

Assignment I1: Eclipse, MVC, Checkstyle and REST

✓ Published

 Edit





⋮

This is an individual assignment. This assignment will be graded as 0 to max points. Scroll down for submission instructions at the end.

In this assignment, you will develop a two-player tic-tac-toe game. We will provide the User Interface (UI) and the skeleton backend. You must complete the backend of the game to enable the players to play the game. You are supposed to complete this assignment in the Java programming language. This assignment will utilize widely used tools and libraries. Please follow the setup instructions to get started with the assignment.

1. Setup instructions

Please follow the instructions below to get started with your first assignment.

1. Download and install the latest Java JDK 11 from <https://www.oracle.com/java/technologies/javase-downloads.html>  (<https://www.oracle.com/java/technologies/javase-downloads.html>). (Instructions at <https://docs.oracle.com/en/java/javase/11/install/overview-jdk-installation.html#GUID-8677A77F-231A-40F7-98B9-1FD0B48C346A>  (<https://docs.oracle.com/en/java/javase/11/install/overview-jdk-installation.html#GUID-8677A77F-231A-40F7-98B9-1FD0B48C346A>)). If you already have an earlier JDK 11 installed, please update to the latest. If you have a different version of JDK installed (e.g., 8 or 14), and you need to continue using that version for some reason, contact [Shirish Singh](mailto:shirish@cs.columbia.edu) (<mailto:shirish@cs.columbia.edu>) for assistance.
2. Install 'Eclipse IDE for Java Developers' from <https://www.eclipse.org/downloads/packages/>  (<https://www.eclipse.org/downloads/packages/>) (Instructions at <https://www.eclipse.org/downloads/packages/installer>  (<https://www.eclipse.org/downloads/packages/installer>)). Eclipse requires JDK 8 or later, and won't fully install unless it finds JDK 8 or later already on your machine, so you really need to do #1 before #2 - and continue to follow these instructions in order.
3. Eclipse should already have Maven built-in. You can check this by running Eclipse, then choose **Window -> Preferences** and look for **Maven** on the list. If you cannot find Maven, contact [Shirish Singh](mailto:shirish@cs.columbia.edu) (<mailto:shirish@cs.columbia.edu>) for assistance.
4. Install the Checkstyle plugin from Eclipse Marketplace. Run Eclipse, then **Help -> Eclipse Marketplace**. Use the search box to find 'Checkstyle Plug-in', then click **Install**. Eclipse will prompt you to restart it after the update. The search may also return something called 'Checkstyle Contributions' and several other plug-ins that

do not need these now. You can check that Checkstyle h
Preferences and look for **Checkstyle** on the list.

5. Fork the github repository from <https://github.com/ProgrammiPublicAssignment.git> to your own github account. Then account to your local machine.

6. Import from your local clone repository to the Eclipse wo select **File -> Import -> Maven -> Existing Maven Proj** where you put the local repository. Select '[hw1/pom.xml [4156-PublicAssignment master]]' in the Package Explore Eclipse UI. Double click this to see folder substructure.

7. The setup is complete. Now, you can get started with the

We recommend using [Google Chrome](https://www.google.com/) for the game, and [Postman](https://www.postman.com/) for based on [Javalin](https://javalin.io/). Refer to Javalin doc the game. To convert models to JSON objects, you can use [gson](https://github.com/google/gson).

2. Tasks

We have built the user interface for the tic-tac-toe game. We have provided the skeleton code for the backend. The application will only support one game at a time, i.e., only two users can play.

