GitHub Stuff\project-4---tic-tac-toe-WildestPantaloons\TicTacToeTester.java

```java
import java.awt.Point;

/**
 * A unit tester for classes that implement TicTacToe.
 *
 * @author mvail
 */
public class TicTacToeTester {

    // possible results expected in tests
    private static enum Result {
        True, False,
        X, O, Tie, InProgress,
        Pass, Fail,
        NoException, UnexpectedException
    };

    //tracking number of tests and test results
    private final int EXPECTED_TOTAL_TESTS;//initialized in constructor
    private int totalTests;
    private int passes = 0;
    private int failures = 0;
    private int totalRun = 0;
    private int secTotal = 0;
    private int secPasses = 0;
    private int secFails = 0;

    //control output - modified by command-line args
    private boolean printFailuresOnly = true;
    private boolean printSectionSummaries = true;

    /**
     * Valid command line args include:
     *  -a : print results from all tests (default is to print failed tests, only)
     *  -m : hide section summaries in output
     * @param args not used
     */
    public static void main(String[] args) {
        // to avoid every method being static
        TicTacToeTester tester = new TicTacToeTester(args);
        tester.runTests();

        /* Set a non-zero exit status if failures occurred during testing */
        if ( tester.getFailures() != 0 ) {
            System.exit(1);
        }

        System.exit(0);
    }

    /** tester constructor
     * @param args command line args
     */
```

```
54      public TicTacToeTester(String[] args) {
55          for (String arg : args) {
56              if (arg.equalsIgnoreCase("-a")) printFailuresOnly = false;
57              if (arg.equalsIgnoreCase("-m")) printSectionSummaries = false;
58          }
59          EXPECTED_TOTAL_TESTS = 219;
60          totalTests = 0;
61      }
62
63      /** Accessor method to check if failures
64       * occurred during testing.
65       * @return The total number of failures
66       */
67      public int getFailures() {
68          return this.failures;
69      }
70
71      /** Print test results in a consistent format
72       * @param testDesc description of the test
73       * @param result indicates if the test passed or failed
74       */
75      private void printTest(String testDesc, boolean result) {
76          totalRun++;
77          if (result) { passes++; }
78          else { failures++; }
79          if (!result || !printFailuresOnly) {
80              System.out.printf("%-46s\t%s\n", testDesc, (result ? "   PASS" : "***FAIL***"));
81          }
82      }
83
84      /** Print a final summary */
85      private void printFinalSummary() {
86          String verdict = String.format("\nTotal Tests Run: %d of %d,  Passed: %d (%.1f%%),
    Failed: %d\n",
87                  totalRun, totalTests, passes, passes*100.0/totalTests, failures);
88          String line = "";
89          for (int i = 0; i < verdict.length(); i++) {
90              line += "-";
91          }
92          System.out.println(line);
93          System.out.println(verdict);
94          if(totalTests != EXPECTED_TOTAL_TESTS) {
95              System.out.printf("Expected %d total tests, but evaluated %d.\n",
96                      EXPECTED_TOTAL_TESTS, totalTests);
97          }
98      }
99
100     /** Print a section summary */
101     private void printSectionSummary(String secLabel) {
102         secTotal = totalTests - secTotal;
103         secPasses = passes - secPasses;
104         secFails = failures - secFails;
105         System.out.printf("\n%s Tests: %d,  Passed: %d,  Failed: %d\n", secLabel, secTotal,
    secPasses, secFails);
106         secTotal = totalTests; //reset for next section
107         secPasses = passes;
108         secFails = failures;
```

```
109         System.out.printf("Tests Run So Far: %d of %d,  Passed: %d (%.1f%%),  Failed: %d\n",
110                 totalRun, EXPECTED_TOTAL_TESTS, passes, passes*100.0/EXPECTED_TOTAL_TESTS,
    failures);
111     }
112
113     // XXX runTests()
114     //  see the blue box on the right of the scroll bar? the triple-X tag aids in navigating
    long files
115
116     /** Run tests to confirm required functionality from list constructors and methods */
117     private void runTests() {
118         //brand new game
119         testNewGame();
120         //progress toward a tie game
121         testX00();
122         testX00O10();
123         testX00O10X11();
124         testX00O10X11O22();
125         testX00O10X11O22X02();
126         testX00O10X11O22X02O01();
127         testX00O10X11O22X02O01X21();
128         testX00O10X11O22X02O01X21O20();
129         testX00O10X11O22X02O01X21O20X12(); //tie game
130         testX02O10X00O22X01(); //X wins - first row
131         testX11O00X10O22X12(); //X wins - second row
132         testX22O00X20O12X21(); //X wins - third row
133         testX00O11X20O12X10(); //X wins - first col
134         testX01O10X21O12X11(); //X wins - second col
135         testX12O11X02O21X22(); //X wins - third col
136         testX11O02X22O12X00(); //X wins - first diagonal
137         testX20O01X11O00X02(); //X wins - second diagonal
138         testX10O01X20O02X11O00(); //O wins - first row
139         testX00O10X20O12X01O11(); //O wins - second row
140         testX00O21X10O20X01O22(); //O wins - third row
141         testX22O10X11O00X01O20(); //O wins - first col
142         testX00O01X22O11X10O21(); //O wins - second col
143         testX11O02X00O22X10O12(); //O wins - third col
144         testX01O00X10O11X21O22(); //O wins - first diagonal
145         testX00O11X10O20X21O02(); //O wins - second diagonal
146         //last move (9th move) is a winning move
147         testX01O12X00O02X10O20X11O21X22(); //X wins
148         testO01X12O00X02O10X20O11X21O22(); //O wins
149         //reset game
150         testX00O10X11NewGame(); //new game after partial game
151         testX01O12X00O02X10O20X11O21X22NewGame(); //new game after X win
152         testO01X12O00X02O10X20O11X21O22NewGame(); //new game after O win
153         //encapsulation tests
154         testEncapsulation();
155
156         // report final verdict
157         printFinalSummary();
158     }
159
160     ////////////////////////
161     //XXX Scenario Tests
162     ////////////////////////
163
```

```
164     private void testNewGame() {
165         TicTacToe.BoardChoice[][] grid = {
166                 {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN,
    TicTacToe.BoardChoice.OPEN},
167                 {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN,
    TicTacToe.BoardChoice.OPEN},
168                 {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN,
    TicTacToe.BoardChoice.OPEN}
169         };
170         Point[] newGameMoves = new Point[0];
171         TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.OPEN;
172         boolean gameOver = false;
173 
174         String scenarioName = "testNewGame";
175         System.out.println("\nSCENARIO: " + scenarioName + "\n");
176         totalTests += 7;
177         try {
178             printTest("testNewGame", testNewGame(newGame()));
179             printTest("testGameOver", testGameOver(newGame(), Result.False));
180             printTest("testGameState", testGameState(newGame(), Result.InProgress));
181             printTest("testGetGameGrid", testGetGameGrid(newGame(), grid));
182             printTest("testGetMoves", testGetMoves(newGame(), newGameMoves));
183             printTest("testChoicesX", testChoices(newGame, TicTacToe.BoardChoice.X,
    lastPlayer, gameOver, grid));
184             printTest("testChoicesO", testChoices(newGame, TicTacToe.BoardChoice.O,
    lastPlayer, gameOver, grid));
185         } catch (Exception e) {
186             System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
187             e.printStackTrace();
188         } finally {
189             if (printSectionSummaries) {
190                 printSectionSummary("Section");
191             }
192         }
193     }
194 
195     private void testX00() {
196         TicTacToe.BoardChoice[][] grid = {
197                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.OPEN,
    TicTacToe.BoardChoice.OPEN},
198                 {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN,
    TicTacToe.BoardChoice.OPEN},
199                 {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN,
    TicTacToe.BoardChoice.OPEN}
200         };
201         Point[] moves = {new Point(0,0)};
202         TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.X;
203         boolean gameOver = false;
204 
205         String scenarioName = "testX00";
206         System.out.println("\nSCENARIO: " + scenarioName + "\n");
207         totalTests += 7;
208         try {
209             printTest("testNewGame", testNewGame(gameX00()));
210             printTest("testGameOver", testGameOver(gameX00(), Result.False));
211             printTest("testGameState", testGameState(gameX00(), Result.InProgress));
212             printTest("testGetGameGrid", testGetGameGrid(gameX00(), grid));
213             printTest("testGetMoves", testGetMoves(gameX00(), moves));
```

```
214             printTest("testChoicesX", testChoices(gameX00, TicTacToe.BoardChoice.X,
    lastPlayer, gameOver, grid));
215             printTest("testChoicesO", testChoices(gameX00, TicTacToe.BoardChoice.O,
    lastPlayer, gameOver, grid));
216         } catch (Exception e) {
217             System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
218             e.printStackTrace();
219         } finally {
220             if (printSectionSummaries) {
221                 printSectionSummary("Section");
222             }
223         }
224     }
225 
226     private void testX00O10() {
227         TicTacToe.BoardChoice[][] grid = {
228                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.OPEN,
    TicTacToe.BoardChoice.OPEN},
229                 {TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.OPEN,
    TicTacToe.BoardChoice.OPEN},
230                 {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN,
    TicTacToe.BoardChoice.OPEN}
231         };
232         Point[] moves = {new Point(0,0), new Point(1,0)};
233         TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.O;
234         boolean gameOver = false;
235 
236         String scenarioName = "testX00O10";
237         System.out.println("\nSCENARIO: " + scenarioName + "\n");
238         totalTests += 7;
239         try {
240             printTest("testNewGame", testNewGame(gameX00O10()));
241             printTest("testGameOver", testGameOver(gameX00O10(), Result.False));
242             printTest("testGameState", testGameState(gameX00O10(), Result.InProgress));
243             printTest("testGetGameGrid", testGetGameGrid(gameX00O10(), grid));
244             printTest("testGetMoves", testGetMoves(gameX00O10(), moves));
245             printTest("testChoicesX", testChoices(gameX00O10, TicTacToe.BoardChoice.X,
    lastPlayer, gameOver, grid));
246             printTest("testChoicesO", testChoices(gameX00O10, TicTacToe.BoardChoice.O,
    lastPlayer, gameOver, grid));
247         } catch (Exception e) {
248             System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
249             e.printStackTrace();
250         } finally {
251             if (printSectionSummaries) {
252                 printSectionSummary("Section");
253             }
254         }
255     }
256 
257     private void testX00O10X11() {
258         TicTacToe.BoardChoice[][] grid = {
259                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.OPEN,
    TicTacToe.BoardChoice.OPEN},
260                 {TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.X,
    TicTacToe.BoardChoice.OPEN},
261                 {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN,
    TicTacToe.BoardChoice.OPEN}
```

```
262         };
263         Point[] moves = {new Point(0,0), new Point(1,0), new Point(1,1)};
264         TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.X;
265         boolean gameOver = false;
266 
267         String scenarioName = "testX00O10X11";
268         System.out.println("\nSCENARIO: " + scenarioName + "\n");
269         totalTests += 7;
270         try {
271             printTest("testNewGame", testNewGame(gameX00O10X11()));
272             printTest("testGameOver", testGameOver(gameX00O10X11(), Result.False));
273             printTest("testGameState", testGameState(gameX00O10X11(), Result.InProgress));
274             printTest("testGetGameGrid", testGetGameGrid(gameX00O10X11(), grid));
275             printTest("testGetMoves", testGetMoves(gameX00O10X11(), moves));
276             printTest("testChoicesX", testChoices(gameX00O10X11, TicTacToe.BoardChoice.X,
lastPlayer, gameOver, grid));
277             printTest("testChoicesO", testChoices(gameX00O10X11, TicTacToe.BoardChoice.O,
lastPlayer, gameOver, grid));
278         } catch (Exception e) {
279             System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
280             e.printStackTrace();
281         } finally {
282             if (printSectionSummaries) {
283                 printSectionSummary("Section");
284             }
285         }
286     }
