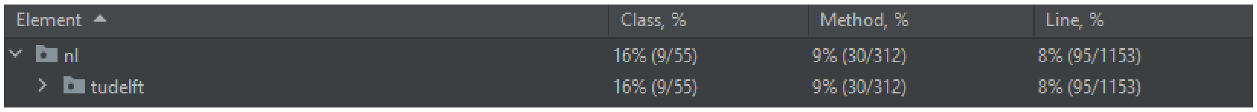
Github Link: https://github.com/zmorgancs/CS472
Task 2.1

Method	**Qualified Method Name**
createGround()	src/main/java/nl/tudelft/jpacman/board/boardFactory.createGround
isInProgress()	src/main/java/nl/tudelft/jpacman/game/Game.isInProgress
stop()	src/main/java/nl/tudelft/jpacman/game/Game.stop

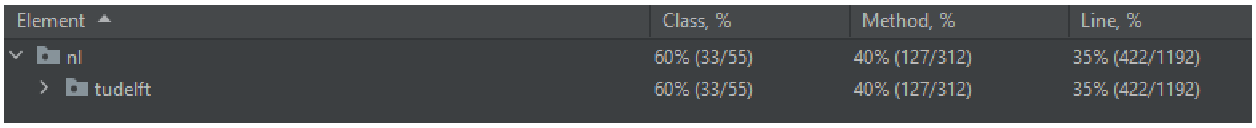
Coverage Report

Initial:



Element	Class, %	Method, %	Line, %
nl	16% (9/55)	9% (30/312)	8% (95/1153)
tudelft	16% (9/55)	9% (30/312)	8% (95/1153)

Final:



Element	Class, %	Method, %	Line, %
nl	60% (33/55)	40% (127/312)	35% (422/1192)
tudelft	60% (33/55)	40% (127/312)	35% (422/1192)

The majority of class coverage gains came from the isInProgress() unit test, followed by createGround(). stop() did not yield much in terms of class coverage, but did bump the method and line coverage up a minor amount.

