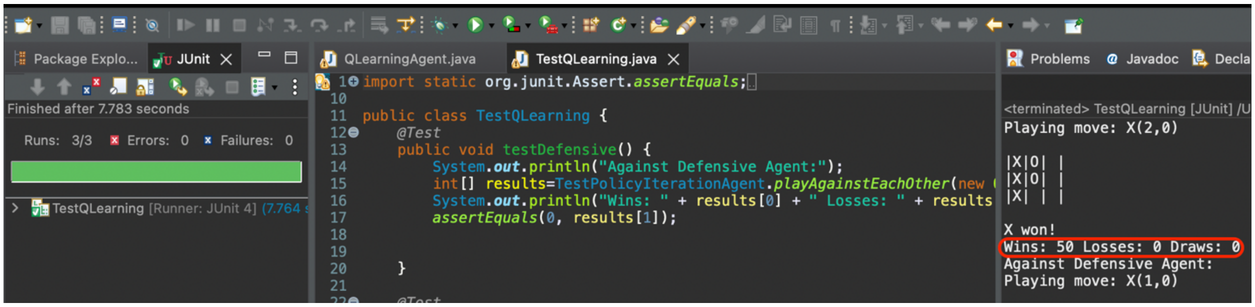
Hosam Farouk

Q-Learning Agent ***VS*** Random Agent:



Finished after 7.783 seconds

Runs: 3/3 Errors: 0 Failures: 0

TestQLearning [Runner: JUnit 4] (7.764 s

```
import static org.junit.Assert.assertEquals;

public class TestQLearning {
    @Test
    public void testDefensive() {
        System.out.println("Against Defensive Agent:");
        int[] results=TestPolicyIterationAgent.playAgainstEachOther(new (
        System.out.println("Wins: " + results[0] + " Losses: " + results
        assertEquals(0, results[1]);


    }

    @Test
```

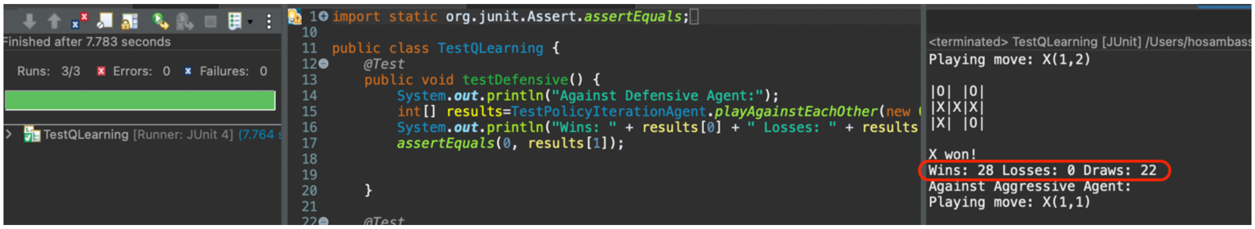
```
<terminated> TestQLearning [JUnit] /U
Playing move: X(2,0)

|X|O| |
|X|O| |
|X| | |

X won!
Wins: 50 Losses: 0 Draws: 0
Against Defensive Agent:
Playing move: X(1,0)
```

Wins: 50 | Losses: 0 | Draws: 0

Q-Learning Agent ***VS*** Defensive Agent:



Finished after 7.783 seconds

Runs: 3/3 Errors: 0 Failures: 0

TestQLearning [Runner: JUnit 4] (7.764 s

```
import static org.junit.Assert.assertEquals;

public class TestQLearning {
    @Test
    public void testDefensive() {
        System.out.println("Against Defensive Agent:");
        int[] results=TestPolicyIterationAgent.playAgainstEachOther(new (
        System.out.println("Wins: " + results[0] + " Losses: " + results
        assertEquals(0, results[1]);


    }

    @Test
```