287 
288     private void testX00O10X11O22() {
289         TicTacToe.BoardChoice[][] grid = {
290                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.OPEN,
TicTacToe.BoardChoice.OPEN},
291                 {TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.X,
TicTacToe.BoardChoice.OPEN},
292                 {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN,
TicTacToe.BoardChoice.O}
293         };
294         Point[] moves = {new Point(0,0), new Point(1,0), new Point(1,1),
295                 new Point(2,2)};
296         TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.O;
297         boolean gameOver = false;
298 
299         String scenarioName = "testX00O10X11O22";
300         System.out.println("\nSCENARIO: " + scenarioName + "\n");
301         totalTests += 7;
302         try {
303             printTest("testNewGame", testNewGame(gameX00O10X11O22()));
304             printTest("testGameOver", testGameOver(gameX00O10X11O22(), Result.False));
305             printTest("testGameState", testGameState(gameX00O10X11O22(), Result.InProgress))
;
306             printTest("testGetGameGrid", testGetGameGrid(gameX00O10X11O22(), grid));
307             printTest("testGetMoves", testGetMoves(gameX00O10X11O22(), moves));
308             printTest("testChoicesX", testChoices(gameX00O10X11O22, TicTacToe.BoardChoice.X,
lastPlayer, gameOver, grid));
309             printTest("testChoicesO", testChoices(gameX00O10X11O22, TicTacToe.BoardChoice.O,
lastPlayer, gameOver, grid));
310         } catch (Exception e) {
311             System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
```

```
312             e.printStackTrace();
313         } finally {
314             if (printSectionSummaries) {
315                 printSectionSummary("Section");
316             }
317         }
318     }
319 
320     private void testX00O10X11O22X02() {
321         TicTacToe.BoardChoice[][] grid = {
322                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.OPEN,
TicTacToe.BoardChoice.X},
323                 {TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.X,
TicTacToe.BoardChoice.OPEN},
324                 {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN,
TicTacToe.BoardChoice.O}
325         };
326         Point[] moves = {new Point(0,0), new Point(1,0), new Point(1,1),
327                 new Point(2,2), new Point(0,2)};
328         TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.X;
329         boolean gameOver = false;
330 
331         String scenarioName = "testX00O10X11O22X02";
332         System.out.println("\nSCENARIO: " + scenarioName + "\n");
333         totalTests += 7;
334         try {
335             printTest("testNewGame", testNewGame(gameX00O10X11O22X02()));
336             printTest("testGameOver", testGameOver(gameX00O10X11O22X02(), Result.False));
337             printTest("testGameState", testGameState(gameX00O10X11O22X02(),
Result.InProgress));
338             printTest("testGetGameGrid", testGetGameGrid(gameX00O10X11O22X02(), grid));
339             printTest("testGetMoves", testGetMoves(gameX00O10X11O22X02(), moves));
340             printTest("testChoicesX", testChoices(gameX00O10X11O22X02,
TicTacToe.BoardChoice.X, lastPlayer, gameOver, grid));
341             printTest("testChoicesO", testChoices(gameX00O10X11O22X02,
TicTacToe.BoardChoice.O, lastPlayer, gameOver, grid));
342         } catch (Exception e) {
343             System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
344             e.printStackTrace();
345         } finally {
346             if (printSectionSummaries) {
347                 printSectionSummary("Section");
348             }
349         }
350     }
351 
352     private void testX00O10X11O22X02O01() {
353         TicTacToe.BoardChoice[][] grid = {
354                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.X},
355                 {TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.X,
TicTacToe.BoardChoice.OPEN},
356                 {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN,
TicTacToe.BoardChoice.O}
357         };
358         Point[] moves = {new Point(0,0), new Point(1,0), new Point(1,1),
359                 new Point(2,2), new Point(0,2), new Point(0,1)};
360         TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.O;
361         boolean gameOver = false;
```

```
362
363         String scenarioName = "testX00O10X11O22X02O01";
364         System.out.println("\nSCENARIO: " + scenarioName + "\n");
365         totalTests += 7;
366         try {
367             printTest("testNewGame", testNewGame(gameX00O10X11O22X02O01()));
368             printTest("testGameOver", testGameOver(gameX00O10X11O22X02O01(), Result.False));
369             printTest("testGameState", testGameState(gameX00O10X11O22X02O01(),
Result.InProgress));
370             printTest("testGetGameGrid", testGetGameGrid(gameX00O10X11O22X02O01(), grid));
371             printTest("testGetMoves", testGetMoves(gameX00O10X11O22X02O01(), moves));
372             printTest("testChoicesX", testChoices(gameX00O10X11O22X02O01,
TicTacToe.BoardChoice.X, lastPlayer, gameOver, grid));
373             printTest("testChoicesO", testChoices(gameX00O10X11O22X02O01,
TicTacToe.BoardChoice.O, lastPlayer, gameOver, grid));
374         } catch (Exception e) {
375             System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
376             e.printStackTrace();
377         } finally {
378             if (printSectionSummaries) {
379                 printSectionSummary("Section");
380             }
381         }
382     }
383
384     private void testX00O10X11O22X02O01X21() {
385         TicTacToe.BoardChoice[][] grid = {
386                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.X},
387                 {TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.X,
TicTacToe.BoardChoice.OPEN},
388                 {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.X,
TicTacToe.BoardChoice.O}
389         };
390         Point[] moves = {new Point(0,0), new Point(1,0), new Point(1,1),
391                 new Point(2,2), new Point(0,2), new Point(0,1), new Point(2,1)};
392         TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.X;
393         boolean gameOver = false;
394
395         String scenarioName = "testX00O10X11O22X02O01X21";
396         System.out.println("\nSCENARIO: " + scenarioName + "\n");
397         totalTests += 7;
398         try {
399             printTest("testNewGame", testNewGame(gameX00O10X11O22X02O01X21()));
400             printTest("testGameOver", testGameOver(gameX00O10X11O22X02O01X21(),
Result.False));
401             printTest("testGameState", testGameState(gameX00O10X11O22X02O01X21(),
Result.InProgress));
402             printTest("testGetGameGrid", testGetGameGrid(gameX00O10X11O22X02O01X21(), grid))
;
403             printTest("testGetMoves", testGetMoves(gameX00O10X11O22X02O01X21(), moves));
404             printTest("testChoicesX", testChoices(gameX00O10X11O22X02O01X21,
TicTacToe.BoardChoice.X, lastPlayer, gameOver, grid));
405             printTest("testChoicesO", testChoices(gameX00O10X11O22X02O01X21,
TicTacToe.BoardChoice.O, lastPlayer, gameOver, grid));
406         } catch (Exception e) {
407             System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
408             e.printStackTrace();
409         } finally {
```

```
410             if (printSectionSummaries) {
411                 printSectionSummary("Section");
412             }
413         }
414     }
415 
416     private void testX00O10X11O22X02O01X21O20() {
417         TicTacToe.BoardChoice[][] grid = {
418                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.X},
419                 {TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.X,
TicTacToe.BoardChoice.OPEN},
420                 {TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.O}
421         };
422         Point[] moves = {new Point(0,0), new Point(1,0), new Point(1,1),
423                 new Point(2,2), new Point(0,2), new Point(0,1), new Point(2,1), new Point(2,
0)};
424         TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.O;
425         boolean gameOver = false;
426 
427         String scenarioName = "testX00O10X11O22X02O01X21O20";
428         System.out.println("\nSCENARIO: " + scenarioName + "\n");
429         totalTests += 7;
430         try {
431             printTest("testNewGame", testNewGame(gameX00O10X11O22X02O01X21O20()));
432             printTest("testGameOver", testGameOver(gameX00O10X11O22X02O01X21O20(),
Result.False));
433             printTest("testGameState", testGameState(gameX00O10X11O22X02O01X21O20(),
Result.InProgress));
434             printTest("testGetGameGrid", testGetGameGrid(gameX00O10X11O22X02O01X21O20(),
grid));
435             printTest("testGetMoves", testGetMoves(gameX00O10X11O22X02O01X21O20(), moves));
436             printTest("testChoicesX", testChoices(gameX00O10X11O22X02O01X21O20,
TicTacToe.BoardChoice.X, lastPlayer, gameOver, grid));
437             printTest("testChoicesO", testChoices(gameX00O10X11O22X02O01X21O20,
TicTacToe.BoardChoice.O, lastPlayer, gameOver, grid));
438         } catch (Exception e) {
439             System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
440             e.printStackTrace();
441         } finally {
442             if (printSectionSummaries) {
443                 printSectionSummary("Section");
444             }
445         }
446     }
447 
448     // complete tie game
449     private void testX00O10X11O22X02O01X21O20X12() {
450         TicTacToe.BoardChoice[][] grid = {
451                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.X},
452                 {TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.X},
453                 {TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.O}
454         };
455         Point[] moves = {new Point(0,0), new Point(1,0), new Point(1,1),
456                 new Point(2,2), new Point(0,2), new Point(0,1), new Point(2,1),
457                 new Point(2,0), new Point(1,2)};
458         TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.X;
459         boolean gameOver = true;
460 
```

```
461         String scenarioName = "testX00O10X11O22X02O01X21O20X12";
462         System.out.println("\nSCENARIO: " + scenarioName + "\n");
463         totalTests += 7;
464         try {
465             printTest("testNewGame", testNewGame(gameX00O10X11O22X02O01X21O20X12()));
466             printTest("testGameOver", testGameOver(gameX00O10X11O22X02O01X21O20X12(),
Result.True));
467             printTest("testGameState", testGameState(gameX00O10X11O22X02O01X21O20X12(),
Result.Tie));
468             printTest("testGetGameGrid", testGetGameGrid(gameX00O10X11O22X02O01X21O20X12(),
grid));
469             printTest("testGetMoves", testGetMoves(gameX00O10X11O22X02O01X21O20X12(), moves)
);
470             printTest("testChoicesX", testChoices(gameX00O10X11O22X02O01X21O20X12,
TicTacToe.BoardChoice.X, lastPlayer, gameOver, grid));
471             printTest("testChoicesO", testChoices(gameX00O10X11O22X02O01X21O20X12,
TicTacToe.BoardChoice.O, lastPlayer, gameOver, grid));
472         } catch (Exception e) {
473             System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
474             e.printStackTrace();
475         } finally {
476             if (printSectionSummaries) {
477                 printSectionSummary("Section");
478             }
479         }
480     }
481
482     //XXX X wins scenarios
483
484     //X wins - first row
485     private void testX02O10X00O22X01() {
486         TicTacToe.BoardChoice[][] grid = {
487                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.X},
488                 {TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.OPEN,
TicTacToe.BoardChoice.OPEN},
489                 {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN,
TicTacToe.BoardChoice.O}
490         };
491         Point[] moves = {new Point(0,2), new Point(1,0), new Point(0,0),
492                 new Point(2,2), new Point(0,1)};
493         TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.X;
494         boolean gameOver = true;
495
496         String scenarioName = "testX02O10X00O22X01";
497         System.out.println("\nSCENARIO: " + scenarioName + "\n");
498         totalTests += 7;
499         try {
500             printTest("testNewGame", testNewGame(gameX02O10X00O22X01()));
501             printTest("testGameOver", testGameOver(gameX02O10X00O22X01(), Result.True));
502             printTest("testGameState", testGameState(gameX02O10X00O22X01(), Result.X));
503             printTest("testGetGameGrid", testGetGameGrid(gameX02O10X00O22X01(), grid));
504             printTest("testGetMoves", testGetMoves(gameX02O10X00O22X01(), moves));
505             printTest("testChoicesX", testChoices(gameX02O10X00O22X01,
TicTacToe.BoardChoice.X, lastPlayer, gameOver, grid));
506             printTest("testChoicesO", testChoices(gameX02O10X00O22X01,
TicTacToe.BoardChoice.O, lastPlayer, gameOver, grid));
507         } catch (Exception e) {
508             System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
```

```
509             e.printStackTrace();
510         } finally {
511             if (printSectionSummaries) {
512                 printSectionSummary("Section");
