



**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.....	3
Introduction.....	3
Testing Aspects.....	3
Tools/Frameworks Used: .....	3
Description of Input Space Partitioning.....	4
Class: GameEngine.java. (ISP done in GameEngineTest.java).....	4
Method: play (int x, int y) .....	4
Method: elt (int x, int y).....	4
Graph Partitioning.....	5
Class: GameEngine.java .....	5
Methods: .....	5
Class: MainActivity.java.....	11
Methods: .....	11

# 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

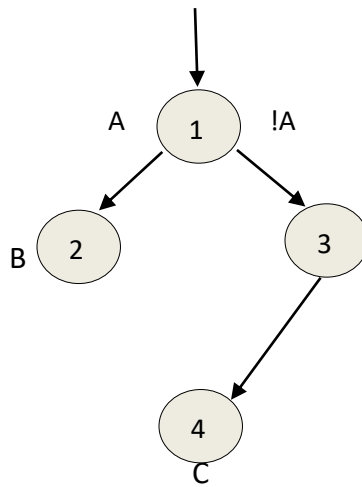
Table 2: Partition for `elt (int x, int y)`

## Graph Partitioning

Class: GameEngine.java

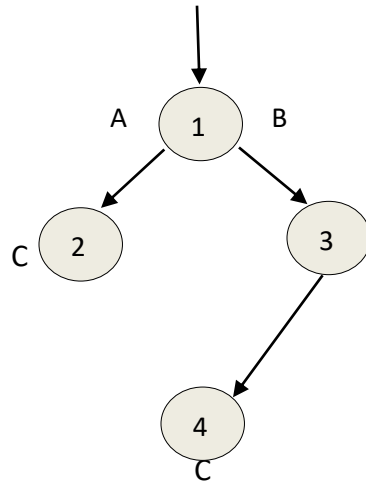
Methods:

```
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
```



*changePlayer ( )*

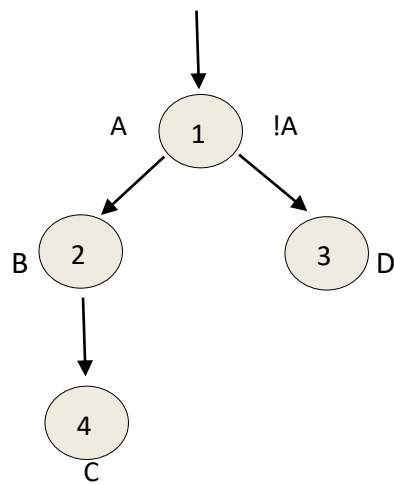
```
public void changePlayer() // 1 - graph coverage
{
    currentPlayer = (currentPlayer == 'X' ? 'Q' : 'X'); //player will continue as X
                                                         // 2,3,4 - graph coverage
}
```



Hardcoded to X

*newGame ( )*

```
public void newGame() // 1 - graph coverage
{
    A → for (int i = 0; i < elts.length; i++) // 4 - graph coverage
    {
        B → elts[i] = ' '; // 2 - graph coverage
    }
    D → currentPlayer = 'X'; //User hardcoded to X while starting the game.
    ended = false; // 3 - graph coverage
}
```



