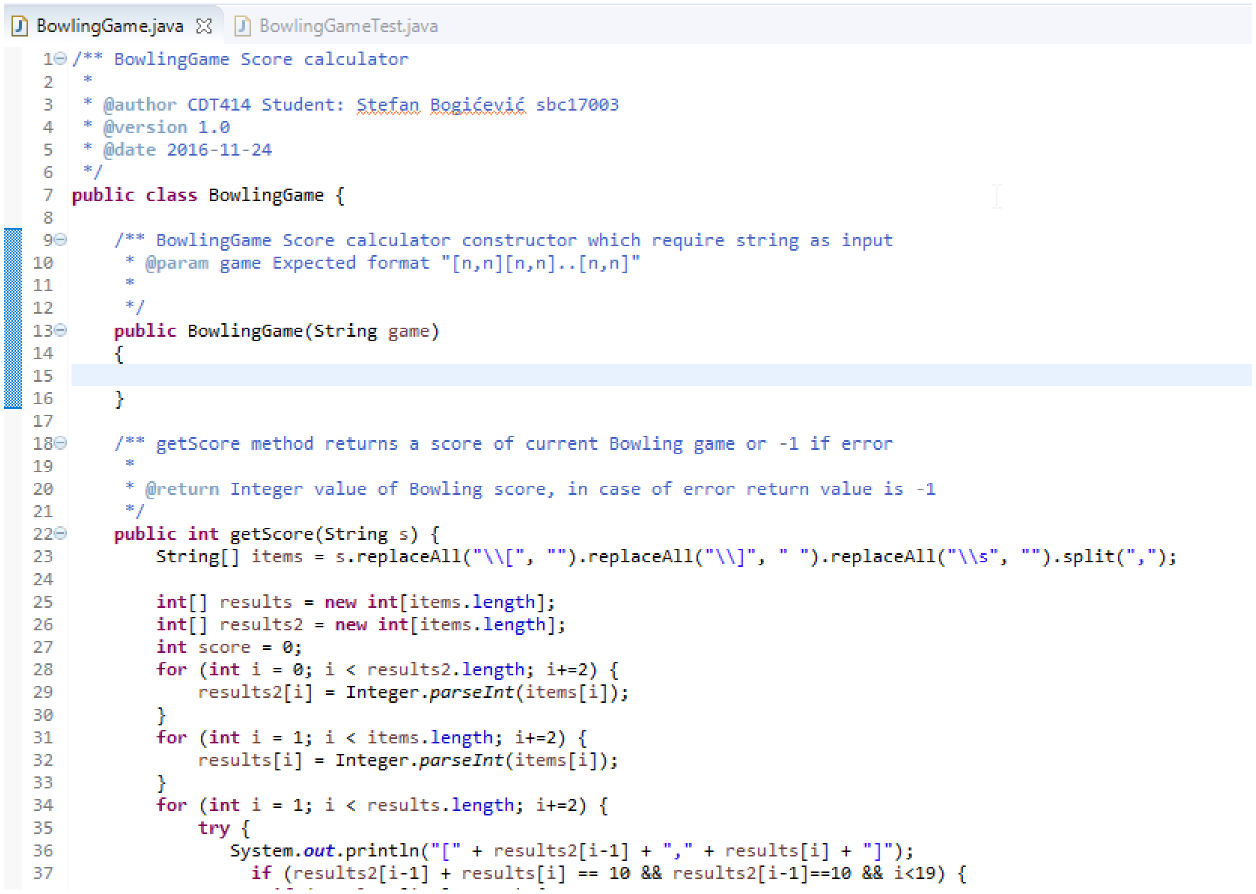
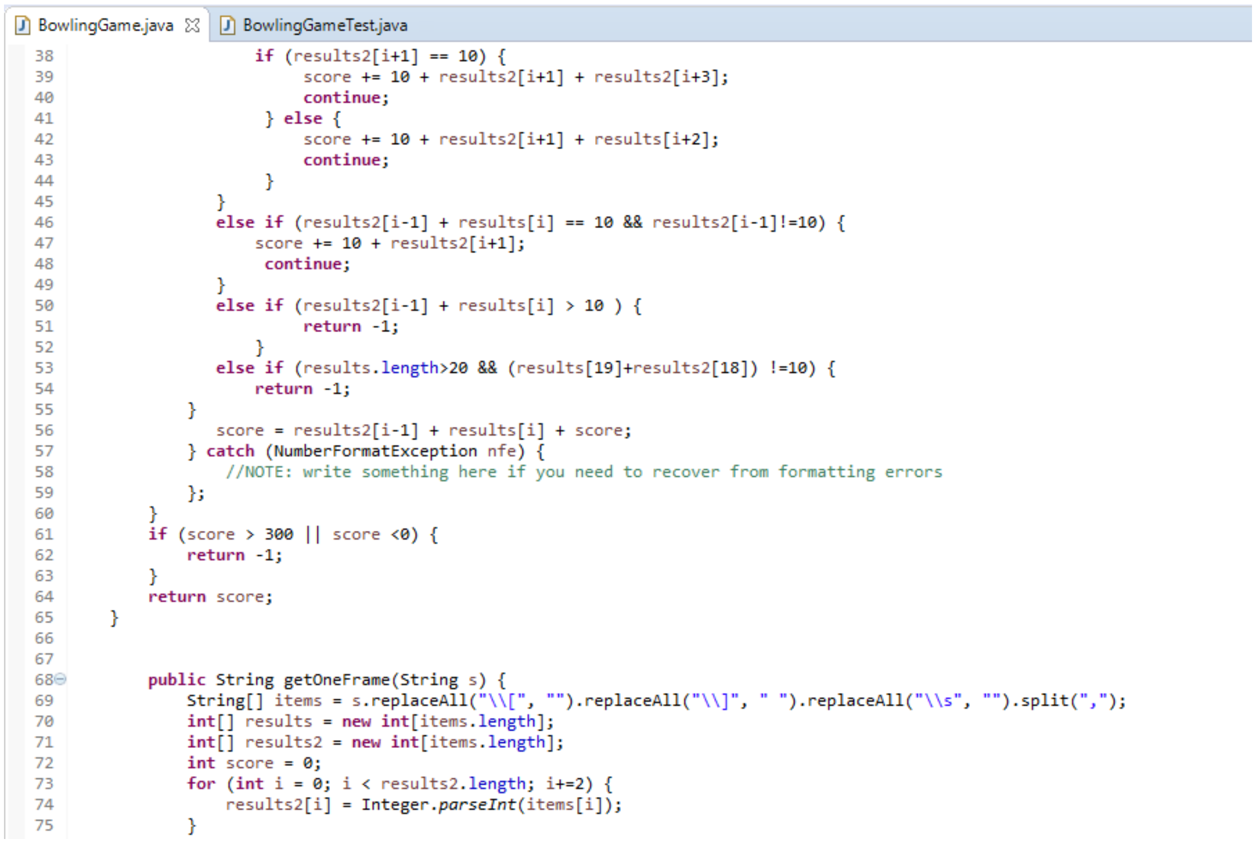
All code used for testing

BowlingGame.java – Class used for implementing code which tests the bowling calculator.


BowlingGame.java BowlingGameTest.java

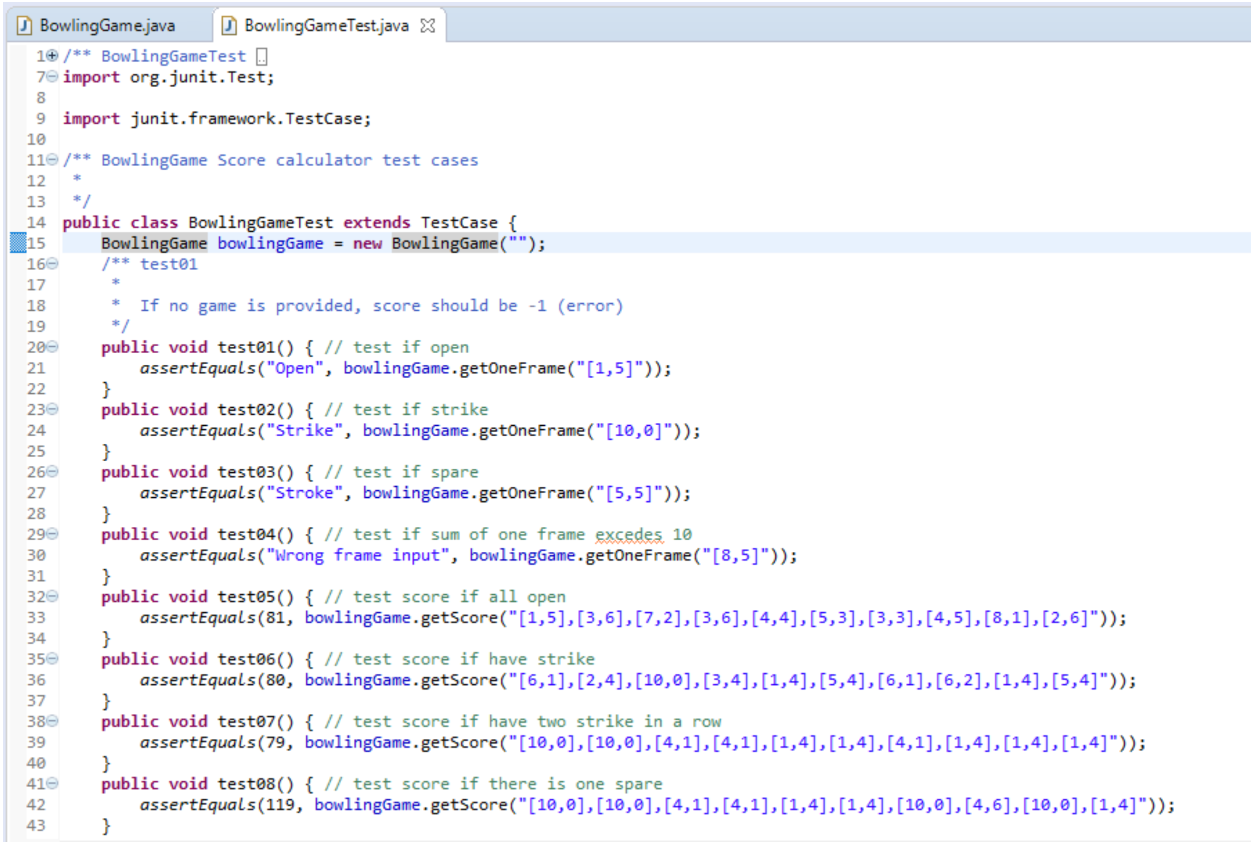
```
/** BowlingGame Score calculator
 *
 * @author CDT414 Student: Stefan Bogićević sbc17003
 * @version 1.0
 * @date 2016-11-24
 */
public class BowlingGame {

    /** BowlingGame Score calculator constructor which require string as input
     * @param game Expected format "[n,n][n,n]..[n,n]"
     *
     */
    public BowlingGame(String game)
    {

    }

    /** getScore method returns a score of current Bowling game or -1 if error
     *
     * @return Integer value of Bowling score, in case of error return value is -1
     */
    public int getScore(String s) {
        String[] items = s.replaceAll("\\[", "").replaceAll("\\]", " ").replaceAll("\\s", "").split(",");

        int[] results = new int[items.length];
        int[] results2 = new int[items.length];
        int score = 0;
        for (int i = 0; i < results2.length; i+=2) {
            results2[i] = Integer.parseInt(items[i]);
        }
        for (int i = 1; i < items.length; i+=2) {
            results[i] = Integer.parseInt(items[i]);
        }
        for (int i = 1; i < results.length; i+=2) {
            try {
               System.out.println("[" + results2[i-1] + "," + results[i] + "]");
