```
1 //---------------------------------- Imported Packages
2 import java.util.ArrayList;
3 import java.util.Collections;
4 import java.util.Scanner;
5
6 import java.io.File;
7 import java.io.IOException;
8
9 //_____________________________________________________________________________
10 /*
11  * Name:            Luke O'Brien
12  * Class Name:      Data Structure and CS1450
13  * Class Section:   Section 002
14  * Assignment #:    4
15  * Due Date:        Feb 19, 2020
16  *
17  * Description:
18  * This program takes in and parses two files. One file contains game board
19  * data, while the other contains playing data. The programs reads the game
20  * board data and creates a 2D array according to the files specifications.
21  * It then goes through and fills that array with Targets based on the file.
22  *
23  * The program calls a playGame function that reads through the play.txt file.
24  * This file contains point data of where the ball will be. The play method
25  * checks to see if the ball hit any targets, and if it does it'll add a hit
26  * to the target in the gameBoard array. While doing this, it will print out
27  * every thing that it hits plus the total score.
28  *
29  * After the game simulation has finished, the program will go on to print
30  * out a final game report with all of the targets and their stats. Plus
31  * how many times they have been hit
32  */
33 //_____________________________________________________________________________
34
35 //-------------------------------------------- Main Class
36 public class OBrienLukeAssignment4
37 {
38     //------------------------------ Main Method
39     public static void main(String[] args) throws IOException
40     {
41         //----- Creates all class objects
42         Scanner machineTargets = new Scanner(new File("pinballMachineTargets.txt"));
43
44         int tempRow = 0;
45         int tempColumn = 0;
46         tempRow = machineTargets.nextInt();
47         tempColumn = machineTargets.nextInt();
48         PinballMachine gameBoard = new PinballMachine(tempRow,tempColumn);
49
50         //----- Adds all the Targets to the playing field
51         while(machineTargets.hasNextLine())
52         {
53             gameBoard.addTargetToPlayingField(machineTargets.nextInt(),
   machineTargets.nextInt(), new
   Target(machineTargets.next(),machineTargets.nextInt(),machineTargets.nextInt()));
54         }
55
```

```
56          //----- Prints out the Game Board
57          gameBoard.displayPlayingField();
58
59          //----- Closes scanners
60          machineTargets.close();
61
62          //----- Runs the Game and prints the report
63          playGame(gameBoard);
64          printReport(gameBoard);
65      }
66
67      //-------------------- PlayGame Method
68      public static void playGame(PinballMachine pinn) throws IOException
69      {
70          //----- Score keeper
71          int scoreKeeper = 0;
72
73          //----- Opens the file "Play.txt," Then creates pinball object
74          Scanner player = new Scanner(new File("Play.txt"));
75
76          //----- Prints out Score headline
77          System.out.println("-------------------------------------------------");
78          System.out.printf("%30s\n", "Game Simulation");
79          System.out.println("-------------------------------------------------");
80          System.out.printf("%-15s%-8s%-12s%-10s\n", "Target Hit", "ID", "Points", "Score");
81          System.out.println("-------------------------------------------------");
82
83          //----- Goes through the File and gives points if object is hit
84          while(player.hasNextLine())
85          {
86              int tempRow = player.nextInt();
87              int tempColumn = player.nextInt();
88
89              if(pinn.getTarget(tempRow, tempColumn) != null)
90              {
91                  String type = pinn.getTarget(tempRow, tempColumn).getType();
92                  int id = pinn.getTarget(tempRow, tempColumn).getID();
93                  int points = pinn.getTarget(tempRow, tempColumn).getPoints();
94
95                  pinn.getTarget(tempRow, tempColumn).incrementHit();
96                  scoreKeeper = scoreKeeper + points;
97
98                  System.out.printf("%-15s%-8d%-12d%-10d\n", type, id, points, scoreKeeper);
99              }
100         }
101
102         //----- Adds a couple lines for PrintReport
103         System.out.println("\n\n");
104
105         //----- Closes the Scanner
106         player.close();
107     }
108
109     //-------------------- PrintReport Method
110     public static void printReport (PinballMachine pinn)
111     {
112         //----- Creates the array list that stores the target reports
```

```
113         ArrayList<TargetReport> list = new ArrayList<>();
114 
115         //----- Prints out a heading for print report
116         System.out.println("*****************************************************");
117         System.out.printf("%42s\n", "PINBALL MACHINE TARGET HIT REPORT");
118         System.out.printf("%40s\n", "(From most hits to least hits)");
119         System.out.println("*****************************************************");
120         System.out.printf("%-7s%-9s%-15s%-9s%-8s%s\n", "Row", "Column", "Type", "Number",
    "Points", "hits");