513             }
514         }
515     }
516 
517     //X wins - second row
518     private void testX11O00X10O22X12() {
519         TicTacToe.BoardChoice[][] grid = {
520                 {TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN},
521                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.X},
522                 {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.O}
523         };
524         Point[] moves = {new Point(1,1), new Point(0,0), new Point(1,0),
525                 new Point(2,2), new Point(1,2)};
526         TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.X;
527         boolean gameOver = true;
528 
529         String scenarioName = "testX11O00X10O22X12";
530         System.out.println("\nSCENARIO: " + scenarioName + "\n");
531         totalTests += 7;
532         try {
533             printTest("testNewGame", testNewGame(gameX11O00X10O22X12()));
534             printTest("testGameOver", testGameOver(gameX11O00X10O22X12(), Result.True));
535             printTest("testGameState", testGameState(gameX11O00X10O22X12(), Result.X));
536             printTest("testGetGameGrid", testGetGameGrid(gameX11O00X10O22X12(), grid));
537             printTest("testGetMoves", testGetMoves(gameX11O00X10O22X12(), moves));
538             printTest("testChoicesX", testChoices(gameX11O00X10O22X12, TicTacToe.BoardChoice.X, lastPlayer, gameOver, grid));
539             printTest("testChoicesO", testChoices(gameX11O00X10O22X12, TicTacToe.BoardChoice.O, lastPlayer, gameOver, grid));
540         } catch (Exception e) {
541             System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
542             e.printStackTrace();
543         } finally {
544             if (printSectionSummaries) {
545                 printSectionSummary("Section");
546             }
547         }
548     }
549 
550     //X wins - third row
551     private void testX22O00X20O12X21() {
552         TicTacToe.BoardChoice[][] grid = {
553                 {TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN},
554                 {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.O},
555                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.X}
556         };
557         Point[] moves = {new Point(2,2), new Point(0,0), new Point(2,0),
558                 new Point(1,2), new Point(2,1)};
559         TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.X;
```

```
560         boolean gameOver = true;
561 
562         String scenarioName = "testX22O00X20O12X21";
563         System.out.println("\nSCENARIO: " + scenarioName + "\n");
564         totalTests += 7;
565         try {
566             printTest("testNewGame", testNewGame(gameX22O00X20O12X21()));
567             printTest("testGameOver", testGameOver(gameX22O00X20O12X21(), Result.True));
568             printTest("testGameState", testGameState(gameX22O00X20O12X21(), Result.X));
569             printTest("testGetGameGrid", testGetGameGrid(gameX22O00X20O12X21(), grid));
570             printTest("testGetMoves", testGetMoves(gameX22O00X20O12X21(), moves));
571             printTest("testChoicesX", testChoices(gameX22O00X20O12X21,
TicTacToe.BoardChoice.X, lastPlayer, gameOver, grid));
572             printTest("testChoicesO", testChoices(gameX22O00X20O12X21,
TicTacToe.BoardChoice.O, lastPlayer, gameOver, grid));
573         } catch (Exception e) {
574             System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
575             e.printStackTrace();
576         } finally {
577             if (printSectionSummaries) {
578                 printSectionSummary("Section");
579             }
580         }
581     }
582 
583     //X wins - first col
584     private void testX00O11X20O12X10() {
585         TicTacToe.BoardChoice[][] grid = {
586                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.OPEN,
TicTacToe.BoardChoice.OPEN},
587                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.O},
588                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.OPEN,
TicTacToe.BoardChoice.OPEN}
589         };
590         Point[] moves = {new Point(0,0), new Point(1,1), new Point(2,0),
591                 new Point(1,2), new Point(1,0)};
592         TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.X;
593         boolean gameOver = true;
594 
595         String scenarioName = "testX00O11X20O12X10";
596         System.out.println("\nSCENARIO: " + scenarioName + "\n");
597         totalTests += 7;
598         try {
599             printTest("testNewGame", testNewGame(gameX00O11X20O12X10()));
600             printTest("testGameOver", testGameOver(gameX00O11X20O12X10(), Result.True));
601             printTest("testGameState", testGameState(gameX00O11X20O12X10(), Result.X));
602             printTest("testGetGameGrid", testGetGameGrid(gameX00O11X20O12X10(), grid));
603             printTest("testGetMoves", testGetMoves(gameX00O11X20O12X10(), moves));
604             printTest("testChoicesX", testChoices(gameX00O11X20O12X10,
TicTacToe.BoardChoice.X, lastPlayer, gameOver, grid));
605             printTest("testChoicesO", testChoices(gameX00O11X20O12X10,
TicTacToe.BoardChoice.O, lastPlayer, gameOver, grid));
606         } catch (Exception e) {
607             System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
608             e.printStackTrace();
609         } finally {
610             if (printSectionSummaries) {
```

```
611                 printSectionSummary("Section");
612             }
613         }
614     }
615
616     //X wins - second col
617     private void testX01O10X21O12X11() {
618         TicTacToe.BoardChoice[][] grid = {
619                 {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.X,
TicTacToe.BoardChoice.OPEN},
620                 {TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.O},
621                 {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.X,
TicTacToe.BoardChoice.OPEN}
622         };
623         Point[] moves = {new Point(0,1), new Point(1,0), new Point(2,1),
624                 new Point(1,2), new Point(1,1)};
625         TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.X;
626         boolean gameOver = true;
627
628         String scenarioName = "testX01O10X21O12X11";
629         System.out.println("\nSCENARIO: " + scenarioName + "\n");
630         totalTests += 7;
631         try {
632             printTest("testNewGame", testNewGame(gameX01O10X21O12X11()));
633             printTest("testGameOver", testGameOver(gameX01O10X21O12X11(), Result.True));
634             printTest("testGameState", testGameState(gameX01O10X21O12X11(), Result.X));
635             printTest("testGetGameGrid", testGetGameGrid(gameX01O10X21O12X11(), grid));
636             printTest("testGetMoves", testGetMoves(gameX01O10X21O12X11(), moves));
637             printTest("testChoicesX", testChoices(gameX01O10X21O12X11,
TicTacToe.BoardChoice.X, lastPlayer, gameOver, grid));
638             printTest("testChoicesO", testChoices(gameX01O10X21O12X11,
TicTacToe.BoardChoice.O, lastPlayer, gameOver, grid));
639         } catch (Exception e) {
640             System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
641             e.printStackTrace();
642         } finally {
643             if (printSectionSummaries) {
644                 printSectionSummary("Section");
645             }
646         }
647     }
648
649     //X wins - third col
650     private void testX12O11X02O21X22() {
651         TicTacToe.BoardChoice[][] grid = {
652                 {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN,
TicTacToe.BoardChoice.X},
653                 {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.O,
TicTacToe.BoardChoice.X},
654                 {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.O,
TicTacToe.BoardChoice.X}
655         };
656         Point[] moves = {new Point(1,2), new Point(1,1), new Point(0,2),
657                 new Point(2,1), new Point(2,2)};
658         TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.X;
659         boolean gameOver = true;
660
661         String scenarioName = "testX12O11X02O21X22";
```

```
662         System.out.println("\nSCENARIO: " + scenarioName + "\n");
663         totalTests += 7;
664         try {
665             printTest("testNewGame", testNewGame(gameX12O11X02O21X22()));
666             printTest("testGameOver", testGameOver(gameX12O11X02O21X22(), Result.True));
667             printTest("testGameState", testGameState(gameX12O11X02O21X22(), Result.X));
668             printTest("testGetGameGrid", testGetGameGrid(gameX12O11X02O21X22(), grid));
669             printTest("testGetMoves", testGetMoves(gameX12O11X02O21X22(), moves));
670             printTest("testChoicesX", testChoices(gameX12O11X02O21X22,
TicTacToe.BoardChoice.X, lastPlayer, gameOver, grid));
671             printTest("testChoicesO", testChoices(gameX12O11X02O21X22,
TicTacToe.BoardChoice.O, lastPlayer, gameOver, grid));
672         } catch (Exception e) {
673             System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
674             e.printStackTrace();
675         } finally {
676             if (printSectionSummaries) {
677                 printSectionSummary("Section");
678             }
679         }
680     }
681
682     //X wins - first diagonal
683     private void testX11O02X22O12X00() {
684         TicTacToe.BoardChoice[][] grid = {
685                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.OPEN,
TicTacToe.BoardChoice.O},
686                 {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.X,
TicTacToe.BoardChoice.O},
687                 {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN,
TicTacToe.BoardChoice.X}
688         };
689         Point[] moves = {new Point(1,1), new Point(0,2), new Point(2,2),
690                 new Point(1,2), new Point(0,0)};
691         TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.X;
692         boolean gameOver = true;
693
694         String scenarioName = "testX11O02X22O12X00";
695         System.out.println("\nSCENARIO: " + scenarioName + "\n");
696         totalTests += 7;
697         try {
698             printTest("testNewGame", testNewGame(gameX11O02X22O12X00()));
699             printTest("testGameOver", testGameOver(gameX11O02X22O12X00(), Result.True));
700             printTest("testGameState", testGameState(gameX11O02X22O12X00(), Result.X));
701             printTest("testGetGameGrid", testGetGameGrid(gameX11O02X22O12X00(), grid));
702             printTest("testGetMoves", testGetMoves(gameX11O02X22O12X00(), moves));
703             printTest("testChoicesX", testChoices(gameX11O02X22O12X00,
TicTacToe.BoardChoice.X, lastPlayer, gameOver, grid));
704             printTest("testChoicesO", testChoices(gameX11O02X22O12X00,
TicTacToe.BoardChoice.O, lastPlayer, gameOver, grid));
705         } catch (Exception e) {
706             System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
707             e.printStackTrace();
708         } finally {
709             if (printSectionSummaries) {
710                 printSectionSummary("Section");
711             }
712         }
```

```
713     }
714
715     //X wins - second diagonal
716     private void testX20O01X11O00X02() {
717         TicTacToe.BoardChoice[][] grid = {
718                 {TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.X},
719                 {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.X,
TicTacToe.BoardChoice.OPEN},
720                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.OPEN,
TicTacToe.BoardChoice.OPEN}
721         };
722         Point[] moves = {new Point(2,0), new Point(0,1), new Point(1,1),
723                 new Point(0,0), new Point(0,2)};
724         TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.X;
725         boolean gameOver = true;
726
727         String scenarioName = "testX20O01X11O00X02";
728         System.out.println("\nSCENARIO: " + scenarioName + "\n");
729         totalTests += 7;
730         try {
731             printTest("testNewGame", testNewGame(gameX20O01X11O00X02()));
732             printTest("testGameOver", testGameOver(gameX20O01X11O00X02(), Result.True));
733             printTest("testGameState", testGameState(gameX20O01X11O00X02(), Result.X));
734             printTest("testGetGameGrid", testGetGameGrid(gameX20O01X11O00X02(), grid));
735             printTest("testGetMoves", testGetMoves(gameX20O01X11O00X02(), moves));
736             printTest("testChoicesX", testChoices(gameX20O01X11O00X02,
TicTacToe.BoardChoice.X, lastPlayer, gameOver, grid));
737             printTest("testChoicesO", testChoices(gameX20O01X11O00X02,
TicTacToe.BoardChoice.O, lastPlayer, gameOver, grid));
738         } catch (Exception e) {
739             System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
740             e.printStackTrace();
741         } finally {
742             if (printSectionSummaries) {
743                 printSectionSummary("Section");
744             }
745         }
746     }
747
748     // X wins in 9th move
749     private void testX01O12X00O02X10O20X11O21X22() {
750         TicTacToe.BoardChoice[][] grid = {
751                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.O},
752                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.O},
