North South University

Project Report
CSE 427
Sec: 1
Title: Tick-Tac-Toe

Submitted By:

Muhammad Sakib Khan	1520016042	muhammad.sakib@norhsouth.edu
Group:10		

GitHub Project Link:

https://github.com/nsuspring2019cse427/Group10

Submitted to:

Shaikh Shawon Arefin Shimon
Lecturer,
Department of Electrical and Computer Engineering
North South University

Table of Contents

Project Description

Introduction

Classic games that were once made for desktop computers at first are now commonly developed for smartphones. Android Operating System is highly used in modern smartphones. In this project, testing of one of such classic games on Android known as Tic-Tac-Toe has been done. The application has been developed using Java Programming Language in backend hence testing was done in Java Programming Language.

The project that has been tested by me was initially developed and available publically at: https://www.ssaurel.com/blog/learn-to-create-a-tic-tac-toe-game-for-android/

Testing Aspects

The following testing aspects have been implemented:

- Unit Testing methods
- Input Space Partitioning
- Graph Partitioning
- Integration Testing
- UI Testing

Tools/Frameworks Used:

- Android Studio
- JUnit4
- Espresso (For UI Testing)

Description of Input Space Partitioning

Interface based approach has been used to do input space partitioning.

The Tic-Tac-Toe grid is a 3X3 grid starting from (0, 0) till (2, 2). Hence negative values are considered as the equivalent positive as the negative sign is omitted.

Class: GameEngine.java. (ISP done in GameEngineTest.java)

Method: play (int x, int y)

Input Characteristics	Values	Code Line in Test Class
Both Inputs are positive and same	(1,1)	162-176
Both inputs are positive but not same	(2,1)	145-159
Both inputs are zero (Min. Grid Position)	(0,0)	127-141
Both Inputs are negative	(-2,-1)	180-198
One Input is negative another is positive	(-2,1)	202-220
Both inputs are same (Max. Grid Position)	(2,2)	108-122

Table 1: Partition for play(int x, int y)

Method: elt (int x, int y)

Input Characteristics	Values	Code Line in Test Class
Both inputs are zero (Min. Grid Position)	(0,0)	238-249
Both inputs are positive but not same	(1,2)	252-263
Both Inputs are negative	(-2,-1)	268-282
Both Inputs are negative same (Neg. Max. Grid Position)	(-2,-2)	285-299
Both inputs are same (Max. Grid Position)	(2,2)	224-235

Table 2: Partition for elt (int x, int y)

Class: MainActivity.java. (ISP done in MainActivityUITest.java)

Input Characteristics	Values	Code Line in Test Class
Input Within Limit	Length of 5. “Sakib”	39-49
Input is blank	Length of 0. “”	52-62
Input outside Limit	Length of >5. “Arsenal”	66-74

Table 3: Partition for Username Input found under the method:
btnChange.setOnClickListener(**new** View.OnClickListener()
onClick(View view){})

Graph Partitioning

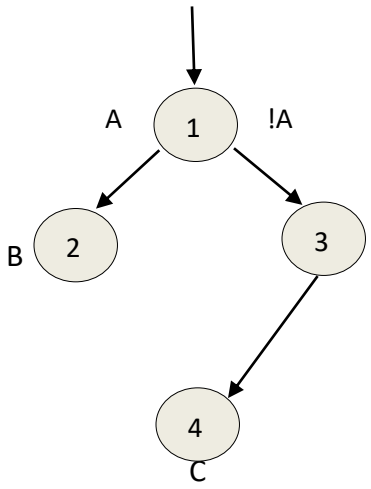
Class: GameEngine.java

Methods:

play (int x, int y)

```
31      public char play(int x, int y)// play method sets mark of the Current Player on the grid (x,y)
32                                    // 1 - graph coverage
33      {
34          if (!ended && elt(Math.abs(x),Math.abs(y)) == ' ') //' ' represents an empty place at the
35                                                             // grid where input can be placed
36                                                            // 2 - graph coverage
37          {
38              elts[3 * (Math.abs(y)) + (Math.abs(x))] = currentPlayer;  //input from player placed
39              changePlayer();   //player changes then
40          }
41          return checkEnd();          //method checks if game is over or not
42                                      // 4 - graph coverage
43      }
44
```