You are required to complete the following programming tasks to finish the assignment:

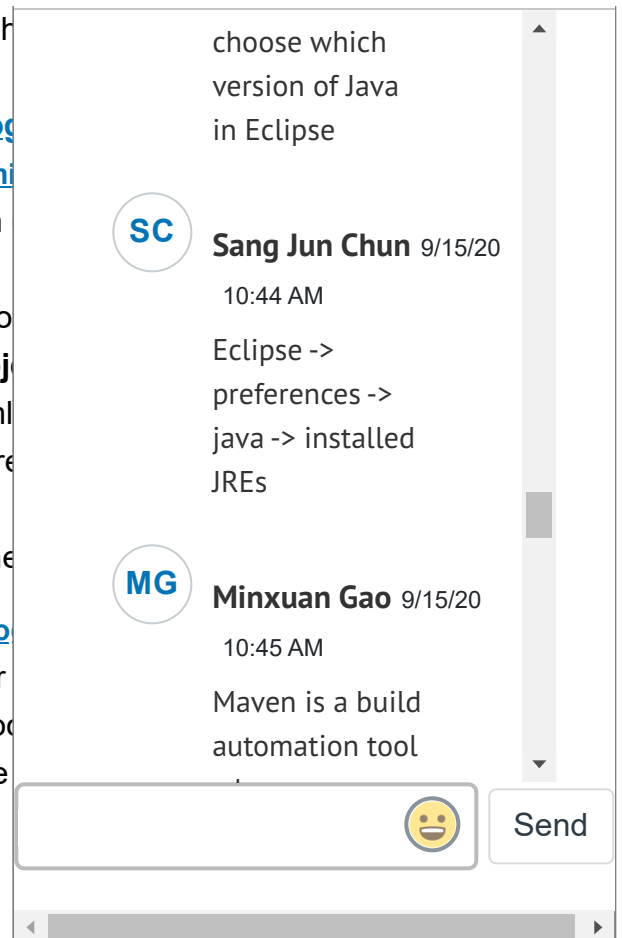
1. Following the MVC architecture, complete the models and controller (See section 2.1).
2. Perform style check using Checkstyle.

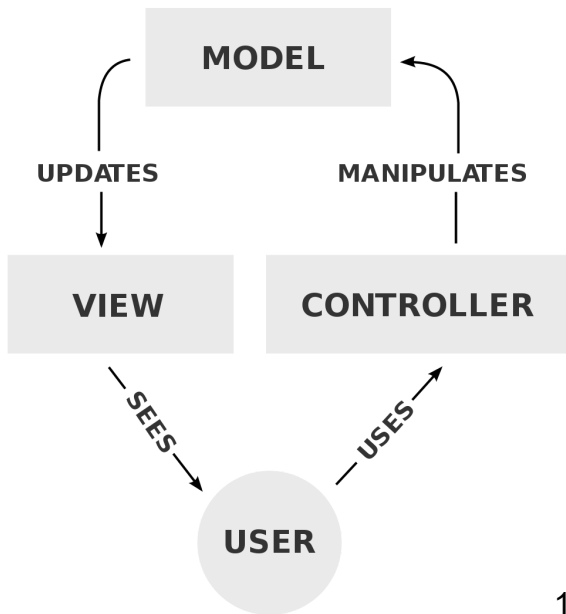
2.1 Model View Controller

<https://camo.githubusercontent.com/67df82db18d8fdc5d7af89cad0ebbf87d8d9243e/68747470733a2f2f757>

MVC (Model-View-Controller) is a popular architecture that is widely used for web and other GUI applications. Read this quick explanation of MVC: [Model-View-Controller \(MVC\) Explained Through Ordering Drinks At The Bar](https://www.freecodecamp.org/news/model-view-controller-mvc-explained-through-ordering-drinks-at-the-bar-efc6a6255053/).

Model: A model represent the entities of the application domain. It also constitutes the domain-specific logic related to the models. For example, in this game, the game board, and operations on the game board, will be the model.





View: A view is what is presented to the users of the application. It acts as the medium of interaction between the user and the application. In this game, the HTML web page (tictactoe.html) will act as a view.

Controller: A controller handles the incoming requests from the users and serves them based on certain rules, also known as the application logic.

2.2 Enabling Checkstyle

Follow the steps to enable the Checkstyle plugin for a particular project in Eclipse:

1. Right-click on the project.
2. Click on **Checkstyle** -> **Activate Checkstyle**
3. Open any file in Eclipse. You will be able to see the checkstyle warnings/errors in the IDE.

