simulator' code that can be used to test your controller – you can find the source code in the file "Lift Simulator.zip" on Moodle.

Programming Task

The complete system consists of four types of classes: People, Lift, Lift Controller and unit tests. Both the People and the Lift are seen as active objects, whereas the Lift Controller is a passive object. There are 9 floors to the building that are identified from 0 (Ground Floor) to 8 (Top Floor).

The simulator framework can be found on Moodle as a zip file, this *should not* be changed! You only need to implement the MyLiftController class *using traditional Java concurrency mechanisms (i.e. synchronized methods and wait/notify/notifyAll conditional synchronization to satisfy the specification given in the LiftController interface).*

Lift Class

The simulator has only a single lift that goes between floors. It starts at the ground floor and goes to the 8th floor, and then it goes back to the ground floor. While doing this it tells the lift controller what floor it is currently on and asks whether it should open its doors for people to enter/exit. The lift announces what floor it is currently on and whether the doors are opening or closing.

People Class

The simulator allows any number of people to be simulated. Each person randomly appears on a particular floor and chooses to go to another floor. They call the lift using the call lift button (either up or down direction). They have to wait for the lift to appear and open its doors. Once inside the lift they then select the floor that they wish. The people announce what floor they are on, where they wish to go to, and what they are doing in terms of pressing buttons.

Lift Controller Class

This is what you need to write. I**t has to satisfy the specification given internally in the code for the LiftController interface**. The MyLiftController class ***should not*** print anything to the output – all the printing is done via the People and Lift classes. **The framework classes should not be modified except for the MyLiftController class** (although you may want to change the number of people in the system at the top of the Main class to test out your controller for more people).

JUnit / MultiThreadedTC Unit Test Classes

In addition to the lift framework, the source directory also contains unit tests that employ the packages JUnit and MultiThreadedTC frameworks. This allows you to test your solution.

To get started ...

You should unzip the “Lift Simulator.zip” framework and get it working with your favourite Java IDE (if you want to use an IDE). Fairly detailed instructions are given for Eclipse below but these will need to be modified appropriately if you want to use a different IDE. Note particularly that the jar files within the lib directory need to be included in the project.

1. Unzip the “Lift Simulator.zip” file to create the “Lift Simulator” directory containing “lib” and “src” subdirectories.
2. The “src” directory contains the lift controller framework and also JUnit / MultiThreadedTC unit tests.
3. The “lib” directory contains MultithreadedTC-1.01.jar file and also junit-4.11.jar / hamcrest-core-1.3.jar files for the JUnit 4 test system.
4. Within Eclipse, do menu item “File > New > Java Project” to create a new Java project and call it “My Lift Controller”.
5. Right-click on the “src” directory of the newly created project and select the “Import” menu item. In terms of import sources, open up the “General” categories and double-click on the “File system” entry. You should get a form with “Into Folder: My Lift Controller/src” as the destination location. Browse the “From directory” location and choose the “src” directory from “Lift Simulator” directory. Tick the “src” directory box in the list and do “Finish”. (This should ensure that the ‘lift’ package is properly put under ‘src’ rather than things being put into a default package.
6. All the test scripts should have red boxes with crosses next to them since the MultithreadedTC-1.01.jar file has not been added to the project.
7. Right click on the Java project and select “Properties”. On the list to the left of the Properties dialogue, select “Java Build Path”. Under the “Libraries” tab, select “Add External JARs …” and add all three jar files in the “Lift Simulator /lib” subdirectory.
8. All the red crosses should disappear and your Eclipse project is ready for development.

Running the main.java file should result in a “trace” similar to the one given in Appendix A. Also you should now be able to right-click on MyLiftControllerTest and “Run As > JUnit Test”. All 10 tests should initially fail (as shown in Appendix B).

Read the code comments within the LIftController.java interface. You need to write Java code containing suitable concurrency mechanisms so that threads are controlled according to these concurrency rules.

Once you have written a successful lift controller that satisfy these rules, your code should pass all the JUnit tests as shown in Appendix C. (You may find this frustrating ... but you should try to get your code to properly satisfy the specifications of the methods to get them to pass these unit tests. The code also needs to be clearly thread-safe ...)