753                 {TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.X}
754         };
755         Point[] moves = {new Point(0,1), new Point(1,2), new Point(0,0),
756                 new Point(0,2), new Point(1,0), new Point(2,0),
757                 new Point(1,1), new Point(2,1), new Point(2,2)};
758         TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.X;
759         boolean gameOver = true;
760
761         String scenarioName = "testX01O12X00O02X10O20X11O21X22";
762         System.out.println("\nSCENARIO: " + scenarioName + "\n");
763         totalTests += 7;
764         try {
765             printTest("testNewGame", testNewGame(gameX01O12X00O02X10O20X11O21X22()));
```

```
766             printTest("testGameOver", testGameOver(gameX01O12X00O02X10O20X11O21X22(),
    Result.True));
767             printTest("testGameState", testGameState(gameX01O12X00O02X10O20X11O21X22(),
    Result.X));
768             printTest("testGetGameGrid", testGetGameGrid(gameX01O12X00O02X10O20X11O21X22(),
    grid));
769             printTest("testGetMoves", testGetMoves(gameX01O12X00O02X10O20X11O21X22(), moves)
    );
770             printTest("testChoicesX", testChoices(gameX01O12X00O02X10O20X11O21X22,
    TicTacToe.BoardChoice.X, lastPlayer, gameOver, grid));
771             printTest("testChoicesO", testChoices(gameX01O12X00O02X10O20X11O21X22,
    TicTacToe.BoardChoice.O, lastPlayer, gameOver, grid));
772         } catch (Exception e) {
773             System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
774             e.printStackTrace();
775         } finally {
776             if (printSectionSummaries) {
777                 printSectionSummary("Section");
778             }
779         }
780     }
781
782     //XXX O wins scenarios
783
784     //O wins - first row
785     private void testX10O01X20O02X11O00() {
786         TicTacToe.BoardChoice[][] grid = {
787                 {TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.O},
788                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.X,
    TicTacToe.BoardChoice.OPEN},
789                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.OPEN,
    TicTacToe.BoardChoice.OPEN}
790         };
791         Point[] moves = {new Point(1,0), new Point(0,1), new Point(2,0),
792                 new Point(0,2), new Point(1,1), new Point(0,0)};
793         TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.O;
794         boolean gameOver = true;
795
796         String scenarioName = "testX10O01X20O02X11O00";
797         System.out.println("\nSCENARIO: " + scenarioName + "\n");
798         totalTests += 7;
799         try {
800             printTest("testNewGame", testNewGame(gameX10O01X20O02X11O00()));
801             printTest("testGameOver", testGameOver(gameX10O01X20O02X11O00(), Result.True));
802             printTest("testGameState", testGameState(gameX10O01X20O02X11O00(), Result.O));
803             printTest("testGetGameGrid", testGetGameGrid(gameX10O01X20O02X11O00(), grid));
804             printTest("testGetMoves", testGetMoves(gameX10O01X20O02X11O00(), moves));
805             printTest("testChoicesX", testChoices(gameX10O01X20O02X11O00,
    TicTacToe.BoardChoice.X, lastPlayer, gameOver, grid));
806             printTest("testChoicesO", testChoices(gameX10O01X20O02X11O00,
    TicTacToe.BoardChoice.O, lastPlayer, gameOver, grid));
807         } catch (Exception e) {
808             System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
809             e.printStackTrace();
810         } finally {
811             if (printSectionSummaries) {
812                 printSectionSummary("Section");
813             }
```

```
814         }
815     }
816
817     //O wins - second row
818     private void testX00O10X20O12X01O11() {
819         TicTacToe.BoardChoice[][] grid = {
820                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.X,
    TicTacToe.BoardChoice.OPEN},
821                 {TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.O},
822                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.OPEN,
    TicTacToe.BoardChoice.OPEN}
823         };
824         Point[] moves = {new Point(0,0), new Point(1,0), new Point(2,0),
825                 new Point(1,2), new Point(0,1), new Point(1,1)};
826         TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.O;
827         boolean gameOver = true;
828
829         String scenarioName = "testX00O10X20O12X01O11";
830         System.out.println("\nSCENARIO: " + scenarioName + "\n");
831         totalTests += 7;
832         try {
833             printTest("testNewGame", testNewGame(gameX00O10X20O12X01O11()));
834             printTest("testGameOver", testGameOver(gameX00O10X20O12X01O11(), Result.True));
835             printTest("testGameState", testGameState(gameX00O10X20O12X01O11(), Result.O));
836             printTest("testGetGameGrid", testGetGameGrid(gameX00O10X20O12X01O11(), grid));
837             printTest("testGetMoves", testGetMoves(gameX00O10X20O12X01O11(), moves));
838             printTest("testChoicesX", testChoices(gameX00O10X20O12X01O11,
    TicTacToe.BoardChoice.X, lastPlayer, gameOver, grid));
839             printTest("testChoicesO", testChoices(gameX00O10X20O12X01O11,
    TicTacToe.BoardChoice.O, lastPlayer, gameOver, grid));
840         } catch (Exception e) {
841             System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
842             e.printStackTrace();
843         } finally {
844             if (printSectionSummaries) {
845                 printSectionSummary("Section");
846             }
847         }
848     }
849
850     //O wins - third row
851     private void testX00O21X10O20X01O22() {
852         TicTacToe.BoardChoice[][] grid = {
853                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.X,
    TicTacToe.BoardChoice.OPEN},
854                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.OPEN,
    TicTacToe.BoardChoice.OPEN},
855                 {TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.O}
856         };
857         Point[] moves = {new Point(0,0), new Point(2,1), new Point(1,0),
858                 new Point(2,0), new Point(0,1), new Point(2,2)};
859         TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.O;
860         boolean gameOver = true;
861
862         String scenarioName = "testX00O21X10O20X01O22";
863         System.out.println("\nSCENARIO: " + scenarioName + "\n");
864         totalTests += 7;
```

```
865         try {
866             printTest("testNewGame", testNewGame(gameX00O21X10O20X01O22()));
867             printTest("testGameOver", testGameOver(gameX00O21X10O20X01O22(), Result.True));
868             printTest("testGameState", testGameState(gameX00O21X10O20X01O22(), Result.O));
869             printTest("testGetGameGrid", testGetGameGrid(gameX00O21X10O20X01O22(), grid));
870             printTest("testGetMoves", testGetMoves(gameX00O21X10O20X01O22(), moves));
871             printTest("testChoicesX", testChoices(gameX00O21X10O20X01O22,
TicTacToe.BoardChoice.X, lastPlayer, gameOver, grid));
872             printTest("testChoicesO", testChoices(gameX00O21X10O20X01O22,
TicTacToe.BoardChoice.O, lastPlayer, gameOver, grid));
873         } catch (Exception e) {
874             System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
875             e.printStackTrace();
876         } finally {
877             if (printSectionSummaries) {
878                 printSectionSummary("Section");
879             }
880         }
881     }
882
883     //O wins - first col
884     private void testX22O10X11O00X01O20() {
885         TicTacToe.BoardChoice[][] grid = {
886                 {TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.X,
TicTacToe.BoardChoice.OPEN},
887                 {TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.X,
TicTacToe.BoardChoice.OPEN},
888                 {TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.OPEN,
TicTacToe.BoardChoice.X}
889         };
890         Point[] moves = {new Point(2,2), new Point(1,0), new Point(1,1),
891                 new Point(0,0), new Point(0,1), new Point(2,0)};
892         TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.O;
893         boolean gameOver = true;
894
895         String scenarioName = "testX22O10X11O00X01O20";
896         System.out.println("\nSCENARIO: " + scenarioName + "\n");
897         totalTests += 7;
898         try {
899             printTest("testNewGame", testNewGame(gameX22O10X11O00X01O20()));
900             printTest("testGameOver", testGameOver(gameX22O10X11O00X01O20(), Result.True));
901             printTest("testGameState", testGameState(gameX22O10X11O00X01O20(), Result.O));
902             printTest("testGetGameGrid", testGetGameGrid(gameX22O10X11O00X01O20(), grid));
903             printTest("testGetMoves", testGetMoves(gameX22O10X11O00X01O20(), moves));
904             printTest("testChoicesX", testChoices(gameX22O10X11O00X01O20,
TicTacToe.BoardChoice.X, lastPlayer, gameOver, grid));
905             printTest("testChoicesO", testChoices(gameX22O10X11O00X01O20,
TicTacToe.BoardChoice.O, lastPlayer, gameOver, grid));
906         } catch (Exception e) {
907             System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
908             e.printStackTrace();
909         } finally {
910             if (printSectionSummaries) {
911                 printSectionSummary("Section");
912             }
913         }
914     }
915
```

```
916     //O wins - second col
917     private void testX00O01X22O11X10O21() {
918         TicTacToe.BoardChoice[][] grid = {
919                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.O,
    TicTacToe.BoardChoice.OPEN},
920                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.O,
    TicTacToe.BoardChoice.OPEN},
921                 {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.O,
    TicTacToe.BoardChoice.X}
922         };
923         Point[] moves = {new Point(0,0), new Point(0,1), new Point(2,2),
924                 new Point(1,1), new Point(1,0), new Point(2,1)};
925         TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.O;
926         boolean gameOver = true;
927 
928         String scenarioName = "testX00O01X22O11X10O21";
929         System.out.println("\nSCENARIO: " + scenarioName + "\n");
930         totalTests += 7;
931         try {
932             printTest("testNewGame", testNewGame(gameX00O01X22O11X10O21()));
933             printTest("testGameOver", testGameOver(gameX00O01X22O11X10O21(), Result.True));
934             printTest("testGameState", testGameState(gameX00O01X22O11X10O21(), Result.O));
935             printTest("testGetGameGrid", testGetGameGrid(gameX00O01X22O11X10O21(), grid));
936             printTest("testGetMoves", testGetMoves(gameX00O01X22O11X10O21(), moves));
937             printTest("testChoicesX", testChoices(gameX00O01X22O11X10O21,
    TicTacToe.BoardChoice.X, lastPlayer, gameOver, grid));
938             printTest("testChoicesO", testChoices(gameX00O01X22O11X10O21,
    TicTacToe.BoardChoice.O, lastPlayer, gameOver, grid));
939         } catch (Exception e) {
940             System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
941             e.printStackTrace();
942         } finally {
943             if (printSectionSummaries) {
944                 printSectionSummary("Section");
945             }
946         }
947     }
948 
949     //O wins - third col
950     private void testX11O02X00O22X10O12() {
951         TicTacToe.BoardChoice[][] grid = {
952                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.OPEN,
    TicTacToe.BoardChoice.O},
953                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.O},
954                 {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN,
    TicTacToe.BoardChoice.O}
955         };
956         Point[] moves = {new Point(1,1), new Point(0,2), new Point(0,0),
957                 new Point(2,2), new Point(1,0), new Point(1,2)};
958         TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.O;
959         boolean gameOver = true;
960 
961         String scenarioName = "testX11O02X00O22X10O12";
962         System.out.println("\nSCENARIO: " + scenarioName + "\n");
963         totalTests += 7;
964         try {
965             printTest("testNewGame", testNewGame(gameX11O02X00O22X10O12()));
966             printTest("testGameOver", testGameOver(gameX11O02X00O22X10O12(), Result.True));
```

```
967             printTest("testGameState", testGameState(gameX11O02X00O22X10O12(), Result.O));
968             printTest("testGetGameGrid", testGetGameGrid(gameX11O02X00O22X10O12(), grid));
969             printTest("testGetMoves", testGetMoves(gameX11O02X00O22X10O12(), moves));
970             printTest("testChoicesX", testChoices(gameX11O02X00O22X10O12,
TicTacToe.BoardChoice.X, lastPlayer, gameOver, grid));
971             printTest("testChoicesO", testChoices(gameX11O02X00O22X10O12,
TicTacToe.BoardChoice.O, lastPlayer, gameOver, grid));
972         } catch (Exception e) {
973             System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
974             e.printStackTrace();
975         } finally {
976             if (printSectionSummaries) {
977                 printSectionSummary("Section");
