```java
package package1;

import java.util.Random;

/**
 * Created by jon on 1/27/14.
 */
public class MineSweeperGame {
    //instance variables
    private Cell[][] board;
    private GameStatus status;
    private int size = 10;
    private int totalMineCount = 10;
    private int flaggedMineCount = 0;

    /**
     * constructor for MineSweeperGame
     * @param pSize width of a side
     * @param pMines number of mines
     */
    public MineSweeperGame(int pSize, int pMines) {
        size = pSize;
        totalMineCount = pMines;
        status = GameStatus.NOT_OVER_YET;
        board = new Cell[size][size];
        for (int row = 0; row < size; row++)
            for (int col = 0; col < size; col++)
                board[row][col] = new Cell();
        Random random = new Random();
        int mineCount = 0;
        while (mineCount < totalMineCount) {
            int col = random.nextInt(size);
            int row = random.nextInt(size);

            if (!board[row][col].isMine()) {
                board[row][col].setMine(true);
                mineCount++;
            }
        }
        for (int row = 0; row < size; row++)
            for (int col = 0; col < size; col++)
                board[row][col].setMineCount(getAdjacentMintCount(row, co
l));
    }

    /**
     * selects the cell
     * @param row row to select
     * @param col col to select
     */
    public void select(int row, int col) {
        if (board[row][col].isExposed()) {
            return;
        }
        if (board[row][col].isMine()) {
            status = GameStatus.LOST;
            return;
        }
        if (!board[row][col].isFlagged()) {
            board[row][col].setExposed(true);
            //recursive selection for surrounding empty cells
            if (board[row][col].getMineCount() == 0) {
                try {
                    if (!board[row][col - 1].isMine()) select(row, col -
                } catch (ArrayIndexOutOfBoundsException e) {
                }
                try {
```

```
67                      if (!board[row - 1][col].isMine()) select(row - 1, c
   ol);
68                  } catch (ArrayIndexOutOfBoundsException e) {
69                  }
70                  try {
71                      if (!board[row + 1][col].isMine()) select(row + 1, c
   ol);
72                  } catch (ArrayIndexOutOfBoundsException e) {
73                  }
74                  try {
75                      if (!board[row][col + 1].isMine()) select(row, col +
76                  } catch (ArrayIndexOutOfBoundsException e) {
77                  }
78              }
79          }
80      }
81
82      /**
83       * set cell as flagged if user thinks it's a mine
84       * @param row row to flag
85       * @param col col to flag
86       */
87      public void flag(int row, int col) {
88          if (board[row][col].isFlagged()) {
89              board[row][col].setFlagged(false);
90              if (board[row][col].isMine()) {
91                  flaggedMineCount--;
92              }
93              return;
94          }
95          if (board[row][col].isExposed()) {
96              return;
97          }
98
99          board[row][col].setFlagged(true);
100         if (board[row][col].isMine()) {
101             flaggedMineCount++;
102         }
103         //this is how you win
104         if (flaggedMineCount == totalMineCount) {
105             status = GameStatus.WON;
106         }
107     }
108
109     /**
110      * gets the game status
111      * @return game status
112      */
113     public GameStatus getGameStatus() {
114         return status;
115     }
116
117     /**
118      * returns the cell requested
119      * @param row row to return
120      * @param col col to return
121      * @return Cell requested
122      */
123     public Cell getCell(int row, int col) {
124         return board[row][col];
125     }
126
127     /**
128      *
129      * @param r row to query
130      * @param c col to query
131      * @return adjacent mine count
```

```
132       */
133      public int getAdjacentMintCount(int r, int c) {
134          //ArrayIndexOutOfBounds exceptions are just ignored to handle
     edge cells
135          int count = 0;
136          try {
137              if (board[r][c - 1].isMine()) {
138                  count++;
139              }
140          } catch (ArrayIndexOutOfBoundsException e) {
141
142          }
143          try {
144              if (board[r - 1][c].isMine()) {
145                  count++;
146              }
147          } catch (ArrayIndexOutOfBoundsException e) {
148
149          }
150          try {
151              if (board[r - 1][c - 1].isMine()) {
152                  count++;
153              }
154          } catch (ArrayIndexOutOfBoundsException e) {
155
156          }
157          try {
158              if (board[r + 1][c + 1].isMine()) {
159                  count++;
160              }
161
162          } catch (ArrayIndexOutOfBoundsException e) {
163
164          }
165          try {
166              if (board[r + 1][c - 1].isMine()) {
167                  count++;
168              }
169          } catch (ArrayIndexOutOfBoundsException e) {
170
171          }
172          try {
173              if (board[r - 1][c + 1].isMine()) {
174                  count++;
175              }
176          } catch (ArrayIndexOutOfBoundsException e) {
177
178          }
179          try {
180              if (board[r + 1][c].isMine()) {
181                  count++;
182              }
183          } catch (ArrayIndexOutOfBoundsException e) {
184
185          }
186          try {
187              if (board[r][c + 1].isMine()) {
188                  count++;
189              }
190          } catch (ArrayIndexOutOfBoundsException e) {
191
192          }
193          return count;
194      }
195  }
196
197
```