                 if (results2[i-1] + results[i] == 10 && results2[i-1]==10 && i<19) {
```



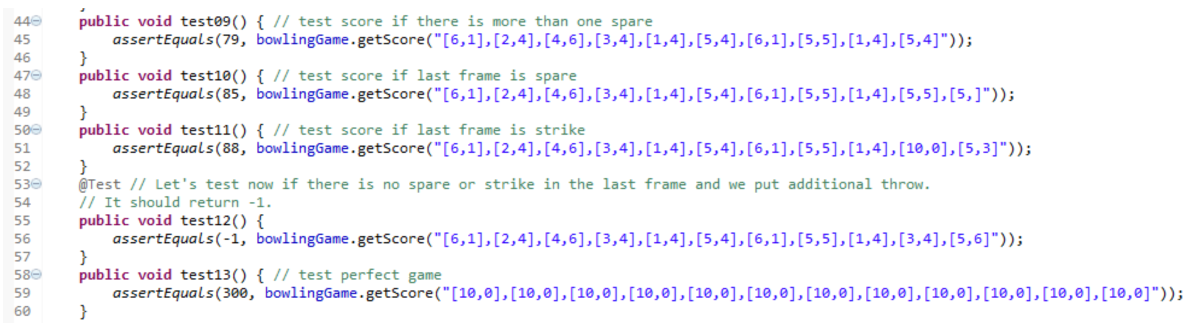
```
                   if (results2[i+1] == 10) {
                        score += 10 + results2[i+1] + results2[i+3];
                        continue;
                    } else {
                        score += 10 + results2[i+1] + results[i+2];
                        continue;
                    }
               }
               else if (results2[i-1] + results[i] == 10 && results2[i-1]!=10) {
                   score += 10 + results2[i+1];
                    continue;
               }
               else if (results2[i-1] + results[i] > 10 ) {
                        return -1;
                   }
               else if (results.length>20 && (results[19]+results2[18]) !=10) {
                   return -1;
            }
               score = results2[i-1] + results[i] + score;
            } catch (NumberFormatException nfe) {
                //NOTE: write something here if you need to recover from formatting errors
            };
        }
        if (score > 300 || score <0) {
            return -1;
        }
        return score;
    }


        public String getOneFrame(String s) {
            String[] items = s.replaceAll("\\[", "").replaceAll("\\]", " ").replaceAll("\\s", "").split(",");
            int[] results = new int[items.length];