978             }
979         }
980     }
981
982     //O wins - first diagonal
983     private void testX01O00X10O11X21O22() {
984         TicTacToe.BoardChoice[][] grid = {
985                 {TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.X,
TicTacToe.BoardChoice.OPEN},
986                 {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.O,
TicTacToe.BoardChoice.OPEN},
987                 {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.X,
TicTacToe.BoardChoice.O}
988         };
989         Point[] moves = {new Point(0,1), new Point(0,0), new Point(1,0),
990                 new Point(1,1), new Point(2,1), new Point(2,2)};
991         TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.O;
992         boolean gameOver = true;
993
994         String scenarioName = "testX01O00X10O11X21O22";
995         System.out.println("\nSCENARIO: " + scenarioName + "\n");
996         totalTests += 7;
997         try {
998             printTest("testNewGame", testNewGame(gameX01O00X10O11X21O22()));
999             printTest("testGameOver", testGameOver(gameX01O00X10O11X21O22(), Result.True));
1000            printTest("testGameState", testGameState(gameX01O00X10O11X21O22(), Result.O));
1001            printTest("testGetGameGrid", testGetGameGrid(gameX01O00X10O11X21O22(), grid));
1002            printTest("testGetMoves", testGetMoves(gameX01O00X10O11X21O22(), moves));
1003            printTest("testChoicesX", testChoices(gameX01O00X10O11X21O22,
TicTacToe.BoardChoice.X, lastPlayer, gameOver, grid));
1004            printTest("testChoicesO", testChoices(gameX01O00X10O11X21O22,
TicTacToe.BoardChoice.O, lastPlayer, gameOver, grid));
1005        } catch (Exception e) {
1006            System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
1007            e.printStackTrace();
1008        } finally {
1009            if (printSectionSummaries) {
1010                printSectionSummary("Section");
1011            }
1012        }
1013    }
1014
1015    //O wins - second diagonal
1016    private void testX00O11X10O20X21O02() {
1017        TicTacToe.BoardChoice[][] grid = {
```

```
1018                {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.OPEN,
    TicTacToe.BoardChoice.O},
1019                {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.O,
    TicTacToe.BoardChoice.OPEN},
1020                {TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.X,
    TicTacToe.BoardChoice.OPEN}
1021        };
1022        Point[] moves = {new Point(0,0), new Point(1,1), new Point(1,0),
1023                new Point(2,0), new Point(2,1), new Point(0,2)};
1024        TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.O;
1025        boolean gameOver = true;
1026
1027        String scenarioName = "testX00O11X10O20X21O02";
1028        System.out.println("\nSCENARIO: " + scenarioName + "\n");
1029        totalTests += 7;
1030        try {
1031            printTest("testNewGame", testNewGame(gameX00O11X10O20X21O02()));
1032            printTest("testGameOver", testGameOver(gameX00O11X10O20X21O02(), Result.True));
1033            printTest("testGameState", testGameState(gameX00O11X10O20X21O02(), Result.O));
1034            printTest("testGetGameGrid", testGetGameGrid(gameX00O11X10O20X21O02(), grid));
1035            printTest("testGetMoves", testGetMoves(gameX00O11X10O20X21O02(), moves));
1036            printTest("testChoicesX", testChoices(gameX00O11X10O20X21O02,
    TicTacToe.BoardChoice.X, lastPlayer, gameOver, grid));
1037            printTest("testChoicesO", testChoices(gameX00O11X10O20X21O02,
    TicTacToe.BoardChoice.O, lastPlayer, gameOver, grid));
1038        } catch (Exception e) {
1039            System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
1040            e.printStackTrace();
1041        } finally {
1042            if (printSectionSummaries) {
1043                printSectionSummary("Section");
1044            }
1045        }
1046    }
1047
1048    // O wins in 9th move
1049    private void testO01X12O00X02O10X20O11X21O22() {
1050        TicTacToe.BoardChoice[][] grid = {
1051                {TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.X},
1052                {TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.O, TicTacToe.BoardChoice.X},
1053                {TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.X, TicTacToe.BoardChoice.O}
1054        };
1055        Point[] moves = {new Point(0,1), new Point(1,2), new Point(0,0),
1056                new Point(0,2), new Point(1,0), new Point(2,0),
1057                new Point(1,1), new Point(2,1), new Point(2,2)};
1058        TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.O;
1059        boolean gameOver = true;
1060
1061        String scenarioName = "testO01X12O00X02O10X20O11X21O22";
1062        System.out.println("\nSCENARIO: " + scenarioName + "\n");
1063        totalTests += 7;
1064        try {
1065            printTest("testNewGame", testNewGame(gameO01X12O00X02O10X20O11X21O22()));
1066            printTest("testGameOver", testGameOver(gameO01X12O00X02O10X20O11X21O22(),
    Result.True));
1067            printTest("testGameState", testGameState(gameO01X12O00X02O10X20O11X21O22(),
    Result.O));
```

```
1068            printTest("testGetGameGrid", testGetGameGrid(gameO01X12O00X02O10X20O11X21O22(), grid));
1069            printTest("testGetMoves", testGetMoves(gameO01X12O00X02O10X20O11X21O22(), moves));
1070            printTest("testChoicesX", testChoices(gameO01X12O00X02O10X20O11X21O22, TicTacToe.BoardChoice.X, lastPlayer, gameOver, grid));
1071            printTest("testChoicesO", testChoices(gameO01X12O00X02O10X20O11X21O22, TicTacToe.BoardChoice.O, lastPlayer, gameOver, grid));
1072        } catch (Exception e) {
1073            System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
1074            e.printStackTrace();
1075        } finally {
1076            if (printSectionSummaries) {
1077                printSectionSummary("Section");
1078            }
1079        }
1080
1081    }
1082
1083    ////////////////////////////
1084    // XXX Reset using New Game
1085    ////////////////////////////
1086
1087    // newGame after partial game
1088    private void testX00O10X11NewGame() {
1089        TicTacToe.BoardChoice[][] grid = {
1090                {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN},
1091                {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN},
1092                {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN}
1093        };
1094        Point[] moves = {};
1095        TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.OPEN;
1096        boolean gameOver = false;
1097
1098        String scenarioName = "testX00O10X11NewGame";
1099        System.out.println("\nSCENARIO: " + scenarioName + "\n");
1100        totalTests += 7;
1101        try {
1102            printTest("testNewGame", testNewGame(gameX00O10X11NewGame()));
1103            printTest("testGameOver", testGameOver(gameX00O10X11NewGame(), Result.False));
1104            printTest("testGameState", testGameState(gameX00O10X11NewGame(), Result.InProgress));
1105            printTest("testGetGameGrid", testGetGameGrid(gameX00O10X11NewGame(), grid));
1106            printTest("testGetMoves", testGetMoves(gameX00O10X11NewGame(), moves));
1107            printTest("testChoicesX", testChoices(gameX00O10X11NewGame, TicTacToe.BoardChoice.X, lastPlayer, gameOver, grid));
1108            printTest("testChoicesO", testChoices(gameX00O10X11NewGame, TicTacToe.BoardChoice.O, lastPlayer, gameOver, grid));
1109        } catch (Exception e) {
1110            System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
1111            e.printStackTrace();
1112        } finally {
1113            if (printSectionSummaries) {
1114                printSectionSummary("Section");
1115            }
```

```
1116            }
1117        }
1118
1119        // newGame after X wins in 9th move
1120        private void testX01O12X00O02X10O20X11O21X22NewGame() {
1121            TicTacToe.BoardChoice[][] grid = {
1122                    {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN,
    TicTacToe.BoardChoice.OPEN},
1123                    {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN,
    TicTacToe.BoardChoice.OPEN},
1124                    {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN,
    TicTacToe.BoardChoice.OPEN}
1125            };
1126            Point[] moves = {};
1127            TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.OPEN;
1128            boolean gameOver = false;
1129
1130            String scenarioName = "testX01O12X00O02X10O20X11O21X22NewGame";
1131            System.out.println("\nSCENARIO: " + scenarioName + "\n");
1132            totalTests += 7;
1133            try {
1134                printTest("testNewGame", testNewGame(gameX01O12X00O02X10O20X11O21X22NewGame()));
1135                printTest("testGameOver", testGameOver(gameX01O12X00O02X10O20X11O21X22NewGame(),
    Result.False));
1136                printTest("testGameState",
    testGameState(gameX01O12X00O02X10O20X11O21X22NewGame(), Result.InProgress));
1137                printTest("testGetGameGrid",
    testGetGameGrid(gameX01O12X00O02X10O20X11O21X22NewGame(), grid));
1138                printTest("testGetMoves", testGetMoves(gameX01O12X00O02X10O20X11O21X22NewGame(),
    moves));
1139                printTest("testChoicesX", testChoices(gameX01O12X00O02X10O20X11O21X22NewGame,
    TicTacToe.BoardChoice.X, lastPlayer, gameOver, grid));
1140                printTest("testChoicesO", testChoices(gameX01O12X00O02X10O20X11O21X22NewGame,
    TicTacToe.BoardChoice.O, lastPlayer, gameOver, grid));
1141            } catch (Exception e) {
1142                System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
1143                e.printStackTrace();
1144            } finally {
1145                if (printSectionSummaries) {
1146                    printSectionSummary("Section");
1147                }
1148            }
1149        }
1150
1151        // newGame after O wins in 9th move
1152        private void testO01X12O00X02O10X20O11X21O22NewGame() {
1153            TicTacToe.BoardChoice[][] grid = {
1154                    {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN,
    TicTacToe.BoardChoice.OPEN},
1155                    {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN,
    TicTacToe.BoardChoice.OPEN},
1156                    {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN,
    TicTacToe.BoardChoice.OPEN}
1157            };
1158            Point[] moves = {};
1159            TicTacToe.BoardChoice lastPlayer = TicTacToe.BoardChoice.OPEN;
1160            boolean gameOver = false;
1161
1162            String scenarioName = "testO01X12O00X02O10X20O11X21O22NewGame";
```

```
1163            System.out.println("\nSCENARIO: " + scenarioName + "\n");
1164            totalTests += 7;
1165            try {
1166                printTest("testNewGame", testNewGame(gameO01X12O00X02O10X20O11X21O22NewGame()));
1167                printTest("testGameOver", testGameOver(gameO01X12O00X02O10X20O11X21O22NewGame(),
    Result.False));
1168                printTest("testGameState",
    testGameState(gameO01X12O00X02O10X20O11X21O22NewGame(), Result.InProgress));
1169                printTest("testGetGameGrid",
    testGetGameGrid(gameO01X12O00X02O10X20O11X21O22NewGame(), grid));
1170                printTest("testGetMoves", testGetMoves(gameO01X12O00X02O10X20O11X21O22NewGame(),
    moves));
1171                printTest("testChoicesX", testChoices(gameO01X12O00X02O10X20O11X21O22NewGame,
    TicTacToe.BoardChoice.X, lastPlayer, gameOver, grid));
1172                printTest("testChoicesO", testChoices(gameO01X12O00X02O10X20O11X21O22NewGame,
    TicTacToe.BoardChoice.O, lastPlayer, gameOver, grid));
1173            } catch (Exception e) {
1174                System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
1175                e.printStackTrace();
1176            } finally {
1177                if (printSectionSummaries) {
1178                    printSectionSummary("Section");
1179                }
1180            }
1181
1182        }
1183
1184        ////////////////////////////
1185        // XXX Encapsulation Tests
1186        ////////////////////////////
1187
1188        private void testEncapsulation() {
1189            String scenarioName = "testEncapsulation";
1190            System.out.println("\nSCENARIO: " + scenarioName + "\n");
1191            totalTests += 2;
1192            try {
1193                printTest("testGetGameGridEncapsulation", testGetGameGridEncapsulation());
1194                printTest("testGetMovesEncapsulation", testGetMovesEncapsulation());
1195            } catch (Exception e) {
1196                System.out.printf("***UNABLE TO RUN/COMPLETE %s***\n", scenarioName + " TESTS");
1197                e.printStackTrace();
1198            } finally {
1199                if (printSectionSummaries) {
1200                    printSectionSummary("Section");
1201                }
1202            }
1203        }
1204
1205        ////////////////////////
1206        // XXX Game Builders
1207        ////////////////////////
1208
1209        /** Returns a new instance of the TicTacToe implementation to be tested.