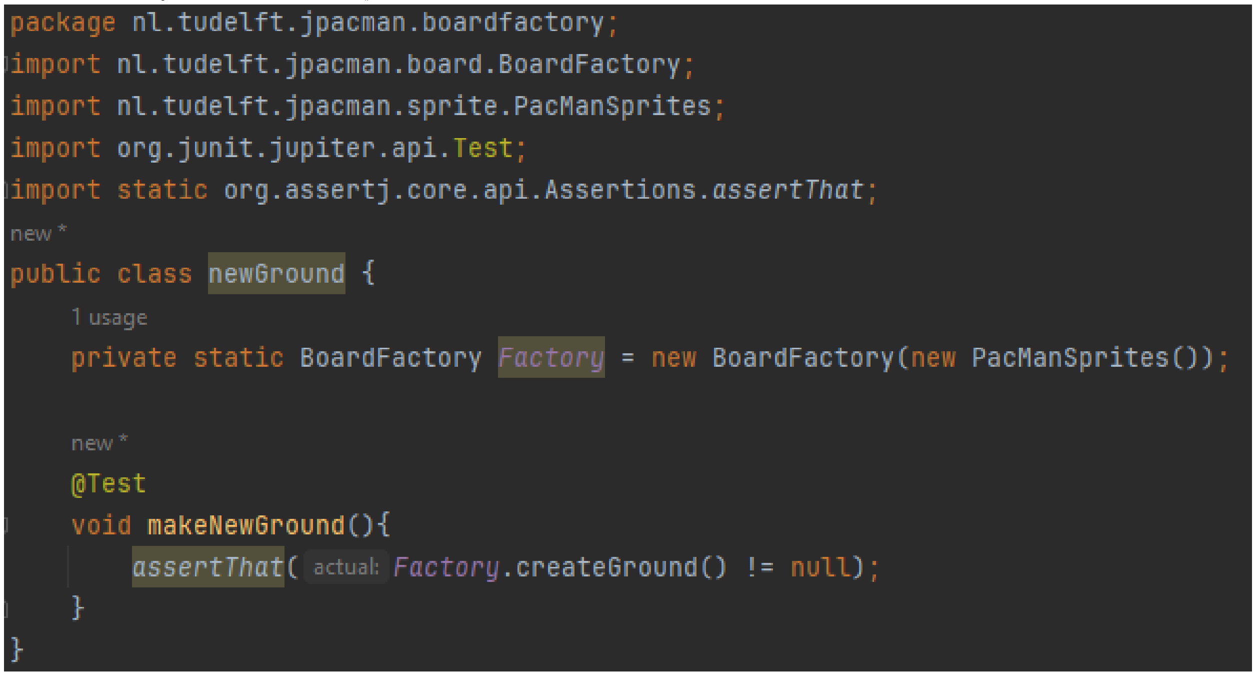
Task 3

- **Are the coverage results from JaCoCo similar to the ones you wrote from intelliJ in the last task? Why or why not?**
  I found that the results from both were quite similar, however certain coverage results were much different which makes me think that JaCoCo takes more consideration in how it determines the coverage score and profile, which seems to result in lower percentage scores in certain areas.
- **Did you find helpful the source code visualization from JaCoCo on uncovered branches?**
  I found them very helpful because the visuals clearly convey which branches and instructions were missed which allows me to determine whether or not I need to cover more ground with my test cases.
- **Which visualization did you prefer and why? IntelliJ's coverage window or JaCoCo's report?**
  I preferred navigating the intelliJ coverage report more so than the JaCoCo report due to the more enjoyable feature set that intelliJ provides. Intellij allowed me to access and edit

source files very quickly and the result window output makes working on test cases go much smoother than having to navigate JaCoCO.

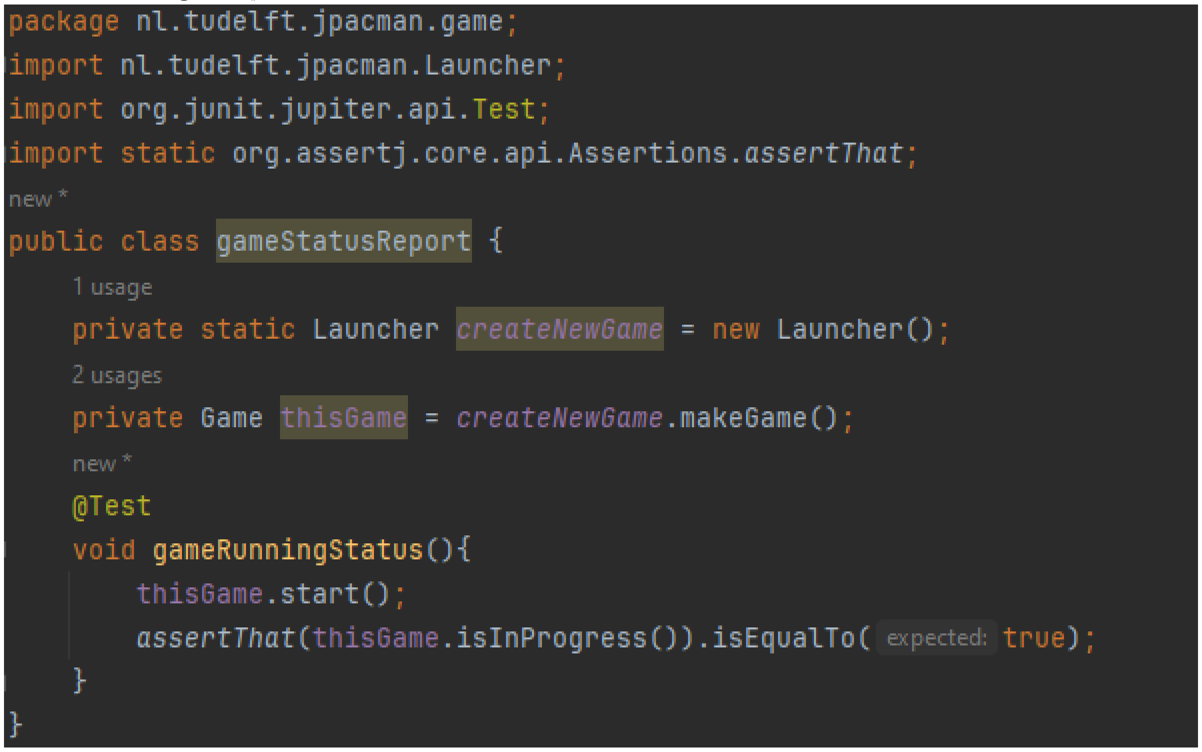
Code Snippets

BoardFactory.createGround()



