The University of Melbourne
Department of Computing and Information Systems
COMP90041 Programming and Software Development
Semester 1, 2016
Project A
Due: 4pm Monday 11th April 2016

1 Background

This project is the first in a series of three, with the ultimate objective of designing and implementing (in Java) the game of Tic Tac Toe. It is a two player game, and the rules are as follows:

1. The game begins with an empty, 3×3 grid.
2. The two players then take turns placing a mark in an empty grid cell. Player O will use the ‘`O`’ (letter ‘O’, not zero) mark and Player X will use the ‘`X`’ mark.
3. The game is over in either case: (i) one player has 3 marks in a sequence (*vertically, horizontally, diagonally, or anti-diagonally*), or (ii) there are no empty cells left.

Below we show a run of the game, where Player O moved first and won the game by placing 3 ‘`O`’s in the second row of the game grid.

```
 | |      | |      | |      | |O     | |O     | |O
-----   -----   -----   -----   -----   -----
 | |     |O|     |O|     |O|     |O|     |O|
-----   -----   -----   -----   -----   -----
 | |      | |      | |X     | |X     |X|X    O|X|X
 (1)     (2)     (3)     (4)     (5)     (6)
```

2 Your Task

For this first project, your task is to write a class named `TicTacToe` that will enable two players to play a single game of Tic Tac Toe. To achieve this goal you will practise the use of instance variables and methods through the following stages.

2.1 Initialise the Game (2 Marks)

Your `TicTacToe` class should have a `main` method. This method will create an object of the `TicTacToe` class named `game` and call a method `run` to run a game:

```
TicTacToe game = new TicTacToe();
game.run();
```

Here the `main` method will only serve as a “testing method” for the `TicTacToe` class. Rather than creating it in a separate “`TicTacToeTest`” class, we put it in the `TicTacToe` class to simplify the project.

The game running process will be handled in the `run` method, which you need to implement. Note that you should make `run` an **instance method** of the `TicTacToe` class, **not a static method**. *The same* *applies for the methods described in the following sections.*

The `run` method will start with printing out a welcome message and prompting for the two players' names:

```
Welcome to Tic Tac Toe!

Enter Player O's name:
Rose
Enter Player X's name:
Jack
```

Note here `Rose` and `Jack` are user input, not part of the program output. You should add proper *instance (non-static)* variables to `TicTacToe` to store the two players' names, as well as accessors and mutators for them. *You may assume that there is no space character (' ') in the player names.*

2.2 Print the Game Grid and Make a Move (4 Marks)

Next, the game starts. The program will print out the game grid:

```
 | |
-----
 | |
-----
 | |
```

You need to add a `printGrid` method to the `TicTacToe` class for printing out the game grid. You should also add proper *instance* variables to keep track of what mark has been placed at each grid cell. For the 9 grid cells, you will need 9 instance variables (you can use an array to simplify this if you know how to use it properly, but you are not required to do so in this project). At start the grid is empty. You should initialise the instance variables properly to reflect this. Your `printGrid` method will access these instance variables to determine what to print at each grid cell.

The `run` method will then prompt for player's move. We assume that Player O always moves first.

```
Rose's move:
1 1
```

Here a player's move is represented by two integers in the range of $\{0, 1, 2\} \times \{0, 1, 2\}$, where "`0 0`" represents the top-left cell and "`2 2`" represents the bottom-right cell. You need to add a proper method to update the instance variables that store the grid cell status after a user has make a move.

2.3 Loop through the Game (4 Marks)

Now add a loop to repeatedly prompt for the players to take turns and make moves. The game grid should be printed out after each move. Here is an example execution after adding the loop:

```
 | |
-----
 | |
-----
 | |
Rose's move:
1 1
 | |
-----
 |O|
-----
 | |
Jack's move:
2 2
 | |
-----
```

```
 |O|
-----
 | |X
Rose's move:
1 2
 | |
-----
 |O|O
-----
 | |X
Jack's move:
2 1
 | |
-----
 |O|O
-----
 |X|X
Rose's move:
1 0
 | |
-----
O|O|O
-----
 |X|X
```

Add an *instance* method named `getGameState` to the `TicTackToe` class that returns the current state of the game after a player's move. There are four cases: (i) Player O has won; (ii) Player X has won; (iii) the game is a draw; (iv) the game can still continue. You should define proper constants to represent these different states. Note: we define "draw" to be a state when the game grid is full and neither player has won. You do not need to write a method that determines whether there is a draw before the game grid is full.