Once the MyLiftController is working, you should also find the framework produces a “trace” similar to the one given in Appendix D where the behaviour of the threads are being properly controlled. (Note the handling of the unusual boundary case where a Person goes from floor 0 to floor 0 ...)

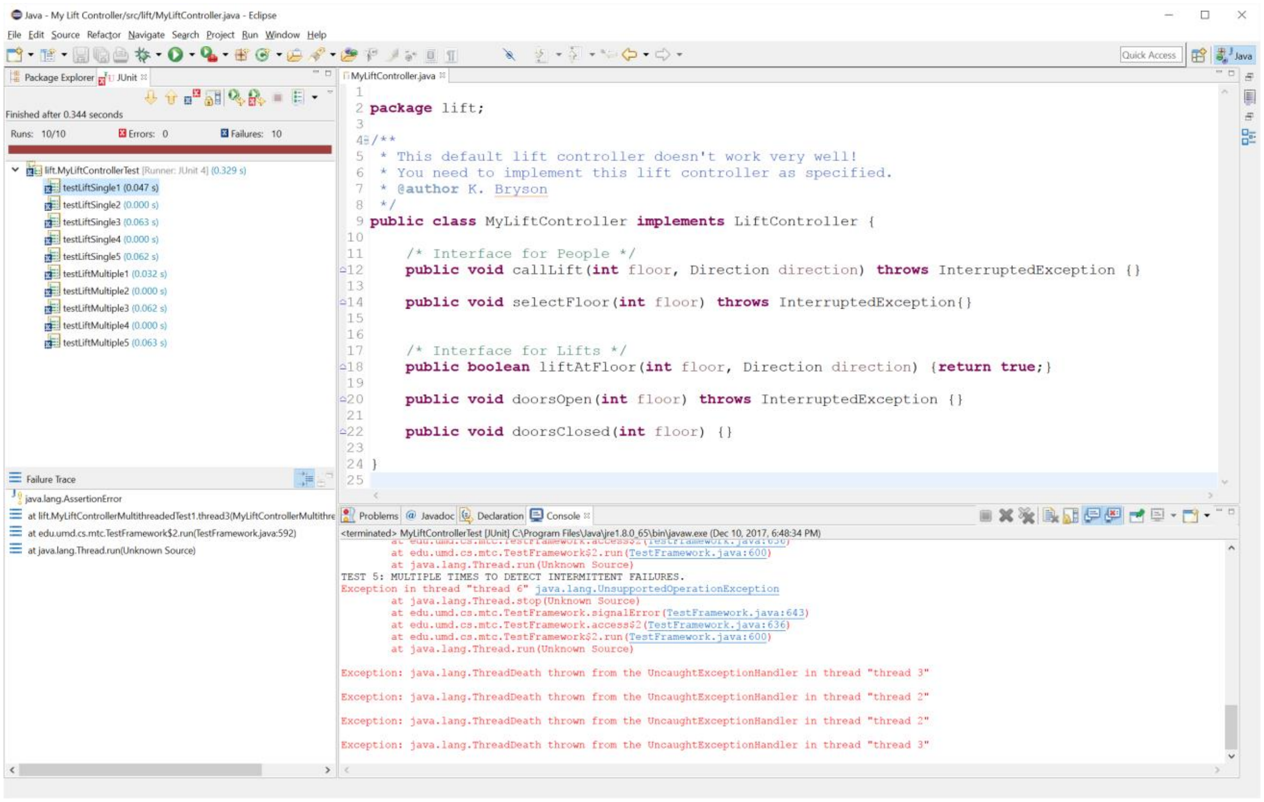
Please use the Moodle discussion board to clarify any questions about the requirements.

Good Luck!