            int[] results2 = new int[items.length];
            int score = 0;
            for (int i = 0; i < results2.length; i+=2) {
                results2[i] = Integer.parseInt(items[i]);
            }
```

```
    for (int i = 1; i < items.length; i+=2) {
        results[i] = Integer.parseInt(items[i]);
    }
    for (int i = 1; i < results.length; i+=2) {
        try {
           score = results2[i-1] + results[i] + score;
           if (results2[i-1] + results[i] < 10) {
            return "Open";
           }

           else  if (results2[i-1] + results[i] == 10 && results2[i-1]==10) {
                return "Strike";
           }
           else if (results2[i-1] + results[i] == 10 && results2[i-1]!=10) {
                return "Stroke";
           }
           else if (results2[i-1] + results[i] > 10 ) {
                    return "Wrong frame input";
               }
        } catch (NumberFormatException nfe) {
            //NOTE: write something here if you need to recover from formatting errors
        };
    }
return String.valueOf(score);
}
```

BowlingGameTest.java – Class containing all the test used for testing bowling calculator.

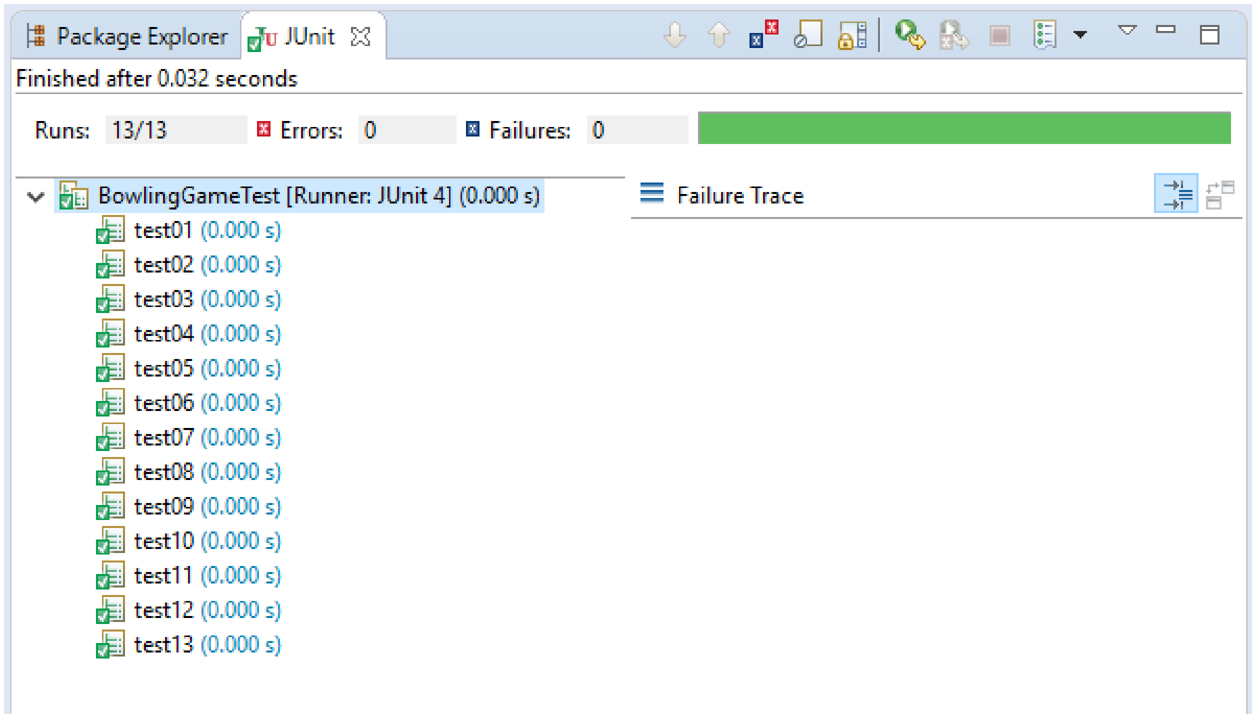


```java
/** BowlingGameTest
import org.junit.Test;

import junit.framework.TestCase;

/** BowlingGame Score calculator test cases
 *
 */
public class BowlingGameTest extends TestCase {
    BowlingGame bowlingGame = new BowlingGame("");
    /** test01