2.2.1 Generating Checkstyle Report

To generate a Checkstyle report, right click on pom.xml file, **Run as** -> **Build...** A pop-up window will appear. In the **Goals** add '**checkstyle:checkstyle**' and click **Run**.

Right-click on the project, and then click **Refresh**. Inside the **target** -> **site** directory, you will find *checkstyle.html*. Open the HTML file in a browser to read the report.

See Google's Style guide [here](https://google.github.io/styleguide/javaguide.html) ➡ [. \(https://google.github.io/styleguide/javaguide.html\)](https://google.github.io/styleguide/javaguide.html).

3. APIs

Most modern mobile/web apps have a RESTful service in the background controlling all the underlying business logic. A RESTful web service runs over the standard HTTP(s) web protocol like a web server.

3.1 RESTful Endpoints:

This assignment uses two standard HTTP methods. You can read more [here](https://www.w3schools.com/tags/ref_httpmethods.asp) ➡ [. \(https://www.w3schools.com/tags/ref_httpmethods.asp\)](https://www.w3schools.com/tags/ref_httpmethods.asp).

Methods	Description
GET	GET is used to request data from a specified resource.
POST	POST is used to send data to a server to create/update a resource.

The list of all endpoints for the assignment with details can be found below:

3.1.1 New Game

This endpoint will be used to return the initial UI, which allows the user to start a new game.

Endpoint

```
GET /newgame
```

Sample Response

Returns a new page for Player 1. Redirect the user to tictactoe.html page.

```
Redirect to /tictactoe.html
```

3.1.2 Start Game

This endpoint will initialize the game board and setup Player 1. The first player will always start the game.

Endpoint

```
POST /startgame
```

Request Body

Request body contains key-value pair.

Key	Type	Value
type	string	Type of the character player 1 has chosen: 'X' or 'O'.

Example- When Player 1 selects 'X' and starts the game, the request body is:

```
type=X
```

Sample Response

Returns the game board object in JSON. Example: Player 1 had started the game and selected 'X'.

```
{
  "p1": {
    "type": "X",
    "id": 1
  },
  "gameStarted": false,
  "turn": 1,
  "boardState": [
    "\u0000",
```

```
        "\u0000",
        "\u0000"
    ],
    [
        "\u0000",
        "\u0000",
        "\u0000"
    ],
    [
        "\u0000",
        "\u0000",
        "\u0000"
    ]
],
"winner": 0,
"isDraw": false
}
```

3.1.3 Join Game

Player 2 uses this link to join the game. This endpoint will initialize player 2 and add the player to the existing game board. After player 2 has joined, the game must start, and all players' UI must be updated with the latest game board state (See section 3.2 for reference). The view will recognize the player based on the URL query string.

Endpoint

GET /joingame

Sample Response Redirects the user to tictactoe.html page with query string "p=2".

Redirect to /tictactoe.html?p=2

3.1.4 Move

Updates the gameboard if the move is valid. Otherwise, it returns an error message.

Endpoint

POST /move/{playerId}

Path Parameters

Path Parameters	Type	Value
playerId	string	Id of the player making a move.

Request Body

Request body contains key-value pair.

Key	Type	Value
x	integer	Board row number
y	integer	Board column number

Example- When a player selects the top left cell to make a move, the request body is:

```
x=0&y=0
```

Here x and y represent the coordinates of the gameboard. For your reference, please see the game board representation (x, y):

0, 0	0, 1	0, 2
1, 0	1, 1	1, 2
2, 0	2, 1	2, 2

Sample Response

The following response will be generated if the move is valid.

```
{
  "moveValidity": true,
  "code": 100,
  "message": ""
}
```

You are free to add more messages to represent different scenarios. When the move is invalid, the **message** part of the response will be shown to the players in the UI.