1210         * @return a new TicTacToe
1211         */
1212        private TicTacToe newGame() {
1213            return (TicTacToe)(new TicTacToeGame());
```

```
1214        }
1215        private Scenario newGame = () -> newGame();
1216
1217        /**
1218         * @return
1219         * X--
1220         * ---
1221         * ---
1222         */
1223        private TicTacToe gameX00() {
1224            TicTacToe game = newGame();
1225            game.choose(TicTacToe.BoardChoice.X, 0, 0);
1226            return game;
1227        }
1228        private Scenario gameX00 = () -> gameX00();
1229
1230        /**
1231         * @return
1232         * X--
1233         * O--
1234         * ---
1235         */
1236        private TicTacToe gameX00O10() {
1237            TicTacToe game = gameX00();
1238            game.choose(TicTacToe.BoardChoice.O, 1, 0);
1239            return game;
1240        }
1241        private Scenario gameX00O10 = () -> gameX00O10();
1242
1243        /**
1244         * @return
1245         * X--
1246         * OX-
1247         * ---
1248         */
1249        private TicTacToe gameX00O10X11() {
1250            TicTacToe game = gameX00O10();
1251            game.choose(TicTacToe.BoardChoice.X, 1, 1);
1252            return game;
1253        }
1254        private Scenario gameX00O10X11 = () -> gameX00O10X11();
1255
1256        /** New Game resets a game in progress
1257         * @return
1258         * ---
1259         * ---
1260         * ---
1261         */
1262        private TicTacToe gameX00O10X11NewGame() {
1263            TicTacToe game = gameX00O10X11();
1264            game.newGame();
1265            return game;
1266        }
1267        private Scenario gameX00O10X11NewGame = () -> gameX00O10X11NewGame();
1268
1269        /**
```

```
1270        * @return
1271        * X--
1272        * OX-
1273        * --O
1274        */
1275       private TicTacToe gameX00O10X11O22() {
1276           TicTacToe game = gameX00O10X11();
1277           game.choose(TicTacToe.BoardChoice.O, 2, 2);
1278           return game;
1279       }
1280       private Scenario gameX00O10X11O22 = () -> gameX00O10X11O22();
1281
1282       /**
1283        * @return
1284        * X-X
1285        * OX-
1286        * --O
1287        */
1288       private TicTacToe gameX00O10X11O22X02() {
1289           TicTacToe game = gameX00O10X11O22();
1290           game.choose(TicTacToe.BoardChoice.X, 0, 2);
1291           return game;
1292       }
1293       private Scenario gameX00O10X11O22X02 = () -> gameX00O10X11O22X02();
1294
1295       /**
1296        * @return
1297        * XOX
1298        * OX-
1299        * --O
1300        */
1301       private TicTacToe gameX00O10X11O22X02O01() {
1302           TicTacToe game = gameX00O10X11O22X02();
1303           game.choose(TicTacToe.BoardChoice.O, 0, 1);
1304           return game;
1305       }
1306       private Scenario gameX00O10X11O22X02O01 = () -> gameX00O10X11O22X02O01();
1307
1308       /**
1309        * @return
1310        * XOX
1311        * OX-
1312        * -XO
1313        */
1314       private TicTacToe gameX00O10X11O22X02O01X21() {
1315           TicTacToe game = gameX00O10X11O22X02O01();
1316           game.choose(TicTacToe.BoardChoice.X, 2, 1);
1317           return game;
1318       }
1319       private Scenario gameX00O10X11O22X02O01X21 = () -> gameX00O10X11O22X02O01X21();
1320
1321       /**
1322        * @return
1323        * XOX
1324        * OX-
1325        * OXO
```

```
1326      */
1327     private TicTacToe gameX00O10X11O22X02O01X21O20() {
1328         TicTacToe game = gameX00O10X11O22X02O01X21();
1329         game.choose(TicTacToe.BoardChoice.O, 2, 0);
1330         return game;
1331     }
1332     private Scenario gameX00O10X11O22X02O01X21O20 = () -> gameX00O10X11O22X02O01X21O20();
1333 
1334     // XXX Tie game
1335 
1336     /**
1337      * @return
1338      * XOX
1339      * OXX
1340      * OXO
1341      */
1342     private TicTacToe gameX00O10X11O22X02O01X21O20X12() {
1343         TicTacToe game = gameX00O10X11O22X02O01X21O20();
1344         game.choose(TicTacToe.BoardChoice.X, 1, 2);
1345         return game;
1346     }
1347     private Scenario gameX00O10X11O22X02O01X21O20X12 = () ->
    gameX00O10X11O22X02O01X21O20X12();
1348 
1349     // XXX X wins
1350 
1351     /**
1352      * @return
1353      * X X X
1354      * O - -
1355      * - - O
1356      */
1357     private TicTacToe gameX02O10X00O22X01() {
1358         TicTacToe game = newGame();
1359         game.choose(TicTacToe.BoardChoice.X, 0, 2);
1360         game.choose(TicTacToe.BoardChoice.O, 1, 0);
1361         game.choose(TicTacToe.BoardChoice.X, 0, 0);
1362         game.choose(TicTacToe.BoardChoice.O, 2, 2);
1363         game.choose(TicTacToe.BoardChoice.X, 0, 1);
1364         return game;
1365     }
1366     private Scenario gameX02O10X00O22X01 = () -> gameX02O10X00O22X01();
1367 
1368     /**
1369      * @return
1370      * O - -
1371      * X X X
1372      * - - O
1373      */
1374     private TicTacToe gameX11O00X10O22X12() {
1375         TicTacToe game = newGame();
1376         game.choose(TicTacToe.BoardChoice.X, 1, 1);
1377         game.choose(TicTacToe.BoardChoice.O, 0, 0);
1378         game.choose(TicTacToe.BoardChoice.X, 1, 0);
1379         game.choose(TicTacToe.BoardChoice.O, 2, 2);
1380         game.choose(TicTacToe.BoardChoice.X, 1, 2);
```

```
1381            return game;
1382        }
1383        private Scenario gameX11O00X10O22X12 = () -> gameX11O00X10O22X12();
1384
1385        /**
1386         * @return
1387         * O - -
1388         * - - O
1389         * X X X
1390         */
1391        private TicTacToe gameX22O00X20O12X21() {
1392            TicTacToe game = newGame();
1393            game.choose(TicTacToe.BoardChoice.X, 2, 2);
1394            game.choose(TicTacToe.BoardChoice.O, 0, 0);
1395            game.choose(TicTacToe.BoardChoice.X, 2, 0);
1396            game.choose(TicTacToe.BoardChoice.O, 1, 2);
1397            game.choose(TicTacToe.BoardChoice.X, 2, 1);
1398            return game;
1399        }
1400        private Scenario gameX22O00X20O12X21 = () -> gameX22O00X20O12X21();
1401
1402        /**
1403         * @return
1404         * X--
1405         * XOO
1406         * X--
1407         */
1408        private TicTacToe gameX00O11X20O12X10() {
1409            TicTacToe game = newGame();
1410            game.choose(TicTacToe.BoardChoice.X, 0, 0);
1411            game.choose(TicTacToe.BoardChoice.O, 1, 1);
1412            game.choose(TicTacToe.BoardChoice.X, 2, 0);
1413            game.choose(TicTacToe.BoardChoice.O, 1, 2);
1414            game.choose(TicTacToe.BoardChoice.X, 1, 0);
1415            return game;
1416        }
1417        private Scenario gameX00O11X20O12X10 = () -> gameX00O11X20O12X10();
1418
1419        /**
1420         * @return
1421         * -X-
1422         * OXO
1423         * -X-
1424         */
1425        private TicTacToe gameX01O10X21O12X11() {
1426            TicTacToe game = newGame();
1427            game.choose(TicTacToe.BoardChoice.X, 0, 1);
1428            game.choose(TicTacToe.BoardChoice.O, 1, 0);
1429            game.choose(TicTacToe.BoardChoice.X, 2, 1);
1430            game.choose(TicTacToe.BoardChoice.O, 1, 2);
1431            game.choose(TicTacToe.BoardChoice.X, 1, 1);
1432            return game;
1433        }
1434        private Scenario gameX01O10X21O12X11 = () -> gameX01O10X21O12X11();
1435
1436        /**
```

```
1437        * @return
1438        * --X
1439        * -OX
1440        * -OX
1441        */
1442       private TicTacToe gameX12O11X02O21X22() {
1443           TicTacToe game = newGame();
1444           game.choose(TicTacToe.BoardChoice.X, 1, 2);
1445           game.choose(TicTacToe.BoardChoice.O, 1, 1);
1446           game.choose(TicTacToe.BoardChoice.X, 0, 2);
1447           game.choose(TicTacToe.BoardChoice.O, 2, 1);
1448           game.choose(TicTacToe.BoardChoice.X, 2, 2);
1449           return game;
1450       }
1451       private Scenario gameX12O11X02O21X22 = () -> gameX12O11X02O21X22();
1452
1453       /**
1454        * @return
1455        * X-O
1456        * -XO
1457        * --X
1458        */
1459       private TicTacToe gameX11O02X22O12X00() {
1460           TicTacToe game = newGame();
1461           game.choose(TicTacToe.BoardChoice.X, 1, 1);
1462           game.choose(TicTacToe.BoardChoice.O, 0, 2);
1463           game.choose(TicTacToe.BoardChoice.X, 2, 2);
1464           game.choose(TicTacToe.BoardChoice.O, 1, 2);
1465           game.choose(TicTacToe.BoardChoice.X, 0, 0);
1466           return game;
1467       }
1468       private Scenario gameX11O02X22O12X00 = () -> gameX11O02X22O12X00();
1469
1470       /**
1471        * @return
1472        * OOX
1473        * -X-
1474        * X--
1475        */
1476       private TicTacToe gameX20O01X11O00X02() {
1477           TicTacToe game = newGame();
1478           game.choose(TicTacToe.BoardChoice.X, 2, 0);
1479           game.choose(TicTacToe.BoardChoice.O, 0, 1);
1480           game.choose(TicTacToe.BoardChoice.X, 1, 1);
1481           game.choose(TicTacToe.BoardChoice.O, 0, 0);
1482           game.choose(TicTacToe.BoardChoice.X, 0, 2);
1483           return game;
1484       }
1485       private Scenario gameX20O01X11O00X02 = () -> gameX20O01X11O00X02();
1486
1487       /**
1488        * @return