     *
     *  If no game is provided, score should be -1 (error)
     */
    public void test01() { // test if open
        assertEquals("Open", bowlingGame.getOneFrame("[1,5]"));
    }
    public void test02() { // test if strike
        assertEquals("Strike", bowlingGame.getOneFrame("[10,0]"));
    }
    public void test03() { // test if spare
        assertEquals("Stroke", bowlingGame.getOneFrame("[5,5]"));
    }
    public void test04() { // test if sum of one frame excedes 10
        assertEquals("Wrong frame input", bowlingGame.getOneFrame("[8,5]"));
    }
    public void test05() { // test score if all open
        assertEquals(81, bowlingGame.getScore("[1,5],[3,6],[7,2],[3,6],[4,4],[5,3],[3,3],[4,5],[8,1],[2,6]"));
    }
    public void test06() { // test score if have strike
        assertEquals(80, bowlingGame.getScore("[6,1],[2,4],[10,0],[3,4],[1,4],[5,4],[6,1],[6,2],[1,4],[5,4]"));
    }
    public void test07() { // test score if have two strike in a row
        assertEquals(79, bowlingGame.getScore("[10,0],[10,0],[4,1],[4,1],[1,4],[1,4],[4,1],[1,4],[1,4],[1,4]"));
    }
    public void test08() { // test score if there is one spare
        assertEquals(119, bowlingGame.getScore("[10,0],[10,0],[4,1],[4,1],[1,4],[1,4],[10,0],[4,6],[10,0],[1,4]"));
    }
```



```java
    public void test09() { // test score if there is more than one spare
        assertEquals(79, bowlingGame.getScore("[6,1],[2,4],[4,6],[3,4],[1,4],[5,4],[6,1],[5,5],[1,4],[5,4]"));