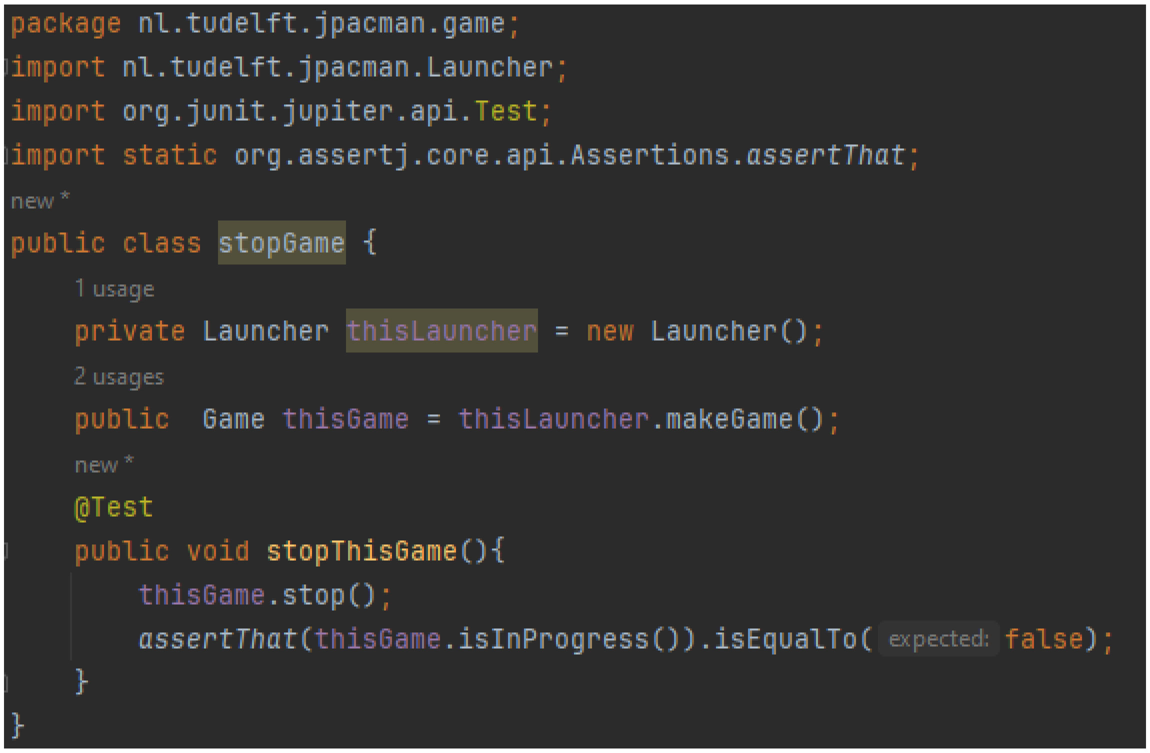
```
package nl.tudelft.jpacman.boardfactory;
import nl.tudelft.jpacman.board.BoardFactory;
import nl.tudelft.jpacman.sprite.PacManSprites;
import org.junit.jupiter.api.Test;
import static org.assertj.core.api.Assertions.assertThat;
new *
public class newGround {
    1 usage
    private static BoardFactory Factory = new BoardFactory(new PacManSprites());

    new *
    @Test
    void makeNewGround(){
        assertThat( actual: Factory.createGround() != null);
    }
}
```

Game.isInProgress()