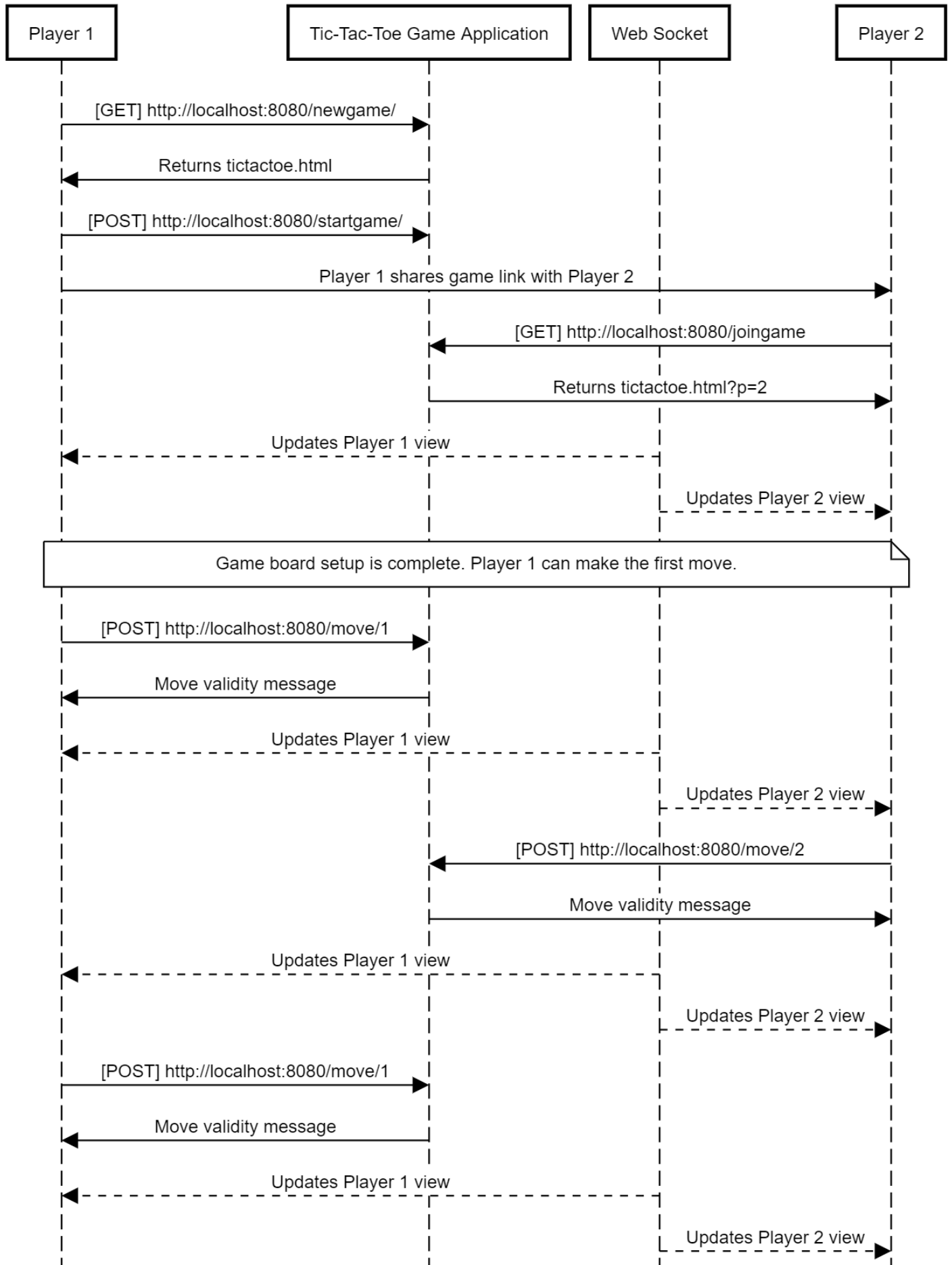
3.2 WebSocket:

In addition to sending the response message, the application must also update the game board of the players' user interface. We have provided a method "sendGameBoardToAllPlayers" which takes the JSON string of the game board object as input and updates the UI of both players. Use this method to update the UI.