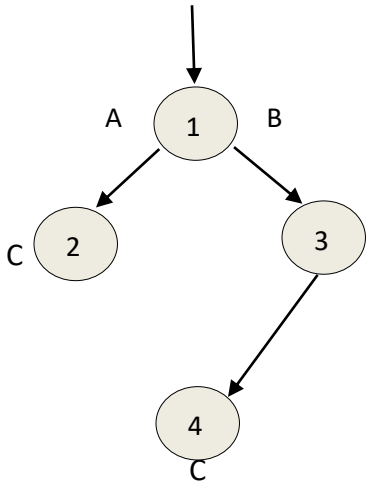
A
B
C



changePlayer ( )

```
public void changePlayer() // 1 - graph coverage
{
    currentPlayer = (currentPlayer == 'X' ? 'O' : 'X'); //player will continue as X
                                                        // 2,3,4 - graph coverage
}
```

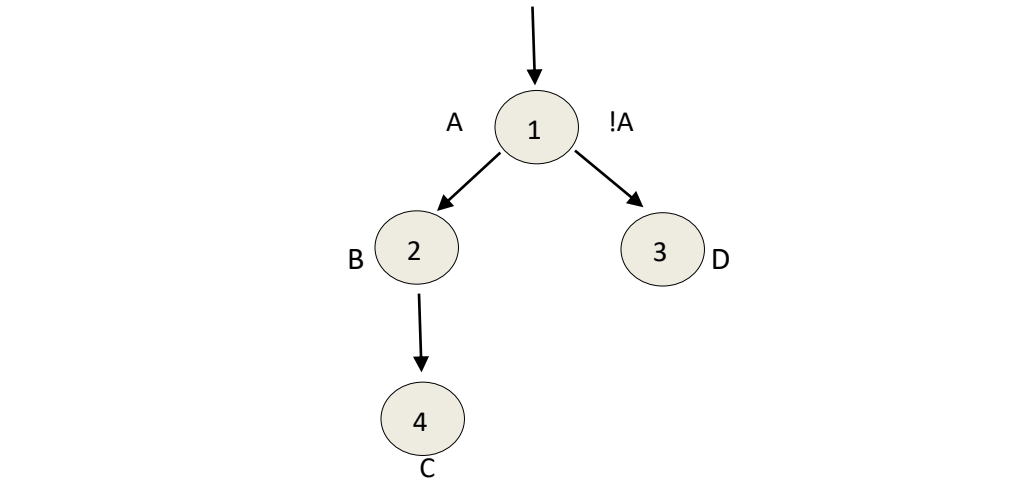
A B C



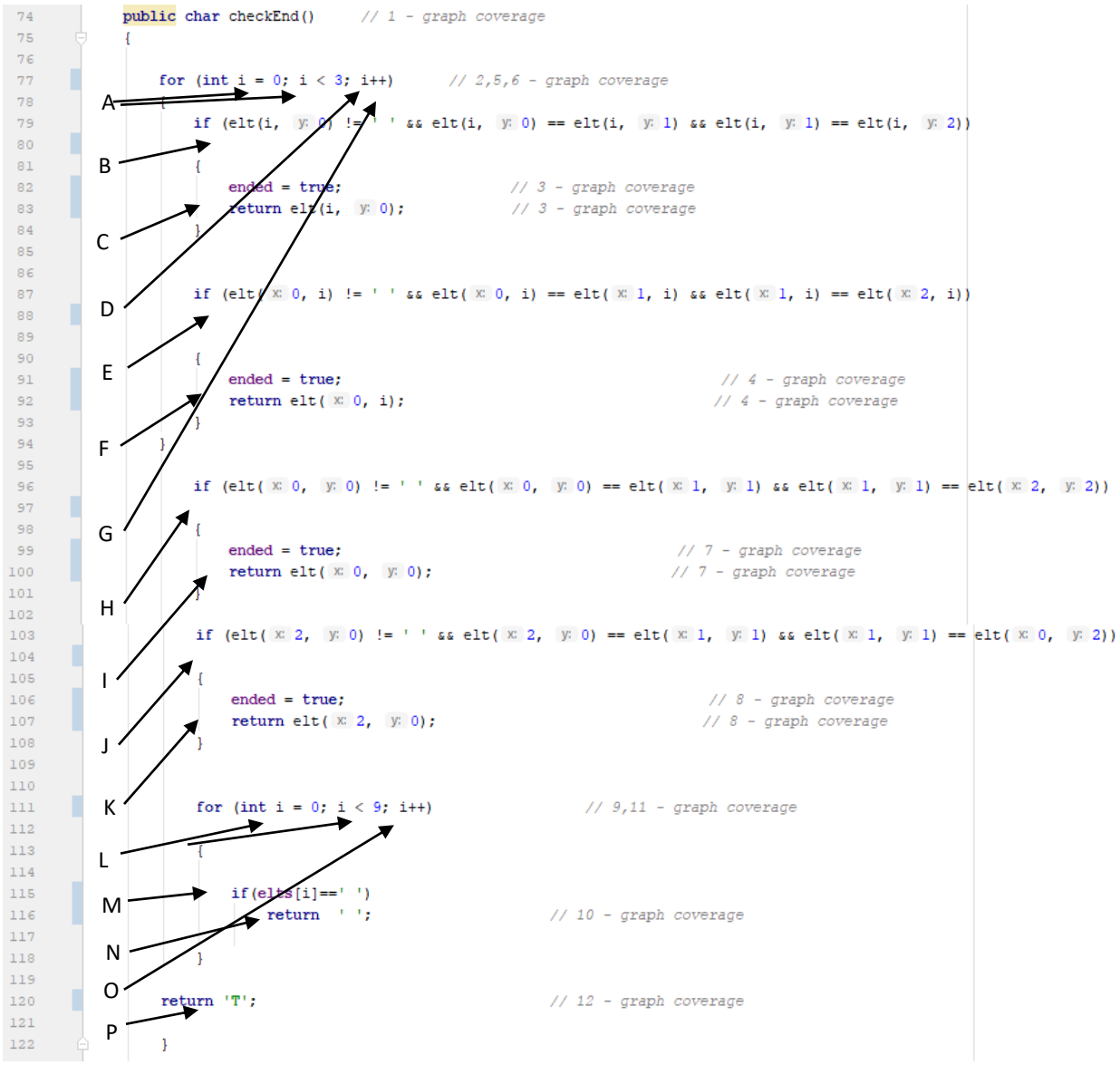
Hardcoded to X

newGame ( )