Appendix A: Example Trace of Lift Simulator with the given broken MyLiftController (the threads are not controlled and havoc results!)

```
Main thread creating Lift and 2 people.
Main thread starting 2 people and Lift.
Application threads have all been started.
Main thread done its work - terminating.
Started Lift
Started Person 1
Started Person 2
Lift on floor 0, going UP
Lift has opened doors at floor 0
Lift closing doors at floor 0
Person 1 wants to go from floor 6 to floor 8
Person 1 selecting UP
Person 1 has entered lift.
Person 1 selecting floor 8
Person 2 wants to go from floor 7 to floor 0
Person 1 getting out of lift.
Person 2 selecting DOWN
Person 2 has entered lift.
Person 2 selecting floor 0
Person 2 getting out of lift.
Person 2 wants to go from floor 4 to floor 5
Person 1 wants to go from floor 6 to floor 4
Person 2 selecting UP
Person 2 has entered lift.
Person 2 selecting floor 5
Person 2 getting out of lift.
Person 1 selecting DOWN
Person 1 has entered lift.
Person 1 selecting floor 4
Person 1 getting out of lift.
Lift on floor 1, going UP
Lift has opened doors at floor 1
Lift closing doors at floor 1
Person 2 wants to go from floor 6 to floor 8
Person 2 selecting UP
Person 2 has entered lift.
Person 2 selecting floor 8
Person 2 getting out of lift.
Person 1 wants to go from floor 0 to floor 8
Person 1 selecting UP
Person 1 has entered lift.
Person 1 selecting floor 8
Person 1 getting out of lift.
Person 1 wants to go from floor 3 to floor 4
Person 1 selecting UP
Person 1 has entered lift.
Person 1 selecting floor 4
Person 2 wants to go from floor 3 to floor 4
Person 2 selecting UP
Person 1 getting out of lift.
Person 2 has entered lift.
Person 2 selecting floor 4
Person 2 getting out of lift.
... etc
```

Appendix B: Unit tests fail with unsupported operation exceptions.


Java - My Lift Controller/src/lift/MyLiftController.java - Eclipse

Finished after 0.344 seconds

Runs: 10/10   Errors: 0   Failures: 10

```
lift.MyLiftControllerTest [Runner: JUnit 4] (0.329 s)
    testLiftSingle1 (0.047 s)
    testLiftSingle2 (0.000 s)
    testLiftSingle3 (0.063 s)
    testLiftSingle4 (0.000 s)
    testLiftSingle5 (0.062 s)
    testLiftMultiple1 (0.032 s)
    testLiftMultiple2 (0.000 s)
    testLiftMultiple3 (0.062 s)
    testLiftMultiple4 (0.000 s)
    testLiftMultiple5 (0.063 s)
```

Failure Trace

```
java.lang.AssertionError
at lift.MyLiftControllerMultithreadedTest1.thread3(MyLiftControllerMultithre
at edu.umd.cs.mtc.TestFramework$2.run(TestFramework.java:592)
at java.lang.Thread.run(Unknown Source)
```

