159.352 Computer Work II:

Weight: 25%

This is a multi-part assessment with different intermediate deadlines.

Deadline: May 31st 2019 (for parts 1, 2, 3)

In this assignment you will create a Java/JavaScript implementation of Tic-Tac-Toe game following the specification below.

Part 1: Server [10 marks]

1. In NetBeans, create a Java Web Application **`assignment2_server_<your student id>`**
2. In this project, create a package **`nz.ac.massey.cs.webtech.s_<your student id>.server`**
3. Develop a game engine that uses the following http services. The game engine is to be implemented in one or multiple servlets providing the services described in the table below. Game state is to be maintained in a session. The user always plays with crosses, the server plays noughts.

URL (all URLs are relative w.r.t. root URL, such as localhost:8080	**Method**	**Meaning**	**Normal response**	**Error response**
`ttt/istart`	post	create a new game, user starts	a new game / session is created	
`ttt/ustart`	post	create a new game, computer starts		
`ttt/move/x3y1`	post	place a cross into the position encoded in the last part of the	On the server, this will trigger the computation of the next	400 Bad Request If the last token of the URL is malformed, or if

		URL, in the example: 3rd column (x=3) and 1st row	nought to be placed.	this position is not valid move on the current board 404 Not Found If there is not active game
`ttt/state?format=txt`	get	get the current board, the format parameter is either **txt** or **png**	the respective content type is text/plain and image/png, respectively, for txt encoding use this format: **ox_** **xox** **__x** Where the respective characters represent noughts, crosses and empty fields	
`ttt/won`	get		content type is text/plain, value is one line: “user” , “computer”,”draw” or “none”	404 Not Found If there is not active game

`ttt/possiblemoves`	get		the content type is text/plain, possible coordinates are encoded as one point per line (example: 1,1\n2,3\n)	

4. Game state is stored in a servlet session. This should also work if the user disables cookies in the browser.

Part 2 - Tests [4 marks]

In NetBeans, create a Java Application **`assignment2_test_<your student id>`**. In this project, write blackbox unit tests for the services implemented in part 1. Use standard junit tests without any particular web application testing frameworks for this purpose. You may use the apache http client library (highly recommended !). For each service described in the table above, there should be at least one test. If an error response is specified, a separate test should be added to test that the respective error response is generated. For the `ttt/state?format=png` service, only the content type needs to be tested.

Note: To run these tests, you will need absolute URLs for the respective services, in particular, there must be a fixed port. Try to use **localhost:8080**. If for some reason this is not possible, specify the root URL in a static final variable in each test class. Example:

```
public class ServiceTests {
        public static final String SERVER_URL = “http://localhost:8084”;
        ...
}
```

Part 3 - User Interface [7 marks]

1. In the server project **`assignment2_server_<your student id>`** , create a JSP page TTT.jsp that displays the game board and has functionality to create new games and makes moves.
2. TTT.jsp uses only the services specified in part 1 to play the game.
3. You have freedom to design the user interface, you can use either the textual (**state?format=txt**) or the image (**state?format=png**) service to display the board.
4. The client must ensure that only valid moves are sent to the server, you can use javascript and the **possiblemoves** server to validate input.

Part 4 - Weekly Tutorials [4 marks]

Tutorial exercises reinforce the concepts introduced in lectures, and provide a foundation for this assessment. Therefore it is important to make steady progress and submit the tutorials on time.

Tutorial exercise	Due
Week 7	April 15
Week 8	May 6
Week 9	May 13
Week 10	May 20

Notes

Interaction between client and server is based on the services specified in the table. I.e., in order to make a move, the web site would use the following services in this order:

1. ttt/move/..   -- to set a cross
2. If response is ok, use ttt/won to find out whether the game has finished, if so, display a message
3. ttt/state   to get an up-to-date game board to be displayed in the web site
4. ttt/possiblemoves to get information to be used to validate user input (in a text field, or using the png gameboard with an image map)

The computer move should be computed in the servlet mapped to ttt/move/ to ensure that the game state is updated before the response is sent back to the client.

To get started with part 1, you can use an http client like postman or curl.

Deliverables

Upload the zipped netbeans projects to stream. Both projects can be zipped into one file, please name this file **assignment2-<studentid>.zip**. It is highly recommended to test the zip (unzip into temporary folder, and try to open and run the projects in this folder with netbeans) before uploading.