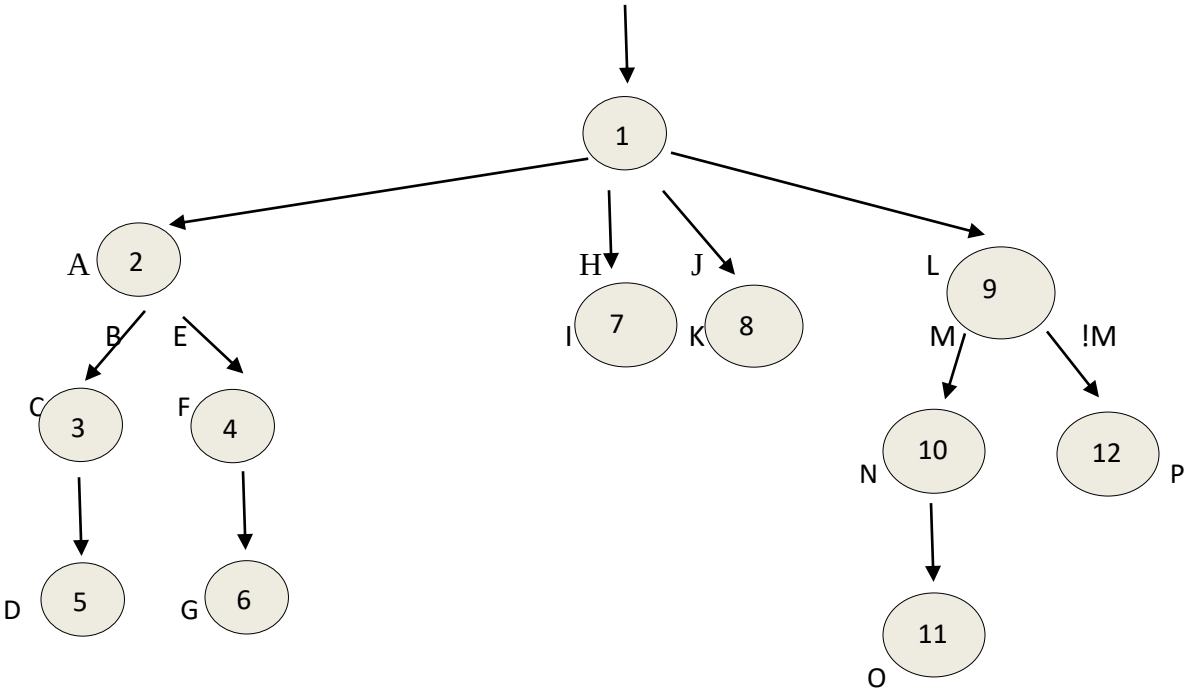
```
public void newGame() // 1 - graph coverage
{
    for (int i = 0; i < elts.length; i++)      // 4 - graph coverage
    {
        elts[i] = ' ';      // 2 - graph coverage
    }

    currentPlayer = 'X'; //User hardcoded to X while starting the game.
    ended = false;       // 3 - graph coverage

}
```



checkEnd ( )



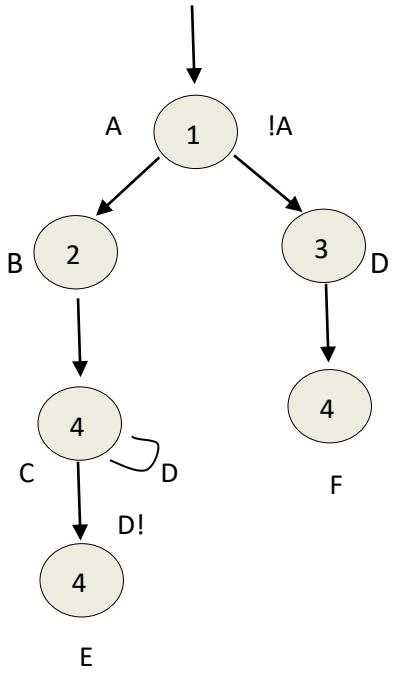

```
74      public char checkEnd()      // 1 - graph coverage
75      {
76
77          for (int i = 0; i < 3; i++)       // 2,5,6 - graph coverage
78          {
79              if (elt(i, 0) != ' ' && elt(i, 0) == elt(i, 1) && elt(i, 1) == elt(i, 2))
80
81              {
82                  ended = true;                    // 3 - graph coverage
83                  return elt(i, 0);                // 3 - graph coverage
84              }
85
86
87              if (elt(0, i) != ' ' && elt(0, i) == elt(1, i) && elt(1, i) == elt(2, i))
88
89
90              {
91                  ended = true;                                       // 4 - graph coverage
92                  return elt(0, i);                                   // 4 - graph coverage
93              }
94          }
95
96          if (elt(0, 0) != ' ' && elt(0, 0) == elt(1, 1) && elt(1, 1) == elt(2, 2))
97
98          {
99              ended = true;                                   // 7 - graph coverage
100             return elt(0, 0);                               // 7 - graph coverage
101         }
102
103         if (elt(2, 0) != ' ' && elt(2, 0) == elt(1, 1) && elt(1, 1) == elt(0, 2))
104
105         {
106             ended = true;                                   // 8 - graph coverage
107             return elt(2, 0);                               // 8 - graph coverage
108         }
109
110
111         for (int i = 0; i < 9; i++)                 // 9,11 - graph coverage
112
113         {
114
115             if(elts[i]==' ')
116                 return  ' ';                    // 10 - graph coverage
117
118         }
119
120     return 'T';                                 // 12 - graph coverage
121
122     }
```