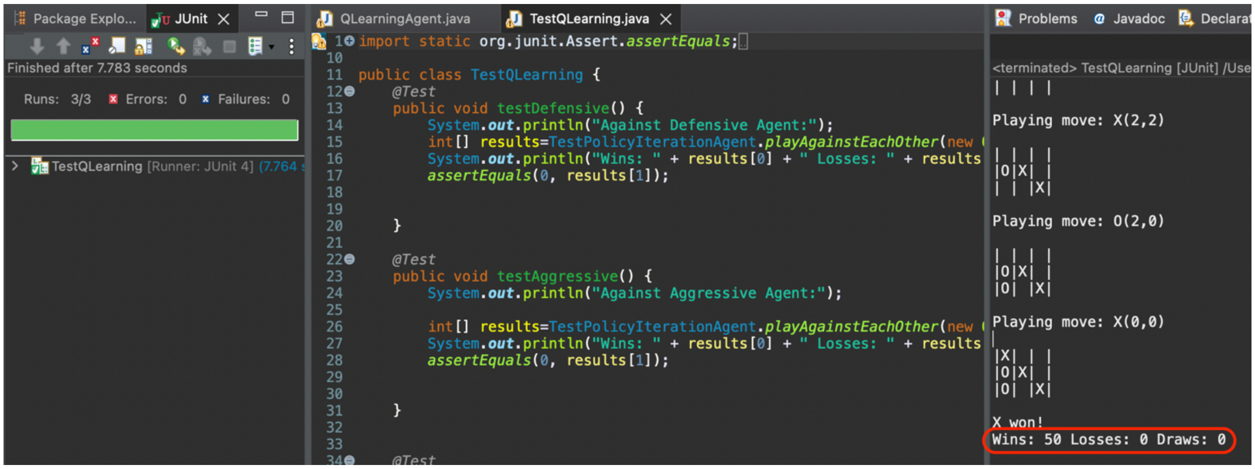
```
<terminated> TestQLearning [JUnit] /Users/hosambass
Playing move: X(1,2)

|O| |O|
|X|X|X|
|X| |O|

X won!
Wins: 28 Losses: 0 Draws: 22
Against Aggressive Agent:
Playing move: X(1,1)
```

Wins: 28 | Losses: 0 | Draws: 22

Q-Learning Agent ***VS*** Aggressive Agent:



Finished after 7.783 seconds

Runs: 3/3 Errors: 0 Failures: 0

TestQLearning [Runner: JUnit 4] (7.764 s

```
import static org.junit.Assert.assertEquals;

public class TestQLearning {
    @Test
    public void testDefensive() {
        System.out.println("Against Defensive Agent:");
        int[] results=TestPolicyIterationAgent.playAgainstEachOther(new (
        System.out.println("Wins: " + results[0] + " Losses: " + results
        assertEquals(0, results[1]);


    }

    @Test
    public void testAggressive() {
        System.out.println("Against Aggressive Agent:");

        int[] results=TestPolicyIterationAgent.playAgainstEachOther(new (
        System.out.println("Wins: " + results[0] + " Losses: " + results
        assertEquals(0, results[1]);


    }


    @Test
```

```
<terminated> TestQLearning [JUnit] /Use
| | | |

Playing move: X(2,2)

| | | |
|O|X| |
| | |X|

Playing move: O(2,0)

| | | |
|O|X| |
|O| |X|

Playing move: X(0,0)
|
|X| | |
|O|X| |
|O| |X|

X won!
Wins: 50 Losses: 0 Draws: 0
```

Wins: 50 | Losses: 0 | Draws: 0

Function Implementations in 2-3 sentences:

train():

- First initialised a variable episode to track how many episodes have been completed by the agent in the environment.
- Then loop for the number of iterations required and in the beginning of each iteration the environment is reset, and as long as the state is non-terminal it follows the epislon greedy strategy on whether to exploit or explore in the next step based on the num genereated by Math.random()
- After updating Q values based on the rewards and transitions, it extracts the optimal policy from the learned Q values

extractPolicy():

- First initialized a Policy Object.

- Then it extracts the optimal policy from the Q-table by iterating through all states and determining the move with the highest Q value for each non-terminal state using the Collections.max method. The extractedPolicy.policy.put.. maps states to their corresponding optimal moves.

getQValueForAllMoves():

- A helper method used to retrieve the Q values for all possible moves for a state which is saved in the variable moves using getPossibleMoves() method.

- It returns a HashMap where each move is a key, and its corresponding Q value is the value.

updateQValue():

- A helper method that updates the Q value for a state-move pair using the Q learning formula, incorporating the immediate reward and the discounted max arg Q value of the next state. The updated Q-value is then stored in the Q table.