When the game is finished, print out a message showing the game winner's name (in the `run` method). For example, in the game playing process above, the player `Rose` has three '`O`'s in the second row and hence won the game:

```
Game over. Rose won!
```

If the game grid is full and no player has won, then print out that there is a draw:

```
Game over. It was a draw!
```

Please note that:

- *You may assume that all user input is valid, that is, you do not need to consider user input errors.*
- There is **no** blank line before the first line, i.e., no `println` before the welcome message is called.
- The last line should be **a full line**, i.e., "`Game over. Rose won!`" is printed out using `println`.
- You will be given a sample test input file `test0.txt` and the corresponding sample output file `test0-output.txt`. When you run your program by the following command in a terminal (or Windows command line):

  ```
  java TicTacToe < test0.txt > my-output.txt
  ```

  your program should produce a file name `my-output.txt` which should be exactly the same as `test0-output.txt`. In this command, "`< test0.txt`" and "`> my-output.txt`" are called "input redirection" and "output redirection." They use the content in `test0.txt` as the command line input, and print the program output into `my-output.txt`.

- Tests will be conducted on your program by automatically compiling, running, and comparing your outputs for several test cases with generated expected outputs. **These test cases used will be different from the sample test file given**. The automatic test will deem your output wrong if your output does not match the expected output, even if the difference is just having an **extra space**. Therefore, it is crucial that **you generate your own test files and test your program extensively**.
- The syntax `import` is available for you to use standard Java packages. However, please **DO NOT** use the `package` syntax to customize your source files. The automatic test system cannot deal with customized packages. If you are using Netbeans as the IDE, please be aware that the project name may automatically be used as the package name. Please remove any lines of the form

  ```
  package ProjA;
  ```

  at the beginning of the source files before you submit them to the system.
- Please use **ONLY ONE** Scanner object throughout your program. Otherwise the automatic test will cause your program to generate exceptions and terminate. The reason is that in the automatic test, multiple lines of test inputs are sent all together to the program. As the program receives the inputs, it will pass them all to the currently active Scanner object, leaving any remaining Scanner objects with nothing to read, causing a run-time exception. Therefore it is crucial that **your program has only one Scanner object.** Arguments such as "It runs correctly when I do manual test, but fails under automatic test" will not be accepted.

3 Assessment

This project is worth 10% of the total marks for the subject.

Your Java program will be assessed based on correctness of the output as well as quality of code implementation. See LMS for a detailed marking scheme.

4 Submission

Your submission should have one Java source code file. You must name it `TicTacToe.java`, and upload it to the Engineering School student server. Then, you can submit your work using the following command:

```
submit 90041 A *.java
```

You should then verify your submission using the following command:

```
verify 90041 A > feedback.txt
```

This will store the verification information in the file `feedback.txt`, which you can then view using the following command:

```
more feedback.txt
```

You should issue the above commands from within the same directory as where the file is stored (to get there you may need to use the `cd` "Change Directory" command). Note that you can submit as many times as you like before the deadline.

How you edit, compile and run your Java program is up to you. You are free to use any editor or development environment. However, **you need to ensure that your program compiles and runs correctly on the Engineering School student server**, using **build 1.8.0** of Oracle's (as Sun Microsystems has been acquired by Oracle in 2010) Java Compiler and Runtime Environment, i.e., `javac` and `java` programs. Submit your program to the server a couple of days before the deadline to ensure that they work (you can still improve your program). **"I can't get my code to work on the student server but it worked on my Windows machine" is not an acceptable excuse for late submissions.**

The deadline for the project is **4pm Monday 11th April 2016**. The allowed time is more than enough for completing the project. Penalties will apply on late submissions at 2 marks per day. Late submissions after 4pm Thursday 14th April 2016 will NOT be accepted.

5 Individual Work

Note well that this project is part of your final assessment, so cheating is not acceptable. Any form of material exchange, whether written, electronic or any other medium is considered cheating, and so is the soliciting of help from electronic newsgroups. Providing undue assistance is considered as serious as receiving it, and in the case of similarities that indicate exchange of more than basic ideas, formal disciplinary action will be taken for all involved parties.