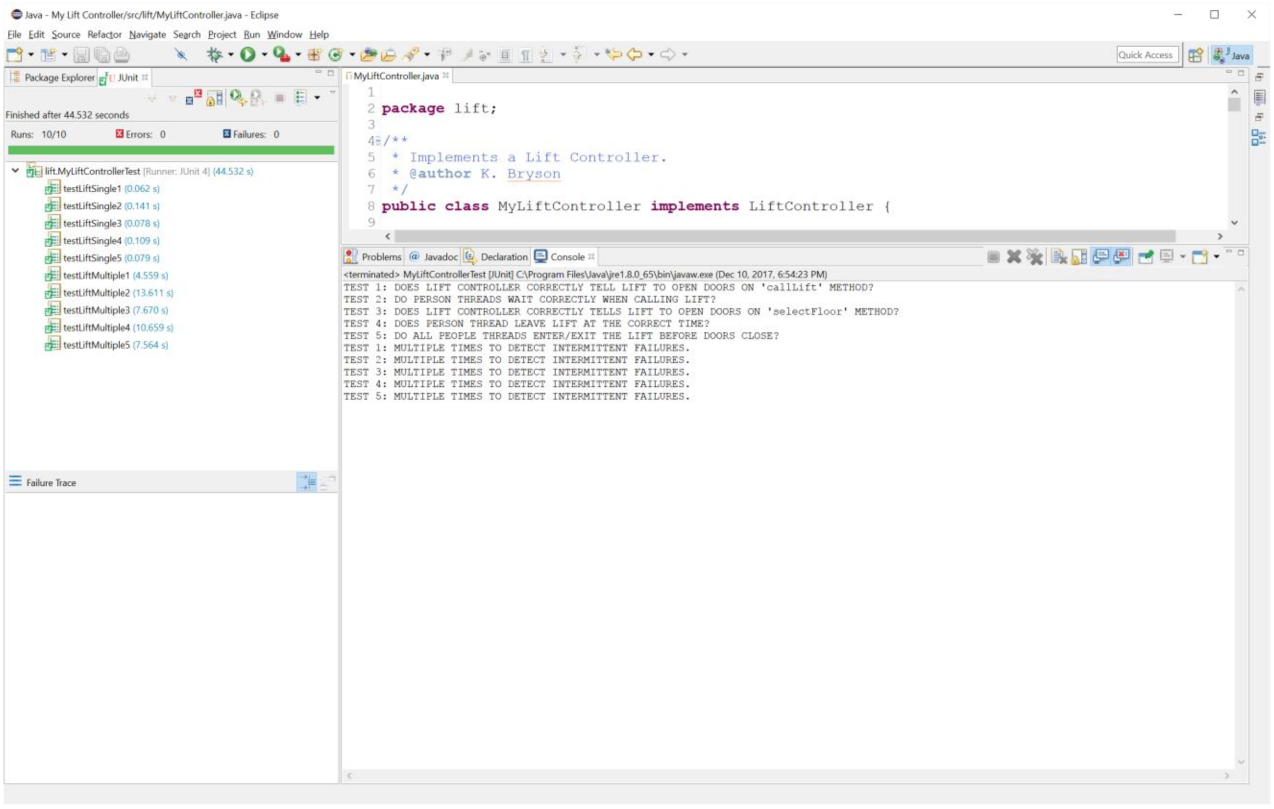
MyLiftController.java

```java

package lift;

/**
 * This default lift controller doesn't work very well!
 * You need to implement this lift controller as specified.
 * @author K. Bryson
 */
public class MyLiftController implements LiftController {

    /* Interface for People */
    public void callLift(int floor, Direction direction) throws InterruptedException {}

    public void selectFloor(int floor) throws InterruptedException{}


    /* Interface for Lifts */
    public boolean liftAtFloor(int floor, Direction direction) {return true;}

    public void doorsOpen(int floor) throws InterruptedException {}

    public void doorsClosed(int floor) {}

}

```

Problems Javadoc Declaration Console

```
<terminated> MyLiftControllerTest [JUnit] C:\Program Files\Java\jre1.8.0_65\bin\javaw.exe (Dec 10, 2017, 6:48:34 PM)
        at edu.umd.cs.mtc.TestFramework.access$2(TestFramework.java:636)
        at edu.umd.cs.mtc.TestFramework$2.run(TestFramework.java:600)
        at java.lang.Thread.run(Unknown Source)
TEST 5: MULTIPLE TIMES TO DETECT INTERMITTENT FAILURES.
Exception in thread "thread 6" java.lang.UnsupportedOperationException
        at java.lang.Thread.stop(Unknown Source)
        at edu.umd.cs.mtc.TestFramework.signalError(TestFramework.java:643)
        at edu.umd.cs.mtc.TestFramework.access$2(TestFramework.java:636)
        at edu.umd.cs.mtc.TestFramework$2.run(TestFramework.java:600)
        at java.lang.Thread.run(Unknown Source)

Exception: java.lang.ThreadDeath thrown from the UncaughtExceptionHandler in thread "thread 3"

Exception: java.lang.ThreadDeath thrown from the UncaughtExceptionHandler in thread "thread 2"

Exception: java.lang.ThreadDeath thrown from the UncaughtExceptionHandler in thread "thread 2"

Exception: java.lang.ThreadDeath thrown from the UncaughtExceptionHandler in thread "thread 3"
```

Appendix C: Unit tests pass with working MyLiftController.


