Assignment 1

Test run 1:	11:00pm, Wednesday, 16th of August 2017.
Test run 2:	11:00pm, Sunday, 20th of August 2017.
Test run 3:	11:00pm, Wednesday, 23rd of August 2017.
The due date:	11:00pm, Wednesday, 30th of August 2017.

This assignment is worth of 14% of the total course mark.

1 Objective

The purpose of this assignment is to make you familiar with the problems posed by event-driven systems. As a supplementary task you also need to implement a simple Graphical User Interface.

2 Specification

Exercise 1 *Simple Reaction Controller*

Imagine that you have been hired by an electronics company to build the software for a simple Reaction-Timer game. The reaction timer has two inputs: a coin-slot that starts the game and a go/stop button that controls it. There is also a display that indicates to the player what he/she should do next. The machine behaves as follows.

- Initially, the display shows ‘‘`Insert coin`’’, and the machine waits for a player to do so.
- When the player inserts a coin, the machine displays ‘‘`Press GO!`’’ and waits for the player to do so.
- When the player presses the go/stop button, the machine displays ‘‘`Wait...`’’ for a random time between 1.0 and 2.5 seconds. After the random delay expires, the machine displays a time-value that increments every 10 milliseconds, starting at zero. The player must now press the go/stop button as soon as possible – the goal of the game is to show fast reactions! If the player presses the go/stop button during the random delay period, i.e. the player tries to “guess” when the delay will expire, the machine aborts the game and immediately demands another coin. To put it simply, there is no reward for trying to cheat! If the user has not pressed stop after two seconds, the machine will stop automatically – no living person could be that slow!
- Whether the player has pressed the go/stop button within the two seconds waiting time period or not, the machine displays the final timer value for three seconds, then the game is over until another coin is inserted. If the player presses the go/stop button while the measured reaction time is being displayed, the machine immediately displays ‘‘`Insert coin`’’.

Exercise 2 *Enhanced Reaction Controller*

After showing the basic reaction timer machine to potential customers, the marketing department has found that some of them would like to pay more for a deluxe reaction timer that allows multiple games after payment of a single coin. In this machine, when a coin is inserted, the player is allowed to play three games. The operating sequence is very similar to the previous one: insert coin, press the go/stop button, wait random delay, press the go/stop button, then display the time for three seconds. If the machine has not yet completed three games, it then displays ‘‘`Wait...`’’. Subsequently, it waits a (new) random delay and proceeds further as in the first game. After displaying the time for the last game, the machine shows the average time ‘‘`Average= t.tt`’’ for five seconds, then the game is over. The enhanced reaction controller should fulfil the following requirements.

- If after inserting the coin the player fails to press go/stop within ten seconds, the game is over.
- If the player presses the go/stop button during the waiting period, the game is aborted, and the average value is not displayed. Once again, no reward for cheating!
- If the player presses the go/stop button while the machine is displaying a reaction-time value, the machine immediately moves on to the next game (or shows the average time) without waiting for the full three seconds of display time.
- If the player presses the go/stop button while the machine is displaying the average time, the game is immediately over.

3 Program Components

Your program must consist of the following parts.

- A controlling program named *ReactionMachine* is provided as a base to test your controllers.
- *SimpleReactionController* and *EnhancedReactionController* are the modules which you need to realize. The both components must conform to the interface *Controller*. This is the main body of the exercises. You are free to write these components in any way you choose, so long as they implement the required interfaces. You must write *SimpleReactionController.java* that functions like the simple reaction machine described above and *EnhancedReactionController.java* that behaves like the enhanced multi-game reaction machine.
- A *Display* is a GUI component which you need to write to test your reaction machine. It must conform to the *Gui* interface. The GUI must have a button labelled ‘‘`Coin inserted`’’ and a button labelled ‘‘`Go/Stop`’’. There must be a display region to show the messages of the machine. Times must be displayed with two decimal places. For example, a value of 1.5 seconds must be shown as ‘‘`1.50`’’. Do not go to too much effort to make your GUI “beautiful”. We are more concerned here with correct operation of your controllers rather than with your artistic ability! You must write *Display.java* that implements the necessary GUI functions.

Important. To ensure that we can automatically test your program, you must comply with the following constraints.

- Obviously, your program needs a source of timing information. You must obtain this information by implementing the *tick* method in your controller. The *ReactionMachine* class guarantees to call this method every 10 milliseconds. **Do not** use a Java timer, or your program cannot be tested and will fail all the tests.
- Your program also needs a source of random numbers. You must obtain your random numbers by calling the *getRandom()* method in the interface *Random*. **Do not** use the Java *Random* class, or your program will fail all the tests.

4 JAVA Files Provided

We provide the following java files for you.

- *Controller.java* contains an interface specification. Both of your reaction controllers must implement this interface.
- *Gui.java* contains an interface specification. Your GUI must implement this interface.
- *Random.java* contains an interface specification for the random number generator.
- *ReactionMachine.java* contains a simple driver program that will allow you to test your reaction controllers.

Please, be aware that you cannot alter the interface files, or we will not be able to test your program. Your display and two controllers must work correctly with *ReactionMachine*.

5 Running the Driver Program

You can run the driver program by typing: `java` *ReactionMachine c*, where *c* is the name of the controller you want to test. For example, `java` *Reactionmachine SimpleReactionController* will run the *SimpleReactionController* program. You are, of course, free to modify this program or build another one to help you with your testing.

6 Testing and Assessment

Since this is a level-three subject, we do not provide access to the full set of our tests before the first test run. The reduced suite of tests is always available for you, but performs only trivial tests to verify that nothing is catastrophically wrong with your program. We therefore encourage you to thoroughly test your code prior to the test runs and the submission deadline.

There will be three stages of testing. To participate in the first test run, your files must be in the automatic submission system by 11:00pm on Wednesday, August 16. The tests will be run sometime afterwards, and the results (plus a log of any exceptions that were generated) will be

posted shortly on the CANVAS forum. You will then have a few days to study the output and make corrections to your program.

The submission deadline for the second test run is 11:00pm on Sunday, August 20. Again, the results of the tests and any exceptions will be posted. The submission deadline for the third test run is 11:00pm on Wednesday, August 23. The final hand-in deadline is 11:00pm on Wednesday, August 30. Again, all the results will be posted.

Please note that if you fail to submit your work on time, you will miss the test run, and therefore will gain no information about your program. We encourage you to start early! The result you achieve on the final test-run will determine your mark for the exercise. No extra marking is expected on top of the marking done automatically by the web-submission system.

Submission instructions for programming code

First, type the following command, all on one line (replacing `aXXXXXXX` with your username):

```
svn mkdir --parents -m "EDC"
https://version-control.adelaide.edu.au/svn/aXXXXXXX/2017/s2/edc/assignment1
```

Then, check out this directory and add your files:

```
svn co https://version-control.adelaide.edu.au/svn/aXXXXXXX/2017/s2/edc/assignment1
cd assignment1
svn add SimpleReactionController.java
svn add EnhancedReactionController.java
svn add Display.java
svn commit -m "assignment1 solution"
```

Next, go to the web submission system at:

https://cs.adelaide.edu.au/services/websubmission/

Navigate to *2017*, *Semester 2, Adelaide*, *Event Driven Computing*, then *Assignment 1*. Click *Make a New Submission for This Assignment* and indicate that you agree to the declaration. The script will then check whether your code compiles. You can make as many resubmissions as you like.

End of Questions