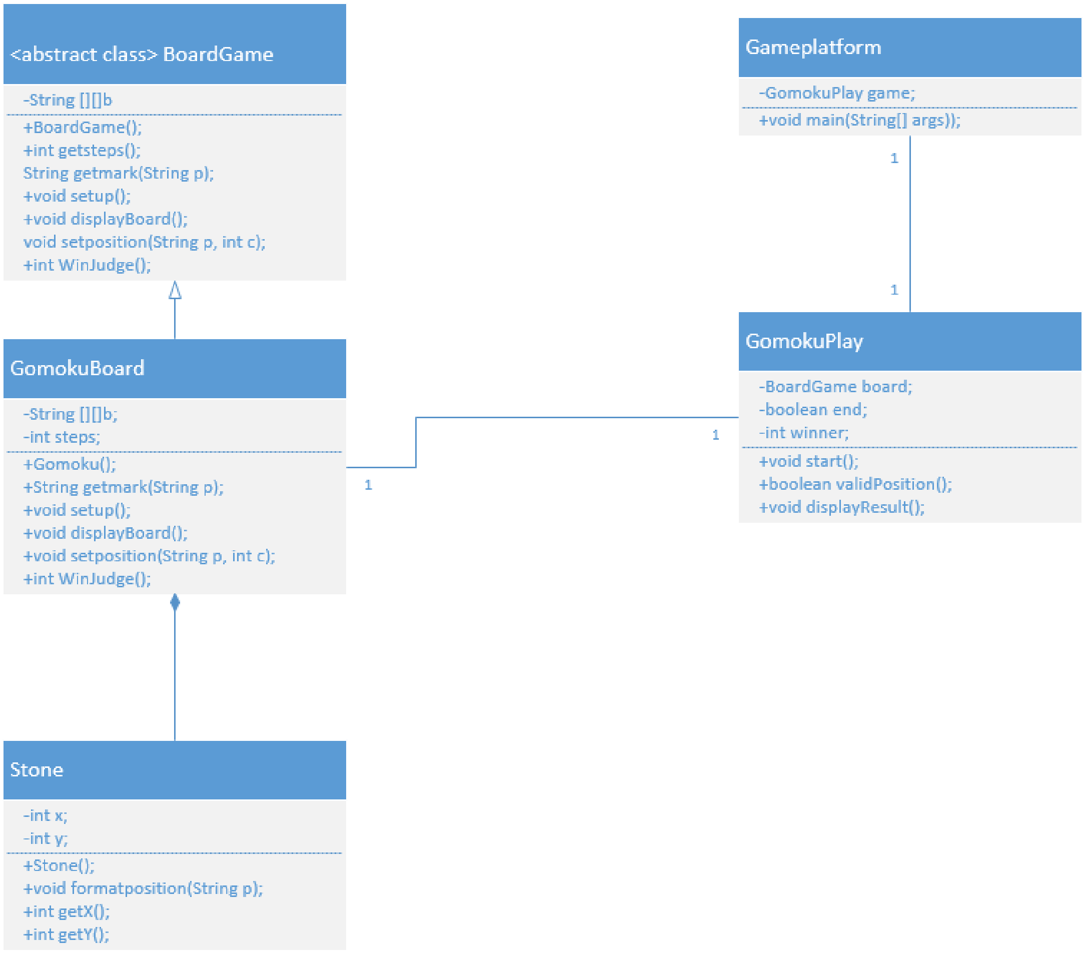
Design Analysis

1. Bad variables type design, doesn’t combine three classes into the complete program. The variables type doesn’t show any relationship between those classes.
2. Missed some function type.
3. Multiple main function.
4. Wrong UML pattern. E.g. check (5 in a row).
5. Doesn’t define appropriate method under corresponding class.
6. Console application will not contains function like MouseClick().