4. Sequence Diagram

The following sequence diagram will help you understand the flow the application. Note that this diagram is merely for illustrating the basic idea and won't cover all the scenarios.


ASE Assignment 1



5. Grading Scheme

We will evaluate your submission on three criteria, as follows:


1. Test cases
2. Check Style compliance
3. Coding best-practices

 [_\(https://github.com/Programming-Systems-Lab/4156-PublicAssignmentBackground/blob/master/Part-1/README.md#51-test-cases\)](https://github.com/Programming-Systems-Lab/4156-PublicAssignmentBackground/blob/master/Part-1/README.md#51-test-cases)


5.1 Test Cases

We have five test cases to validate the tic-tac-toe game. Your game must follow these as described below:


1. A player cannot make a move until both players have joined the game.
2. Player 1 always makes the first move.
3. A player cannot make two moves in their turn.
4. A player should be able to win a game.
5. A game should be a draw if all the positions are exhausted and no one has won.

 [_\(https://github.com/Programming-Systems-Lab/4156-PublicAssignmentBackground/blob/master/Part-1/README.md#52-check-style-compliance\)](https://github.com/Programming-Systems-Lab/4156-PublicAssignmentBackground/blob/master/Part-1/README.md#52-check-style-compliance)

5.2 Check Style Compliance

In this assignment, we will follow [Google Java Style](https://google.github.io/styleguide/javaguide.html)  [_\(https://google.github.io/styleguide/javaguide.html\)](https://google.github.io/styleguide/javaguide.html). Please go through the document to understand the recommended style and make the necessary changes to your application's code.

You don't have to worry about style violations in the UiWebSocket file and anything related to "package-info.java"

 [_\(https://github.com/Programming-Systems-Lab/4156-PublicAssignmentBackground/blob/master/Part-1/README.md#53-coding-best-practices\)](https://github.com/Programming-Systems-Lab/4156-PublicAssignmentBackground/blob/master/Part-1/README.md#53-coding-best-practices)