1
A
2
H
7
J
8
L
9
B
E
I
K
M
!M
C
3
F
4
10
12
N
P
D
5
G
6
11
O

computer ( )

```
124     public char computer()          // 1 - graph coverage
125     {
126
127     if (!ended){
128         int position;          //initial position of computer is outside boundary to make sure
129                                // user places first
130                                // 2 - graph coverage
131         do {
132             position = RANDOM.nextInt( bound: 9);     // user places his turn
133         }while (elts[position] != ' ');               // when it sees an empty on the grid
134                                                       // 4 - graph coverage
135         elts[position] = currentPlayer;
136         changePlayer();                           //switches back to user if game not over
137                                                   // 5 - graph coverage
138
139     }
140         return checkEnd();              //returns to check if the game is over or not
141
142 }
```

A
B
C
D
E
F



Class: MainActivity.java

Methods:

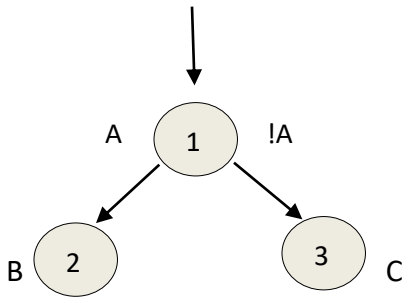
onOptionsItemSelected (MainItem item)

```
86      @Override
87      public boolean onOptionsItemSelected(MenuItem item) // 1 - graph coverage
88
89      {
90          if (item.getItemId() == R.id.action_new_game)
91          {
92
93              newGame();                                  // 2 - graph coverage
94          }
95
96          return super.onOptionsItemSelected(item);       // 3 - graph coverage
97      }
98
```

A

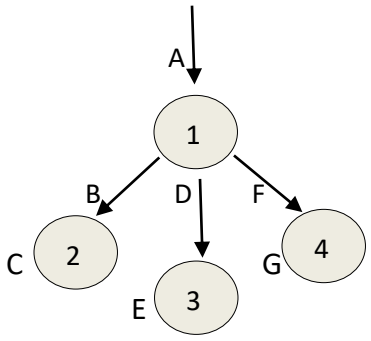
B

C



OnClick (View view)

```
54      @Override
55      public void onClick(View view)  {
56
57          vibrator.vibrate( milliseconds: 500);                          // 1 - graph coverage
58
59          if((Textchange.length()>0) && (Textchange.length()<=5)){
60          User.setText(Textchange.getText());                             // 2 - graph coverage
61
62          }
63          if((Textchange.length()<=0)) {
64              Toast.makeText(getApplicationContext(),  text: "Outside Limit", Toast.LENGTH_SHORT).show(); // 3 - graph coverage
65              User.setText(Textchange.getText());                         // 3 - graph coverage
66
67          }
68
69          if((Textchange.length()>5)) {
70              Toast.makeText(getApplicationContext(),  text: "Outside Limit", Toast.LENGTH_SHORT).show(); // 4 - graph coverage
71
72          }
73
74      }
75
76  });
```



Node Coverage and Edge Coverage = [1, 2], [1, 3], [1, 4]