New design (UML Class diagram):



Design Modification

-

1. The given design doesn’t contain inheritance or polymorphism design.

2. Heritance design: In the UML class diagram above, I add an abstract class called BoardGame which has its child class called GomokuBoard. In the program The GomokuBoard could use the protected integer variable called steps. Obviously I didn’t define this variable in the Class GomokuBoard. Chile could access father’s protected variable is one of feature of inheritance.

3. Polymorphism design: In the Father class (BoardGame), it has its own set up function called void setup();, and its child class(GomokuBoard) can overwritten the setup function.
   When I define the board in the GomokuPlay as follow (notice the red words):

```
public class GomokuPlay{
    private BoardGame board;
    private boolean end;
    private int winner;//0 draw, 1 Black win, 2 White win

    public GomokuPlay(){
        board = new GomokuBoard();
        end = false;
```

```
            winner = 3;
    }
```

I define as BoardGame type, but then I initialed the “board” to the GomokuBoard instance.

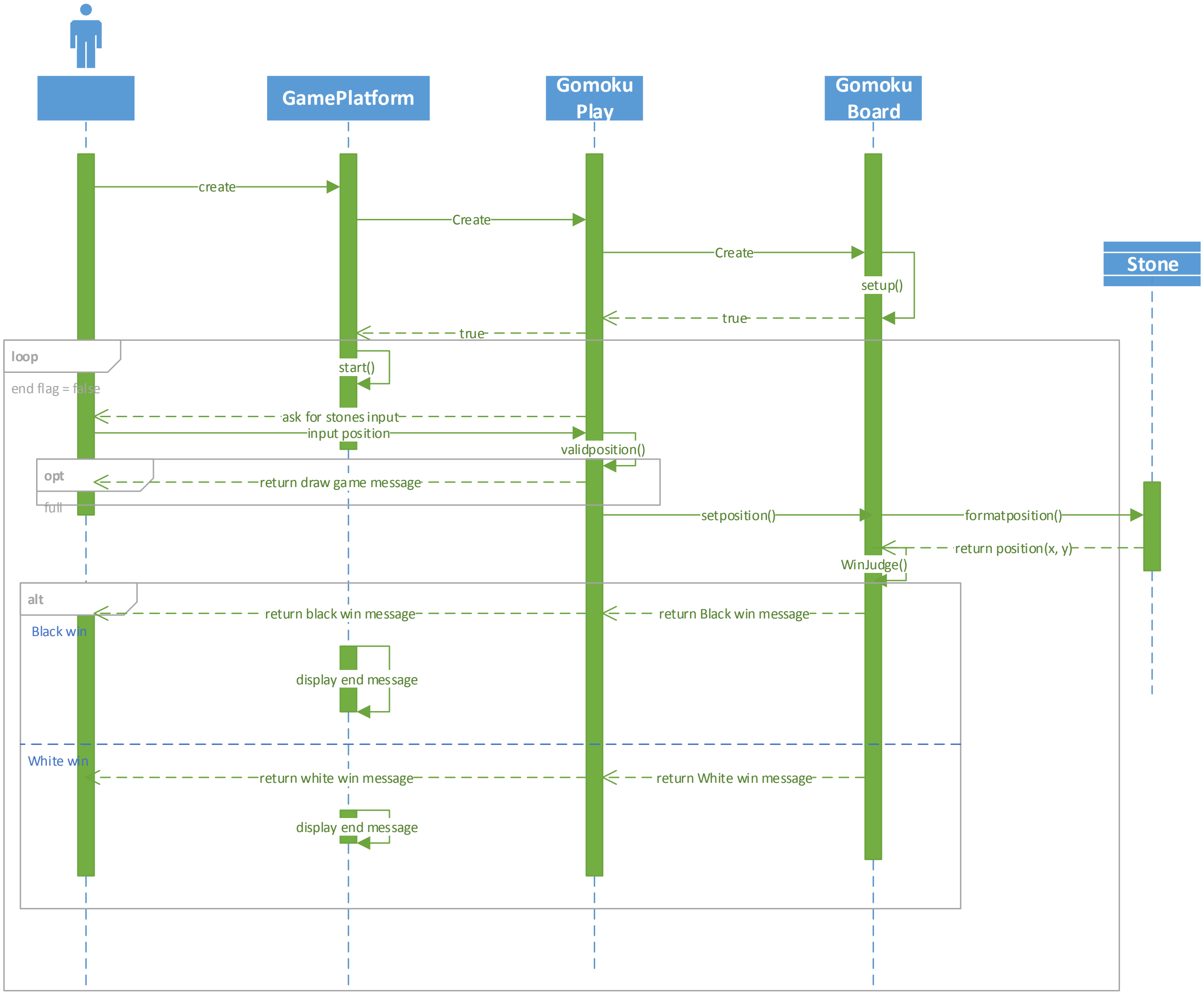
When I called the function as code shows below:

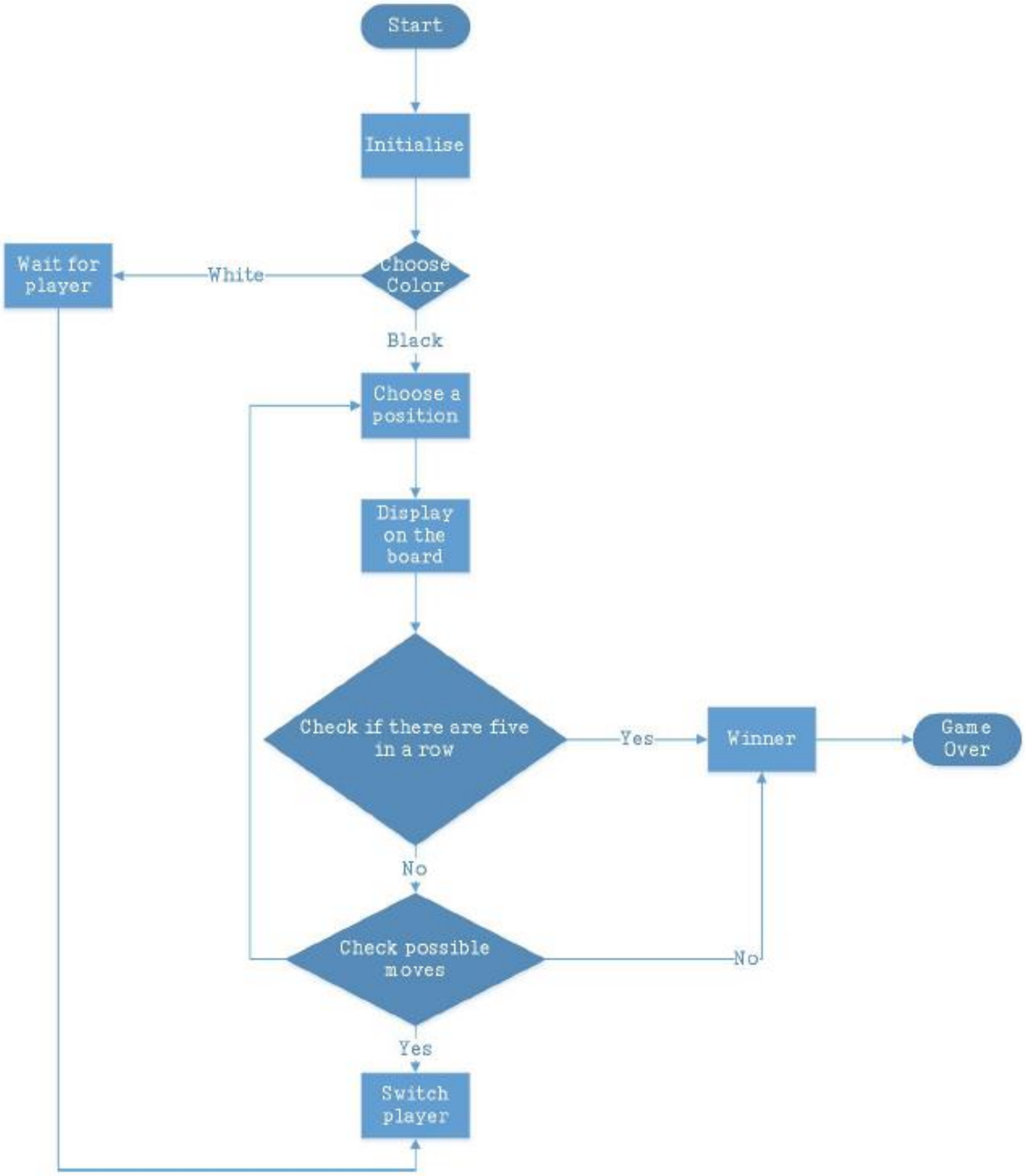
```
board.setposition(position,0);
```

The program actually called the function defined under the child class(GomokuBoard) but not the function in the father class.

PS: Its not a good design for this program but it can explain the polymorphism well and while there are more board games included in this program, it will be a good design.

UML diagrams




Start
Initialise
Choose Color
White
Wait for player
Black
Choose a position
Display on the board
Check if there are five in a row
Yes
Winner
Game Over
No
Check possible moves
No
Yes
Switch player