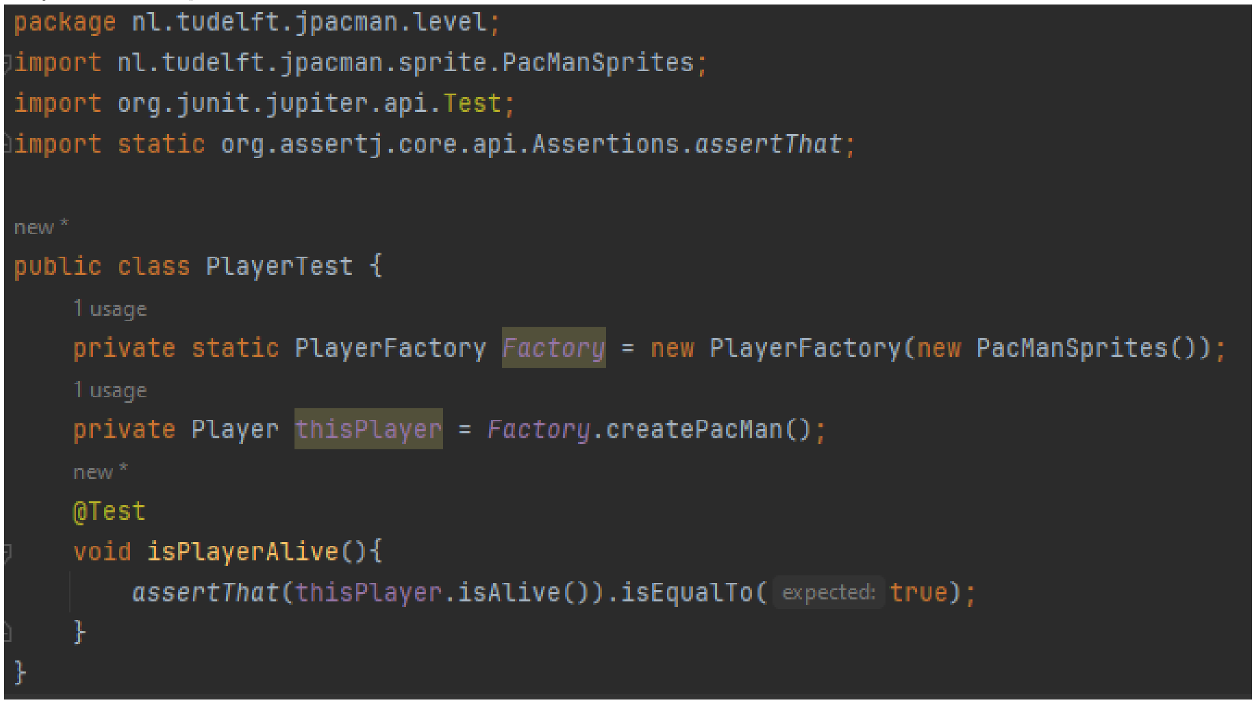
```
package nl.tudelft.jpacman.game;
import nl.tudelft.jpacman.Launcher;
import org.junit.jupiter.api.Test;
import static org.assertj.core.api.Assertions.assertThat;
new *
public class gameStatusReport {
    1 usage
    private static Launcher createNewGame = new Launcher();
    2 usages
    private Game thisGame = createNewGame.makeGame();
    new *
    @Test
    void gameRunningStatus(){
        thisGame.start();
        assertThat(thisGame.isInProgress()).isEqualTo( expected: true);
    }
}
```

Game.stop()



```
package nl.tudelft.jpacman.game;
import nl.tudelft.jpacman.Launcher;
import org.junit.jupiter.api.Test;
import static org.assertj.core.api.Assertions.assertThat;
new *
public class stopGame {
    1 usage
    private Launcher thisLauncher = new Launcher();
    2 usages
    public  Game thisGame = thisLauncher.makeGame();
    new *
    @Test
    public void stopThisGame(){
        thisGame.stop();
        assertThat(thisGame.isInProgress()).isEqualTo( expected: false);
    }
}
```

Player.isAlive()



```
package nl.tudelft.jpacman.level;
import nl.tudelft.jpacman.sprite.PacManSprites;
import org.junit.jupiter.api.Test;
import static org.assertj.core.api.Assertions.assertThat;

new *
public class PlayerTest {
    1 usage
    private static PlayerFactory Factory = new PlayerFactory(new PacManSprites());
    1 usage
    private Player thisPlayer = Factory.createPacMan();
    new *
    @Test
    void isPlayerAlive(){
        assertThat(thisPlayer.isAlive()).isEqualTo( expected: true);
    }
}
```