    }
    public void test10() { // test score if last frame is spare
        assertEquals(85, bowlingGame.getScore("[6,1],[2,4],[4,6],[3,4],[1,4],[5,4],[6,1],[5,5],[1,4],[5,5],[5,]"));
    }
    public void test11() { // test score if last frame is strike
        assertEquals(88, bowlingGame.getScore("[6,1],[2,4],[4,6],[3,4],[1,4],[5,4],[6,1],[5,5],[1,4],[10,0],[5,3]"));
    }
    @Test // Let's test now if there is no spare or strike in the last frame and we put additional throw.
    // It should return -1.
    public void test12() {
        assertEquals(-1, bowlingGame.getScore("[6,1],[2,4],[4,6],[3,4],[1,4],[5,4],[6,1],[5,5],[1,4],[3,4],[5,6]"));
    }
    public void test13() { // test perfect game
        assertEquals(300, bowlingGame.getScore("[10,0],[10,0],[10,0],[10,0],[10,0],[10,0],[10,0],[10,0],[10,0],[10,0],[10,0],[10,0]"));
    }
```

After writing code and testing it, all test are marked as pass, which can be seen in the picture below.



```
Finished after 0.032 seconds

Runs: 13/13    Errors: 0    Failures: 0

BowlingGameTest [Runner: JUnit 4] (0.000 s)
    test01 (0.000 s)
    test02 (0.000 s)
    test03 (0.000 s)
    test04 (0.000 s)
    test05 (0.000 s)
    test06 (0.000 s)
    test07 (0.000 s)
    test08 (0.000 s)
    test09 (0.000 s)
    test10 (0.000 s)
    test11 (0.000 s)
    test12 (0.000 s)
    test13 (0.000 s)
```

Stefan Bogićević (sbc17003)