Derived Code

```
/*************************
Gameplatform.java*************************/
package Gomoku;

public class Gameplatform {
	private static GomokuPlay game = null;
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		System.out.println("~~~~~~~~ Welcome ~~~~~~~~~ ");
		game = new GomokuPlay();
		game.start();
		System.out.println("~~~~~~~~ game finished ~~~~~~~~~ ");
	}

}
/************************* BoardGame.java*************************/

package Gomoku;

import java.util.*;

import java.lang.*;


public class BoardGame {


	protected String [][]b;

	protected int steps;

	public void setup(){

		b = new String[15][15];

		for (int i=0; i<15 ; i++){
```

```
                for(int j = 0; j<15; j++){
                    b[i][j] = "*";
                }
            }
    }

    public String getmark(String p){
            return p;
    }

    public int getsteps(){
            return steps;
    }

    public void displayBoard(){
            for (int i=0; i<15 ; i++){
                for(int j = 0; j<15; j++){
                    System.out.print(b[i][j]);
                }
                System.out.println();
            }
    }

    public void setposition(String p, int c){
    }
    public int WinJudge(){//0 draw, 1 Black win, 2 White win , 3 game continue
        return 3;
    }
}
/************************** GomokuPlay.java************************/

package Gomoku;

import java.util.*;
import java.lang.*;

public class GomokuPlay{
    private BoardGame board;
    private boolean end;
    private int winner;//0 draw, 1 Black win, 2 White win

    public GomokuPlay(){
        board = new GomokuBoard();
        end = false;
        winner = 3;
    }
    public void start(){
        int counter = 1;
        Scanner scan = new Scanner(System.in);
        String position = null;


        while(!end){

            board.displayBoard();

            if(counter%2 == 0){
                System.out.println("Black's turn: Where do you wish to place your disc?");
                System.out.print("Please type x and y locations : ");
                position  = scan.nextLine();

                if(validPosition(position)){
                    board.setposition(position, 0);
                }else{
                    do{
                System.out.print("Invalid position,Please re-type x and y locations : ");
                        position = scan.nextLine();
                        //System.out.println("######" + position);
                    }while(!validPosition(position));

                    board.setposition(position, 0);
                }


            }else{
                System.out.println("White's turn: Where do you wish to place your disc?");
                System.out.print("Please type x and y locations : ");
                position  = scan.nextLine();


                if(validPosition(position)){
                    board.setposition(position, 1);
                }else{
                    do{
                        System.out.print("Invalid position,Please re-type x and y locations : ");
                        position = scan.nextLine();
                    }while(!validPosition(position));

                    board.setposition(position, 1);
                }
            }
            winner = board.WinJudge();
            if(winner != 3 ){
                end = true;
            }
            counter ++; //counter the amount of steps.
        }
        displayResult(winner);
    }



    //Check if the position input is correct
    public boolean validPosition(String p){
        try{
//          System.out.println(p.substring(0, 1));
//          System.out.println(p.substring(p.length()-1, p.length()));