5.3 Coding best-practices

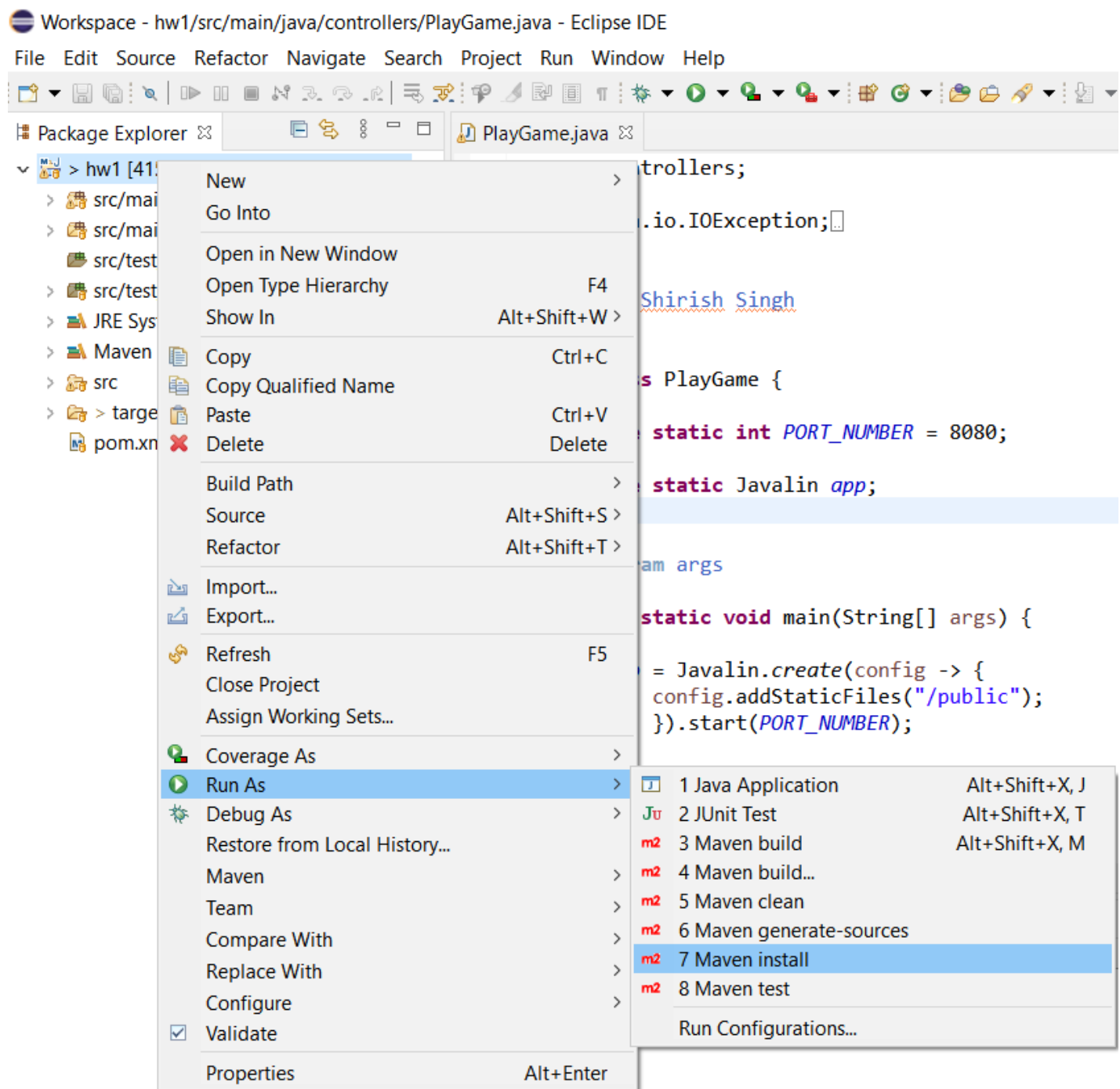
In a software engineering team, you will often work with one or more colleagues to deliver a feature/product. Your program must be clear and consistent so that others can effortlessly understand and use it. In this assignment, we require that you follow some basic best practices, such as:

1. Putting comments for methods describing their function,
2. Adding getters and setters for variables,
3. Giving meaningful names to variables and methods.