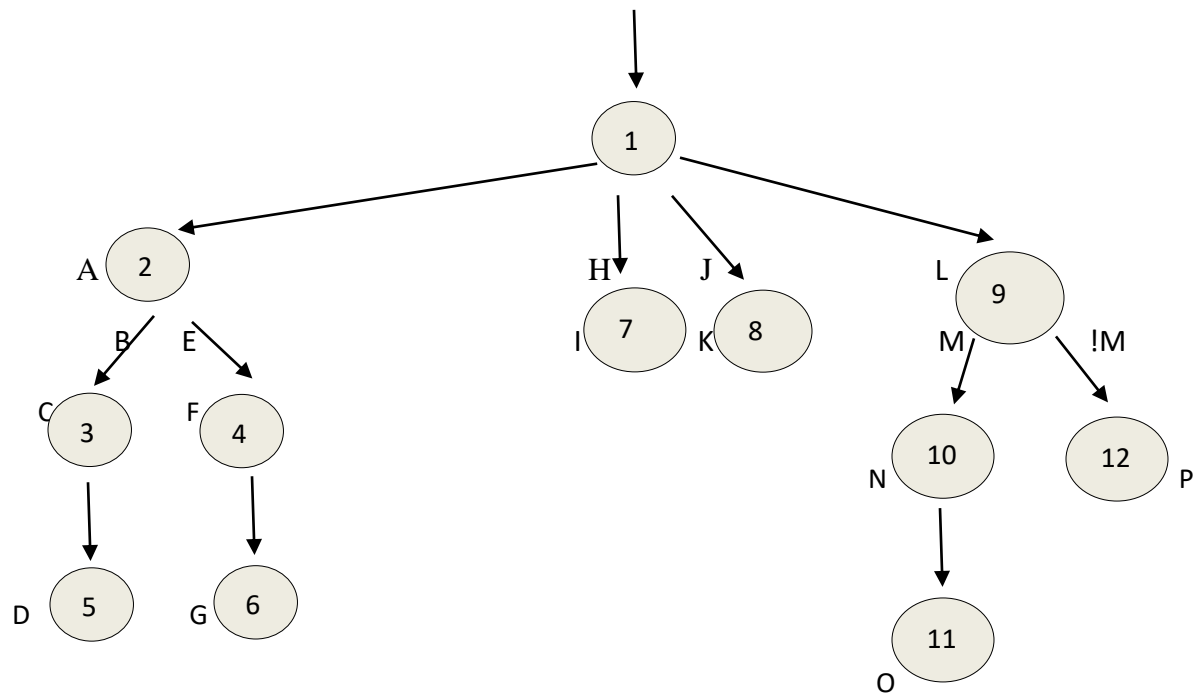
## *checkEnd ( )*

```

74  public char checkEnd()    // 1 - graph coverage
75  {
76
77      for (int i = 0; i < 3; i++)    // 2,5,6 - graph coverage
78          A →
79          if (elt(i, y:0) != ' ' && elt(i, y:0) == elt(i, y:1) && elt(i, y:1) == elt(i, y:2))
80
81              B →
82              {
83                  ended = true;    // 3 - graph coverage
84                  return elt(i, y:0);    // 3 - graph coverage
85              }
86              C →
87          if (elt(x:0, i) != ' ' && elt(x:0, i) == elt(x:1, i) && elt(x:1, i) == elt(x:2, i))
88
89              D →
90              {
91                  ended = true;    // 4 - graph coverage
92                  return elt(x:0, i);    // 4 - graph coverage
93              }
94              E →
95              F →
96          if (elt(x:0, y:0) != ' ' && elt(x:0, y:0) == elt(x:1, y:1) && elt(x:1, y:1) == elt(x:2, y:2))
97
98              G →
99              {
100                  ended = true;    // 7 - graph coverage
101                  return elt(x:0, y:0);    // 7 - graph coverage
102              }
103              H →
104          if (elt(x:2, y:0) != ' ' && elt(x:2, y:0) == elt(x:1, y:1) && elt(x:1, y:1) == elt(x:0, y:2))
105
106              I →
107              {
108                  ended = true;    // 8 - graph coverage
109                  return elt(x:2, y:0);    // 8 - graph coverage
110              }
111              J →
112          for (int i = 0; i < 9; i++)    // 9,11 - graph coverage
113              K →
114              {
115                  L →
116                  if (elts[i] == ' ')
117                      M →
118                      return ' ';    // 10 - graph coverage
119                  N →
120              }
121          O →
122          return 'T';    // 12 - graph coverage
123      }
124      P →

```



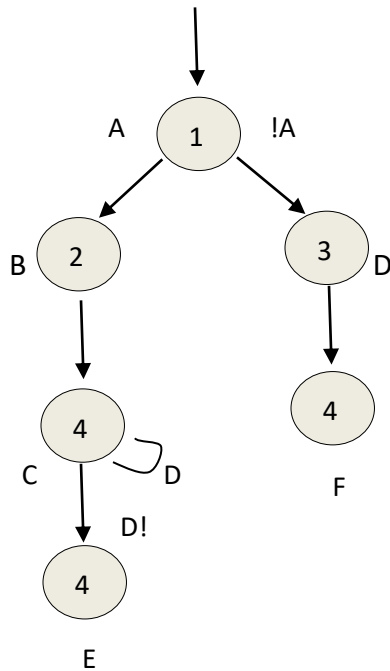


## computer ( )

```

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

```

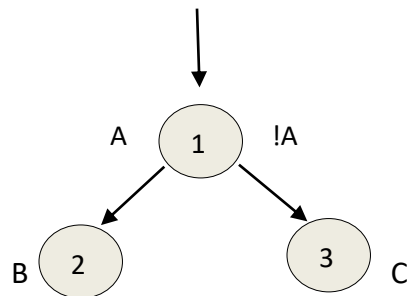


Class: MainActivity.java

Methods:

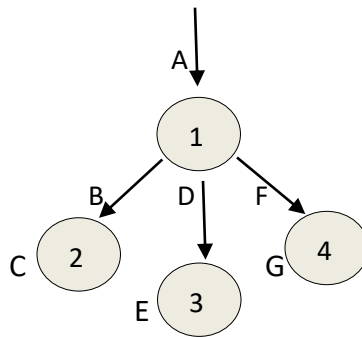
*onOptionsItemSelected (MainItem item)*

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



### OnClick (View view)

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



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