            if(!board.getmark(p).equalsIgnoreCase("*")){
                return false;
            }
            if(board.getsteps()>64){
                return false;
            }
            if(p.length()>4){
                return false;
            }
```

```
}

            if(!Character.isDigit(Integer.parseInt(p.substring(0, 1))) &&
Integer.parseInt(p.substring(0, 1)) > 8){
                return false;
            }
            if(!Character.isDigit( Integer.parseInt( p.substring(p.length()-1,
p.length())) ) && Integer.parseInt(p.substring(p.length()-1, p.length())) > 8  ){
                return false;
            }
        }catch(Exception e){
            return false;
        }
//
        return true;
    }

    public void displayResult(int i){
        if(i == 0){
            System.out.println("Draw!");
        }else if(i == 1){
            System.out.println("Black win!");
        }else{
            System.out.println("White win!");
        }
    }
}
```

```
/**************************
GomokuBoard.java************************/
package Gomoku;
import java.util.*;
import java.lang.*;

public class GomokuBoard extends BoardGame{

	//String [][] b;
	//int steps;

	public GomokuBoard(){
		setup();
		steps = 0;
	}

	public String getmark(String p){
		Stone stone = new Stone();
		stone.formatposition(p);
		return b[stone.getX()][stone.getY()];
	}
```

```
//initialize
public void setup(){
        b = new String[8][8];
        for (int i=0; i<8 ; i++){
                for(int j = 0; j<8; j++){
                        b[i][j] = "*";
                }
        }
}

//show the Gomoku Board
public void displayBoard(){

        for (int i=0; i<8 ; i++){
                for(int j = 0; j<8; j++){
                        System.out.print(b[i][j]);
                }
                System.out.println();
        }

}

//Place the stones
public void setposition(String p, int c){
    Stone stone = new Stone();
    stone.formatposition(p);

    if(c == 0){//Black stone
        b[stone.getX()][stone.getY()] = "X";
    }else{//White stone
        b[stone.getX()][stone.getY()] = "O";
    }

}


public int WinJudge(){//0 draw, 1 Black win, 2 White win , 3 game continue

    int i,j,k;
     int tmp;
     for(i=0;i<8;i++)
     {
      for(j=0;j<8-5;j++)
      {
       tmp=b[i][j];
       if( tmp != 0)
       {
        for(k=1;k<5;k++)
         if(b[i][j+k] != tmp)
          break;
         if(k==6)
          return tmp;
       }
      }
     }

     for(i=0;i<8-5;i++)
     {
      for(j=0;j<8;j++)
      {
       tmp=b[i][j];
       if( tmp != 0)
       {
        for(k=1;k<5;k++)
         if(b[i+k][j] != tmp)
          break;
         if(k==6)
          return tmp;
       }
      }
     }
```

```
for(i=5;i<8;i++)
{
 for(j=0;j<8-5;j++)
 {
  tmp=b[i][j];
  if( tmp != 0)
  {
   for(k=1;k<5;k++)
    if(b[i-k][j+k] != tmp)
     break;
    if(k==6)
     return tmp;
  }
 }
}

for(i=0;i<8-5;i++)
{
 for(j=0;j<8-5;j++)
 {
  tmp=b[i][j];
  if( tmp != 0)
  {
   for(k=1;k<5;k++)
```

```
          if(b[i+k][j+k] != tmp)
           break;
          if(k==6)
           return tmp;
         }
        }
        }
        return 0;
       return 3;
    }
}
```

```
/****************************Stone.java*****************************
*****/
package Gomoku;

public class Stone {
    private int x;
    private int y;

    public Stone(){
        x = 0;
        y = 0;
    }

    public void formatposition(String p){
        x = Integer.parseInt(p.substring(0,1));
        y = Integer.parseInt(p.substring(p.length()-1,p.length()));
    }
    public int getX(){
        return x;
    }

    public int getY(){
        return y;
    }

}
```