1489        * XXO
1490        * XXO
1491        * OOX
1492        */
```

```
1493      private TicTacToe gameX01O12X00O02X10O20X11O21X22() {
1494          TicTacToe game = newGame();
1495          game.choose(TicTacToe.BoardChoice.X, 0, 1);
1496          game.choose(TicTacToe.BoardChoice.O, 1, 2);
1497          game.choose(TicTacToe.BoardChoice.X, 0, 0);
1498          game.choose(TicTacToe.BoardChoice.O, 0, 2);
1499          game.choose(TicTacToe.BoardChoice.X, 1, 0);
1500          game.choose(TicTacToe.BoardChoice.O, 2, 0);
1501          game.choose(TicTacToe.BoardChoice.X, 1, 1);
1502          game.choose(TicTacToe.BoardChoice.O, 2, 1);
1503          game.choose(TicTacToe.BoardChoice.X, 2, 2);
1504          return game;
1505      }
1506      private Scenario gameX01O12X00O02X10O20X11O21X22 = () ->
      gameX01O12X00O02X10O20X11O21X22();
1507
1508      /** New Game after X wins
1509       * @return
1510       * ---
1511       * ---
1512       * ---
1513       */
1514      private TicTacToe gameX01O12X00O02X10O20X11O21X22NewGame() {
1515          TicTacToe game = newGame();
1516          game.choose(TicTacToe.BoardChoice.X, 0, 1);
1517          game.choose(TicTacToe.BoardChoice.O, 1, 2);
1518          game.choose(TicTacToe.BoardChoice.X, 0, 0);
1519          game.choose(TicTacToe.BoardChoice.O, 0, 2);
1520          game.choose(TicTacToe.BoardChoice.X, 1, 0);
1521          game.choose(TicTacToe.BoardChoice.O, 2, 0);
1522          game.choose(TicTacToe.BoardChoice.X, 1, 1);
1523          game.choose(TicTacToe.BoardChoice.O, 2, 1);
1524          game.choose(TicTacToe.BoardChoice.X, 2, 2);
1525          game.newGame();
1526          return game;
1527      }
1528      private Scenario gameX01O12X00O02X10O20X11O21X22NewGame = () ->
      gameX01O12X00O02X10O20X11O21X22NewGame();
1529
1530      // XXX O wins
1531
1532      /**
1533       * @return
1534       * OOO
1535       * XX-
1536       * X--
1537       */
1538      private TicTacToe gameX10O01X20O02X11O00() {
1539          TicTacToe game = newGame();
1540          game.choose(TicTacToe.BoardChoice.X, 1, 0);
1541          game.choose(TicTacToe.BoardChoice.O, 0, 1);
1542          game.choose(TicTacToe.BoardChoice.X, 2, 0);
1543          game.choose(TicTacToe.BoardChoice.O, 0, 2);
1544          game.choose(TicTacToe.BoardChoice.X, 1, 1);
1545          game.choose(TicTacToe.BoardChoice.O, 0, 0);
1546          return game;
1547      }
```

```
1548      private Scenario gameX10O01X20O02X11O00 = () -> gameX10O01X20O02X11O00();
1549
1550      /**
1551       * @return
1552       * XX-
1553       * OOO
1554       * X--
1555       */
1556      private TicTacToe gameX00O10X20O12X01O11() {
1557          TicTacToe game = newGame();
1558          game.choose(TicTacToe.BoardChoice.X, 0, 0);
1559          game.choose(TicTacToe.BoardChoice.O, 1, 0);
1560          game.choose(TicTacToe.BoardChoice.X, 2, 0);
1561          game.choose(TicTacToe.BoardChoice.O, 1, 2);
1562          game.choose(TicTacToe.BoardChoice.X, 0, 1);
1563          game.choose(TicTacToe.BoardChoice.O, 1, 1);
1564          return game;
1565      }
1566      private Scenario gameX00O10X20O12X01O11 = () -> gameX00O10X20O12X01O11();
1567
1568      /**
1569       * @return
1570       * XX-
1571       * X--
1572       * OOO
1573       */
1574      private TicTacToe gameX00O21X10O20X01O22() {
1575          TicTacToe game = newGame();
1576          game.choose(TicTacToe.BoardChoice.X, 0, 0);
1577          game.choose(TicTacToe.BoardChoice.O, 2, 1);
1578          game.choose(TicTacToe.BoardChoice.X, 1, 0);
1579          game.choose(TicTacToe.BoardChoice.O, 2, 0);
1580          game.choose(TicTacToe.BoardChoice.X, 0, 1);
1581          game.choose(TicTacToe.BoardChoice.O, 2, 2);
1582          return game;
1583      }
1584      private Scenario gameX00O21X10O20X01O22 = () -> gameX00O21X10O20X01O22();
1585
1586      /**
1587       * @return
1588       * OX-
1589       * OX-
1590       * O-X
1591       */
1592      private TicTacToe gameX22O10X11O00X01O20() {
1593          TicTacToe game = newGame();
1594          game.choose(TicTacToe.BoardChoice.X, 2, 2);
1595          game.choose(TicTacToe.BoardChoice.O, 1, 0);
1596          game.choose(TicTacToe.BoardChoice.X, 1, 1);
1597          game.choose(TicTacToe.BoardChoice.O, 0, 0);
1598          game.choose(TicTacToe.BoardChoice.X, 0, 1);
1599          game.choose(TicTacToe.BoardChoice.O, 2, 0);
1600          return game;
1601      }
1602      private Scenario gameX22O10X11O00X01O20 = () -> gameX22O10X11O00X01O20();
1603
```

```
1604      /**
1605       * @return
1606       * XO-
1607       * XO-
1608       * -OX
1609       */
1610      private TicTacToe gameX00O01X22O11X10O21() {
1611          TicTacToe game = newGame();
1612          game.choose(TicTacToe.BoardChoice.X, 0, 0);
1613          game.choose(TicTacToe.BoardChoice.O, 0, 1);
1614          game.choose(TicTacToe.BoardChoice.X, 2, 2);
1615          game.choose(TicTacToe.BoardChoice.O, 1, 1);
1616          game.choose(TicTacToe.BoardChoice.X, 1, 0);
1617          game.choose(TicTacToe.BoardChoice.O, 2, 1);
1618          return game;
1619      }
1620      private Scenario gameX00O01X22O11X10O21 = () -> gameX00O01X22O11X10O21();
1621
1622      /**
1623       * @return
1624       * X-O
1625       * XXO
1626       * --O
1627       */
1628      private TicTacToe gameX11O02X00O22X10O12() {
1629          TicTacToe game = newGame();
1630          game.choose(TicTacToe.BoardChoice.X, 1, 1);
1631          game.choose(TicTacToe.BoardChoice.O, 0, 2);
1632          game.choose(TicTacToe.BoardChoice.X, 0, 0);
1633          game.choose(TicTacToe.BoardChoice.O, 2, 2);
1634          game.choose(TicTacToe.BoardChoice.X, 1, 0);
1635          game.choose(TicTacToe.BoardChoice.O, 1, 2);
1636          return game;
1637      }
1638      private Scenario gameX11O02X00O22X10O12 = () -> gameX11O02X00O22X10O12();
1639
1640      /**
1641       * @return
1642       * OX-
1643       * XO-
1644       * -XO
1645       */
1646      private TicTacToe gameX01O00X10O11X21O22() {
1647          TicTacToe game = newGame();
1648          game.choose(TicTacToe.BoardChoice.X, 0, 1);
1649          game.choose(TicTacToe.BoardChoice.O, 0, 0);
1650          game.choose(TicTacToe.BoardChoice.X, 1, 0);
1651          game.choose(TicTacToe.BoardChoice.O, 1, 1);
1652          game.choose(TicTacToe.BoardChoice.X, 2, 1);
1653          game.choose(TicTacToe.BoardChoice.O, 2, 2);
1654          return game;
1655      }
1656      private Scenario gameX01O00X10O11X21O22 = () -> gameX01O00X10O11X21O22();
1657
1658      /**
1659       * @return
```

```
1660        * X-O
1661        * XO-
1662        * OX-
1663        */
1664       private TicTacToe gameX00O11X10O20X21O02() {
1665           TicTacToe game = newGame();
1666           game.choose(TicTacToe.BoardChoice.X, 0, 0);
1667           game.choose(TicTacToe.BoardChoice.O, 1, 1);
1668           game.choose(TicTacToe.BoardChoice.X, 1, 0);
1669           game.choose(TicTacToe.BoardChoice.O, 2, 0);
1670           game.choose(TicTacToe.BoardChoice.X, 2, 1);
1671           game.choose(TicTacToe.BoardChoice.O, 0, 2);
1672           return game;
1673       }
1674       private Scenario gameX00O11X10O20X21O02 = () -> gameX00O11X10O20X21O02();
1675
1676       /**
1677        * @return
1678        * OOX
1679        * OOX
1680        * XXO
1681        */
1682       private TicTacToe gameO01X12O00X02O10X20O11X21O22() {
1683           TicTacToe game = newGame();
1684           game.choose(TicTacToe.BoardChoice.O, 0, 1);
1685           game.choose(TicTacToe.BoardChoice.X, 1, 2);
1686           game.choose(TicTacToe.BoardChoice.O, 0, 0);
1687           game.choose(TicTacToe.BoardChoice.X, 0, 2);
1688           game.choose(TicTacToe.BoardChoice.O, 1, 0);
1689           game.choose(TicTacToe.BoardChoice.X, 2, 0);
1690           game.choose(TicTacToe.BoardChoice.O, 1, 1);
1691           game.choose(TicTacToe.BoardChoice.X, 2, 1);
1692           game.choose(TicTacToe.BoardChoice.O, 2, 2);
1693           return game;
1694       }
1695       private Scenario gameO01X12O00X02O10X20O11X21O22 = () ->
     gameO01X12O00X02O10X20O11X21O22();
1696
1697       /**
1698        * @return
1699        * OOX
1700        * OOX
1701        * XXO
1702        */
1703       private TicTacToe gameO01X12O00X02O10X20O11X21O22NewGame() {
1704           TicTacToe game = newGame();
1705           game.choose(TicTacToe.BoardChoice.O, 0, 1);
1706           game.choose(TicTacToe.BoardChoice.X, 1, 2);
1707           game.choose(TicTacToe.BoardChoice.O, 0, 0);
1708           game.choose(TicTacToe.BoardChoice.X, 0, 2);
1709           game.choose(TicTacToe.BoardChoice.O, 1, 0);
1710           game.choose(TicTacToe.BoardChoice.X, 2, 0);
1711           game.choose(TicTacToe.BoardChoice.O, 1, 1);
1712           game.choose(TicTacToe.BoardChoice.X, 2, 1);
1713           game.choose(TicTacToe.BoardChoice.O, 2, 2);
1714           game.newGame();
```

```
1715            return game;
1716        }
1717        private Scenario gameO01X12O00X02O10X20O11X21O22NewGame = () ->
    gameO01X12O00X02O10X20O11X21O22NewGame();
1718
1719        // ////////////////////////
1720        // XXX TEST METHODS
1721        // ////////////////////////
1722
1723        /**
1724         * Runs newGame() method of a TicTacToe. No exceptions expected.