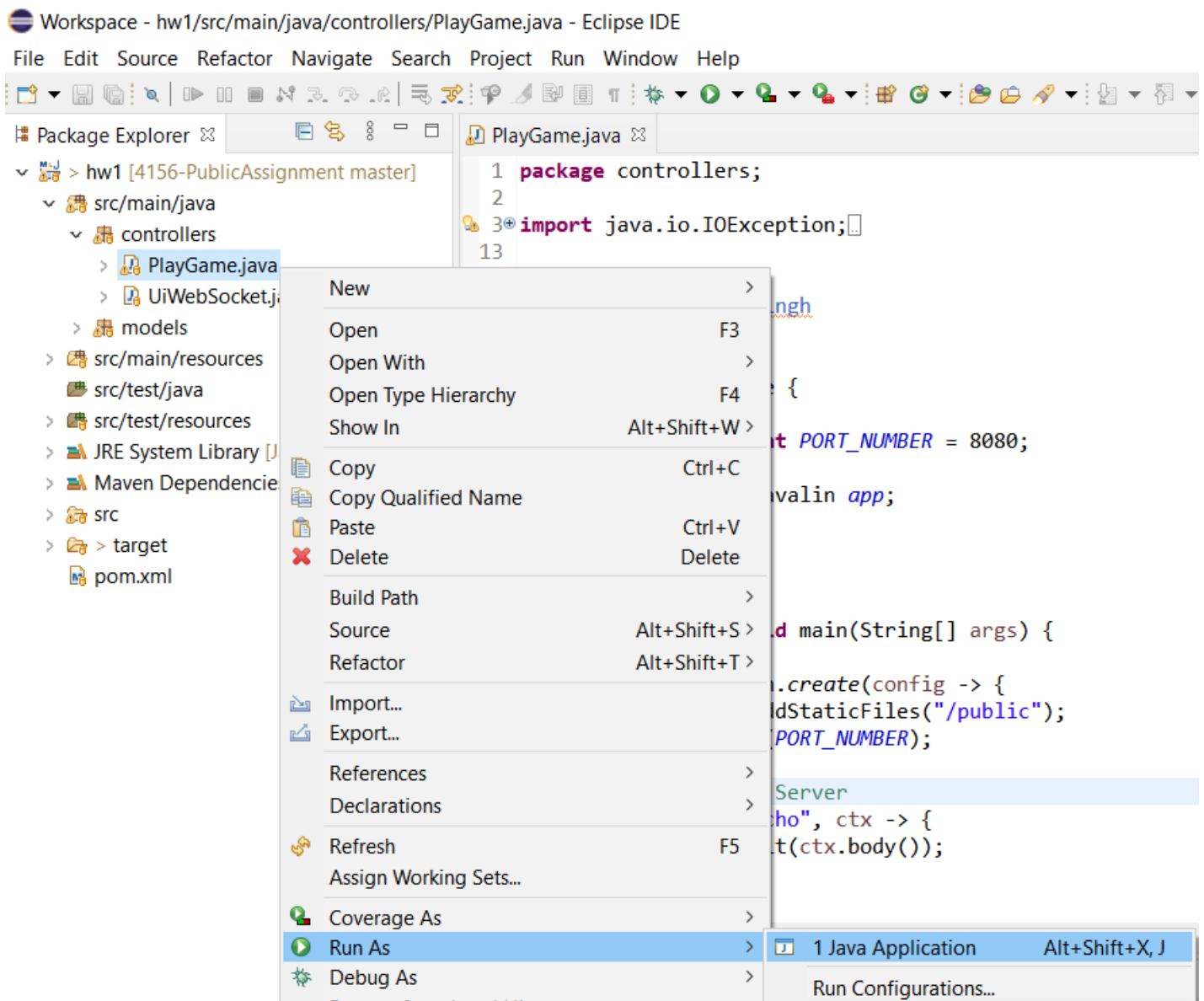
6. How to run the app?

You can follow the instructions below to build the application package and run the app:

1. Run 'maven install' to download all dependencies and package the project.



2. Now that all the dependencies have been downloaded, you can run the application by selecting and right clicking the 'PlayGame.java' file -> 'Run As' -> 'Java Application'



You can ask for help with any of the technologies involved in this assignment by posting questions or problems in this course's individual_project folder on [piazza](https://piazza.com/columbia.edu/spring16/courseworks2), (https://courseworks2.columbia.edu/courses/104335/external_tools/1456) on relevant community forums, on stackoverflow, etc. However, do not ask for help writing the code, this is a violation of the university's academic honesty policy.

Submission instructions: Submit a zip file containing hw1 directory. You may resubmit repeatedly until the deadline. If you were unable to complete this assignment for any reason, submit anyway. In that case you should include a text explaining how far you got and what went wrong.

Points 20

Submitting a file upload

File Types zip

Due	For	Available from	Until
Sep 24, 2020	Everyone	Sep 1, 2020 at 12:01am	Oct 1, 2020 at 11:59pm

Assignment 1 Rubric			
Criteria	Ratings		Pts
Passing all test cases We have created 5 test cases that tests the logic of the game, as defined by the rules. You will be getting two points for each passing test case.	10 to >0.0 pts Full Marks	0 pts No Marks	10 pts
CheckStyle Compliance 1 point deduction per 2 errors (other than indentation and tabs)	5 to >0.0 pts Full Marks	0 pts No Marks	5 pts
Coding standards and best practices 0.5 point deduction for missing comments [Total of 2 points] 0.5 point deduction for lack of getter/setters [Total of 2 points] 1 point deduction for non-meaningful variable and method names	5 to >0.0 pts Full Marks	0 pts No Marks	5 pts
			Total Points: 20