Java - My Lift Controller/src/lift/MyLiftController.java - Eclipse

Finished after 44.532 seconds

Runs: 10/10   Errors: 0   Failures: 0

```
lift.MyLiftControllerTest [Runner: JUnit 4] (44.532 s)
    testLiftSingle1 (0.062 s)
    testLiftSingle2 (0.141 s)
    testLiftSingle3 (0.078 s)
    testLiftSingle4 (0.109 s)
    testLiftSingle5 (0.079 s)
    testLiftMultiple1 (4.559 s)
    testLiftMultiple2 (13.611 s)
    testLiftMultiple3 (7.670 s)
    testLiftMultiple4 (10.659 s)
    testLiftMultiple5 (7.564 s)
```

Failure Trace

MyLiftController.java

```java

package lift;

/**
 * Implements a Lift Controller.
 * @author K. Bryson
 */
public class MyLiftController implements LiftController {

```

Problems Javadoc Declaration Console

```
<terminated> MyLiftControllerTest [JUnit] C:\Program Files\Java\jre1.8.0_65\bin\javaw.exe (Dec 10, 2017, 6:54:23 PM)
TEST 1: DOES LIFT CONTROLLER CORRECTLY TELL LIFT TO OPEN DOORS ON 'callLift' METHOD?
TEST 2: DO PERSON THREADS WAIT CORRECTLY WHEN CALLING LIFT?
TEST 3: DOES LIFT CONTROLLER CORRECTLY TELLS LIFT TO OPEN DOORS ON 'selectFloor' METHOD?
TEST 4: DOES PERSON THREAD LEAVE LIFT AT THE CORRECT TIME?
TEST 5: DO ALL PEOPLE THREADS ENTER/EXIT THE LIFT BEFORE DOORS CLOSE?
TEST 1: MULTIPLE TIMES TO DETECT INTERMITTENT FAILURES.
TEST 2: MULTIPLE TIMES TO DETECT INTERMITTENT FAILURES.
TEST 3: MULTIPLE TIMES TO DETECT INTERMITTENT FAILURES.
TEST 4: MULTIPLE TIMES TO DETECT INTERMITTENT FAILURES.
TEST 5: MULTIPLE TIMES TO DETECT INTERMITTENT FAILURES.
```

Appendix D: Example Trace of Lift Simulator with working MyLiftController

```
Main thread creating Lift and 2 people.
Main thread starting 2 people and Lift.
Application threads have all been started.
Started Lift
Started Person 2
Started Person 1
Main thread done its work - terminating.
Lift on floor 0, going UP
Lift on floor 1, going UP
Person 1 wants to go from floor 8 to floor 5
Person 1 selecting DOWN
Person 2 wants to go from floor 2 to floor 1
Person 2 selecting DOWN
Lift on floor 2, going UP
Lift on floor 3, going UP
Lift on floor 4, going UP
Lift on floor 5, going UP
Lift on floor 6, going UP
Lift on floor 7, going UP
Lift on floor 8, going DOWN
Lift has opened doors at floor 8
Person 1 has entered lift.
Person 1 selecting floor 5
Lift closing doors at floor 8
Lift on floor 7, going DOWN
Lift on floor 6, going DOWN
Lift on floor 5, going DOWN
Lift has opened doors at floor 5
Person 1 getting out of lift.
Lift closing doors at floor 5
Person 1 wants to go from floor 0 to floor 0
Person 1 selecting UP
Lift on floor 4, going DOWN
Lift on floor 3, going DOWN
Lift on floor 2, going DOWN
Lift has opened doors at floor 2
Person 2 has entered lift.
Lift closing doors at floor 2
Person 2 selecting floor 1
Lift on floor 1, going DOWN
Lift has opened doors at floor 1
Person 2 getting out of lift.
Lift closing doors at floor 1
Person 2 wants to go from floor 6 to floor 7
Person 2 selecting UP
Lift on floor 0, going UP
Lift has opened doors at floor 0
Person 1 has entered lift.
Person 1 selecting floor 0
Person 1 getting out of lift.
Lift closing doors at floor 0
Person 1 wants to go from floor 2 to floor 8
Person 1 selecting UP
Lift on floor 1, going UP
Lift on floor 2, going UP
And so on ...
```