1725         * @return test success
1726         */
1727        private boolean testNewGame(TicTacToe game) {
1728            boolean success = true;
1729            try {
1730                game.newGame();
1731            } catch (Exception e) {
1732                System.out.printf("%s caught unexpected %s\n", "testNewGame", e.toString());
1733                e.printStackTrace();
1734                success = false;
1735            }
1736            return success;
1737        }
1738
1739        /**
1740         * Runs gameOver() method on a newly created TicTacToe.
1741         * @return test success
1742         */
1743        private boolean testGameOver(TicTacToe game, Result expectedResult) {
1744            Result result;
1745            try {
1746                if (game.gameOver()) {
1747                    result = Result.True;
1748                } else {
1749                    result = Result.False;
1750                }
1751            } catch (Exception e) {
1752                System.out.printf("%s caught unexpected %s\n", "testGameOver", e.toString());
1753                e.printStackTrace();
1754                result = Result.UnexpectedException;
1755            }
1756            return result == expectedResult;
1757        }
1758
1759        /**
1760         * Runs getGameState() method on a TicTacToe.
1761         * @return test success
1762         */
1763        private boolean testGameState(TicTacToe game, Result expectedResult) {
1764            Result result;
1765            try {
1766                if (game.getGameState() == TicTacToe.GameState.IN_PROGRESS) {
1767                    result = Result.InProgress;
1768                } else if (game.getGameState() == TicTacToe.GameState.TIE) {
1769                    result = Result.Tie;
```

```
1770            } else if (game.getGameState() == TicTacToe.GameState.X_WON) {
1771                result = Result.X;
1772            } else if (game.getGameState() == TicTacToe.GameState.O_WON) {
1773                result = Result.O;
1774            } else {
1775                result = Result.Fail;
1776            }
1777        } catch (Exception e) {
1778            System.out.printf("%s caught unexpected %s\n", "testGameState", e.toString());
1779            e.printStackTrace();
1780            result = Result.UnexpectedException;
1781        }
1782        return result == expectedResult;
1783    }
1784
1785    /**
1786     * Runs getGameGrid() method on a TicTacToe.
1787     * @return test success
1788     */
1789    private boolean testGetGameGrid(TicTacToe game, TicTacToe.BoardChoice[][] expectedGrid)
    {
1790        Result result;
1791        try {
1792            TicTacToe.BoardChoice[][] returnedGrid = game.getGameGrid();
1793            if (equivalentArrays(returnedGrid, expectedGrid)) {
1794                result = Result.Pass;
1795            } else {
1796                result = Result.Fail;
1797            }
1798        } catch (Exception e) {
1799            System.out.printf("%s caught unexpected %s\n", "testGetGameGrid", e.toString());
1800            e.printStackTrace();
1801            result = Result.UnexpectedException;
1802        }
1803        return result == Result.Pass;
1804    }
1805
1806    /**
1807     * Runs getMoves() method on a TicTacToe.
1808     * @return test success
1809     */
1810    private boolean testGetMoves(TicTacToe game, Point[] expectedMoves) {
1811        Result result;
1812        try {
1813            Point[] returnedMoves = game.getMoves();
1814            if (equivalentArrays(returnedMoves, expectedMoves)) {
1815                result = Result.Pass;
1816            } else {
1817                result = Result.Fail;
1818            }
1819        } catch (Exception e) {
1820            System.out.printf("%s caught unexpected %s\n", "testGetMoves", e.toString());
1821            e.printStackTrace();
1822            result = Result.UnexpectedException;
1823        }
1824        return result == Result.Pass;
```

```
1825     }
1826
1827     /**
1828      * Runs tests on choose() method on a TicTacToe for every position.
1829      * @return test success
1830      */
1831     private boolean testChoices(Scenario gameBuilder, TicTacToe.BoardChoice player,
TicTacToe.BoardChoice lastPlayer,
1832             boolean gameOver, TicTacToe.BoardChoice[][] grid) {
1833         boolean result = true;
1834         try {
1835             if (player == lastPlayer || gameOver) { //no positions are valid
1836                 for (int row = 0; row < grid.length; row++) {
1837                     for (int col = 0; col < grid[row].length; col++) {
1838                         TicTacToe game = gameBuilder.setUpGame();
1839                         if (game.choose(player, row, col)) {
1840                             System.out.printf("\tchoose(%s, %d, %d) returned true, expected
false\n", player, row, col);
1841                             result = false;
1842                         }
1843                     }
1844                 }
1845             } else { //only open positions are valid
1846                 for (int row = 0; row < grid.length; row++) {
1847                     for (int col = 0; col < grid[row].length; col++) {
1848                         TicTacToe game = gameBuilder.setUpGame();
1849                         if (grid[row][col] == TicTacToe.BoardChoice.OPEN) {
1850                             if (game.choose(player, row, col) == false) {
1851                                 System.out.printf("\tchoose(%s, %d, %d) returned false,
expected true\n", player, row, col);
1852                                 result = false;
1853                             }
1854                         } else if (game.choose(player, row, col) == true) {
1855                             System.out.printf("\tchoose(%s, %d, %d) returned true, expected
false\n", player, row, col);
1856                             result = false;
1857                         }
1858                     }
1859                 }
1860             }
1861         } catch (Exception e) {
1862             System.out.printf("%s caught unexpected %s\n", "testChoices", e.toString());
1863             e.printStackTrace();
1864             result = false;
1865         }
1866         return result;
1867     }
1868
1869 //  /**
1870 //   * Runs choose() method on a TicTacToe. Replaced by superior testChoices() tests.
1871 //   * @return test success
1872 //   */
1873 //  private boolean testChoose(TicTacToe game, TicTacToe.Player player, int row, int col,
Result expectedResult) {
1874 //      Result result;
1875 //      try {
1876 //          if (game.choose(player, row, col)) {
```

```
1877 //              result = Result.True;
1878 //          } else {
1879 //              result = Result.False;
1880 //          }
1881 //      } catch (Exception e) {
1882 //          System.out.printf("%s caught unexpected %s\n", "testChoose", e.toString());
1883 //          e.printStackTrace();
1884 //          result = Result.UnexpectedException;
1885 //      }
1886 //      return result == expectedResult;
1887 //  }
1888 
1889     /** Confirm that the getGameGrid() method does not return a reference to instance data
1890      * @return true if encapsulation has been preserved, else false
1891      */
1892     private boolean testGetGameGridEncapsulation() {
1893         TicTacToe.BoardChoice[][] grid = {
1894                 {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN,
TicTacToe.BoardChoice.OPEN},
1895                 {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN,
TicTacToe.BoardChoice.OPEN},
1896                 {TicTacToe.BoardChoice.OPEN, TicTacToe.BoardChoice.OPEN,
TicTacToe.BoardChoice.OPEN}
1897         };
1898         boolean passed = false;
1899         try {
1900             TicTacToe game = newGame();
1901             game.getGameGrid()[0][0] = TicTacToe.BoardChoice.X;
1902             passed = equivalentArrays(grid, game.getGameGrid());
1903         } catch (Exception e) {
1904             System.out.printf("%s caught unexpected %s\n", "testGetGameGridEncapsulation",
e.toString());
1905             e.printStackTrace();
1906         }
1907         return passed;
1908     }
1909 
1910     /** Confirm that the getMoves() method does not return a reference to instance data
1911      * @return true if encapsulation has been preserved, else false
1912      */
1913     private boolean testGetMovesEncapsulation() {
1914         Point[] gameMoves = {new Point(0,0)};
1915         boolean passed = false;
1916         try {
1917             TicTacToe game = gameX00();
1918             game.getMoves()[0] = new Point(1,2);
1919             passed = equivalentArrays(gameMoves, game.getMoves());
1920         } catch (Exception e) {
1921             System.out.printf("%s caught unexpected %s\n", "testGetMovesEncapsulation",
e.toString());
1922             e.printStackTrace();
1923         }
1924         return passed;
1925     }
1926 
1927     ////////////////////////
1928     // XXX HELPER METHODS
```

```
1929      //////////////////////////
1930
1931      /** Compare two two-dimensional double arrays for equivalence.
1932       * @param a1 first Player[][]
1933       * @param a2 second Player[][]
1934       * @return true if all values in a1 and a2 are the same, else false
1935       */
1936      private boolean equivalentArrays(TicTacToe.BoardChoice[][] a1, TicTacToe.BoardChoice[][]
      a2) {
1937          boolean equivalent = true;
1938          if (a1.length != a2.length || (a1.length > 1 && a1[0].length != a2[0].length)) {
1939              equivalent = false;
1940          } else {
1941              for (int row = 0; row < a1.length; row++) {
1942                  for (int col = 0; col < a1[0].length; col++) {
1943                      if (row >= a2.length || a1[row].length != a2[row].length) {
1944                          equivalent = false;
1945                      } else {
1946                          if (a1[row][col] != a2[row][col]) {
1947                              equivalent = false;
1948                          }
1949                      }
1950                  }
1951              }
1952          }
1953          return equivalent;
1954      }
1955
1956      /** Compare two one-dimensional Point arrays for equivalence.
1957       * @param a1 first Point[]
1958       * @param a2 second Point[]
1959       * @return true if all values in a1 and a2 are the same, else false
1960       */
1961      private boolean equivalentArrays(Point[] a1, Point[] a2) {
1962          boolean equivalent = true;
1963          if (a1.length != a2.length) {
1964              equivalent = false;
1965          } else {
1966              for (int row = 0; row < a1.length; row++) {
1967                  if (!a1[row].equals(a2[row])) {
1968                      equivalent = false;
1969                  }
1970              }
1971          }
1972          return equivalent;
1973      }
1974
1975      /** Interface for builder method Lambda references used above */
1976      private interface Scenario {
1977          TicTacToe setUpGame();
1978      }
1979
1980  }// end class TicTacToeTester
1981
```