```
1   //Name -
2   //Date -
3   //Class -
4
5   import java.util.Scanner;
6   import static java.lang.System.*;
7
8   public class KittyMap
9   {
10      private boolean[][] kittyGrid;
11      private int[][] kittyCountsGrid;
12
13      /*
14       *this constructor instantiates the kittyGrid and kittyCountsGrid
15       *then loads in random true and false values in kittyGrid
16       *then it sets the kittyCountsGrid using the setKittyCountsGrid method
17       */
18      public KittyMap(int rows, int cols)
19      {
20          kittyGrid = new boolean[rows][cols];
21          kittyCountsGrid = new int[rows][cols];
22
23          for( int o = 0; o < rows; o++ )
24              for( int i = 0; i < cols; i++ )
25                  if( (int)( Math.random() * 5 ) == 0 )
26                      kittyGrid[o][i] = true;
27                  else
28                      kittyGrid[o][i] = false;
29          setKittyCountsGrid();
30      }
31
32      /*
33       *this method will call setKittyCountsGrid
34       *to retrieve the kitty counts for the kittyGrid
35       *then display the kittyCountsGrid
36       */
37      public void printKittyCount()
38      {
39          for( int o = 0; o < kittyGrid.length; o++ ) {
40              for( int i = 0; i < kittyGrid[o].length; i++ )
41                  out.print( getKittyCount(o, i) + " " );
42              out.print("\n");
43          }
44      }
45
46      /*
47       *this method will set the kitty counts for each cell
48       *by calling getKittyCount for each cell
49       */
50      private void setKittyCountsGrid()
51      {
52          for( int o = 0; o < kittyGrid.length; o++ )
53              for( int i = 0; i < kittyGrid[o].length; i++ )
54                  kittyCountsGrid[o][i] = getKittyCount( o, i );
55      }
56
57      /*
58       *this method will return the value of the current cell
59       *if this cell contains a kitty - return 9
60       *otherwise - return the count of all kitties in adjacent cells
61       *reminder:: check to see if the cell is inBounds
62       */
63      public int getKittyCount( int r, int c)
64      {
65          if( !inBounds(r, c) )
66              return 0;
```

```
67
68          if( kittyGrid[r][c] )
69              return 9;
70
71          if( !( c == kittyGrid[0].length - 1 || r == kittyGrid.length - 1 || r == 0 || c
            == 0 ) ) { // Checks to see if its not on the outside edge of the grid
72              int ret = 0;
73
74              ret += bToI(kittyGrid[r - 1][c - 1])    +
75                     bToI(kittyGrid[r - 1][c])        +
76                     bToI(kittyGrid[r - 1][c + 1])    +
77
78                     bToI(kittyGrid[r][c + 1])        +
79                     bToI(kittyGrid[r][c - 1])        +
80
81                     bToI(kittyGrid[r + 1][c - 1])    +
82                     bToI(kittyGrid[r + 1][c])        +
83                     bToI(kittyGrid[r + 1][c + 1])    ;
84              return ret;
85          }
86
87          // will run if its on the outside edge of the grid
88          int ret = 0;
89
90          if( c < kittyGrid[0].length - 1 )
91              ret += bToI(kittyGrid[r][c + 1]) ;
92          if( c > 0 )
93              ret += bToI(kittyGrid[r][c - 1]) ;
94          if( r > 0 )
95              ret += bToI(kittyGrid[r - 1][c]) ;
96          if( r < kittyGrid.length - 1 )
97              ret += bToI(kittyGrid[r + 1][c]) ;
98
99          if( r != 0 && c != 0 )
100             ret += bToI(kittyGrid[r - 1][c - 1]) ;
101         if( r != kittyGrid.length - 1 && c != kittyGrid[0].length - 1 )
102             ret += bToI(kittyGrid[r + 1][c + 1]) ;
103         if( r != 0 && c != kittyGrid[0].length - 1 )
104             ret += bToI(kittyGrid[r - 1][c + 1]) ;
105         if( r != kittyGrid.length - 1 && c != 0 )
106             ret += bToI(kittyGrid[r + 1][c - 1]) ;
107
108         return ret;
109     }
110
111     /*
112      *check r and c to ensure they are inside the grid
113      */
114     private boolean inBounds( int row, int col)
115     {
116         return (row < kittyGrid.length) && (row>=0) && (col <kittyGrid[0].length) && (
            col>=0);
117     }
118
119     /*
120      *return the kittyGrid as a string of locations
121      */
122     public String toString()
123     {
124         String output="Kitty locations:\n";
125         for( int r=0; r<kittyGrid.length; r++)
126         {
127             for(int c=0; c<kittyGrid[r].length; c++)
128             {
129                 if(kittyGrid[r][c])
130                     output+= "["+r+", "+c+"]\t";
```

```
131                 }
132                 output+="\n";
133             }
134             return output;
135         }
136 
137         private static int bToI(final Boolean input) {
138             return input.booleanValue() ? 1 : 0;
139         }
140     }
```