121         System.out.println("-----------------------------------------------------");
122 
123         //----- Scans through the playing field and
124         //       puts targets into target report
125         for(int x=0; x<pinn.getNumRows(); x++)
126         {
127             for(int y=0; y<pinn.getNumColumns(); y++)
128             {
129                 if(pinn.getTarget(x, y) != null)
130                 {
131                     String type = pinn.getTarget(x, y).getType();
132                     int id = pinn.getTarget(x, y).getID();
133                     int points = pinn.getTarget(x, y).getPoints();
134                     int hits = pinn.getTarget(x, y).getHits();
135 
136                     list.add(new TargetReport(x, y, type, id, points, hits));
137                 }
138             }
139         }
140 
141         Collections.sort(list);
142 
143         for(TargetReport x : list)
144             System.out.println(x.print());
145     }
146 
147 }
148 
149 //-------------------------------------- PinballMachine Class
150 class PinballMachine
151 {
152     private int numRows;
153     private int numColumns;
154     private Target[][] playingField;
155 
156     PinballMachine(){
157         //Default Constructor
158     }
159 
160     PinballMachine(int numRows, int numColumns)
161     {
162         this.numRows = numRows;
163         this.numColumns = numColumns;
164         this.playingField = new Target[numRows][numColumns];
165     }
166 
167     //-------------------- addTargetToPlayingField Method
168     void addTargetToPlayingField(int row, int column, Target x)
```

```
169     {
170         playingField[row][column] = x;
171     }
172 
173     //---------------------------------- displayPlayingField Method Star
174     void displayPlayingField()
175     {
176         System.out.println("**********************************************************************
    ************");
177         System.out.printf("%41s\n", "Game board");
178         System.out.println("**********************************************************************
    ************\n");
179         System.out.printf("%7s", " ");
180         for(int col=0; col<getNumColumns(); col++)
181         {
182             System.out.printf("%10s%2d","Column:", col);
183         }
184         System.out.println();
185 
186         for(int x=0; x<getNumRows(); x++)
187         {
188             System.out.printf("%s%d","Row: ", x);
189 
190             for(int y=0; y<getNumColumns(); y++)
191             {
192                 if(getTarget(x,y) != null)
193                 {
194                     System.out.printf("%12s" ,getTarget(x,y).getType());
195                 }
196                 else
197                 {
198                     System.out.printf("%12s", "--------");
199                 }
200             }
201             System.out.println();
202         }
203         System.out.printf("\n\n\n");
204     }
205     //---------------------------------- displayPlayingField Method END
206 
207     //-------------------- getNumRows Method
208     int getNumRows()
209     {
210         return numRows;
211     }
212 
213     //-------------------- getNumColloums Method
214     int getNumColumns()
215     {
216         return numColumns;
217     }
218 
219     //-------------------- getTarget Method
220     Target getTarget(int row, int column)
221     {
222         return playingField[row][column];
223     }
```

```
}

//---------------------------------------- Target Class
class Target
{
    private String type;
    private int id;
    private int points;
    private int hits;

    Target() {
        //Default Constructor
    }

    Target(String type, int id, int points)
    {
        this.type = type;
        this.id = id;
        this.points = points;
    }

    //-------------------- getID Method
    int getID()
    {
        return id;
    }

    //-------------------- getType Method
    String getType()
    {
        return type;
    }

    //-------------------- getPoints Method
    int getPoints()
    {
        return points;
    }

    //-------------------- getHits Method
    int getHits()
    {
        return hits;
    }

    //-------------------- incrementHit Method
    void incrementHit()
    {
        hits++;
    }
}

//---------------------------------------- Target report Class
class TargetReport implements Comparable<TargetReport>
{
    private int rowNum;
    private int columnNum;
```

```
281     private String type;
282     private int id;
283     private int points;
284     private int hits;
285 
286     TargetReport() {
287         //Default Constructor
288     }
289 
290     TargetReport(int rowNum, int columnNum, String type, int id, int points, int hits)
291     {
292         this.rowNum = rowNum;
293         this.columnNum = columnNum;
294         this.type = type;
295         this.id = id;
296         this.points = points;
297         this.hits = hits;
298     }
299 
300     //-------------------- Print Method
301     public String print()
302     {
303         return String.format("%d\t%d\t%-15s%d\t%d\t%d", rowNum, columnNum, type, id, points,
    hits);
304     }
305 
306     //-------------------- CompairTo Method
307     @Override
308     public int compareTo(TargetReport other)
309     {
310         return Integer.compare(other.hits, this.hits);
311     }
312 }
```