**Design Pattern Assessment**

Our project Tic Tac Toe follows the MVC Pattern (Model View Controller).

**List of Classes**

**Model**: DataModel.java

**View**: JFrameView.java

**Controller**: AbstractButton.java (Base class for Controller)

One of the **controller** classes is MyButton.java

The undo function of MyButton.java calls the mutator:

@Override

public void undo() {

setText("");

setopposition();

setEnabled(true);

Model.undo();

undo\_count++;

}

**Model Class**: DataModel.java

**Here you can see that it notifies all the registered observers.**

public void set\_model(int index, int player) {

if (index == 0)

data[0][0] = player;

else {

int row = index / 3;

int col = index % 3;

data[row][col] = player;

}

pre\_player = index;

if (check\_win()) {

// notice jframe to pop out window.

// reset

System.out.println("notify");

win = true;

setChanged();

notifyObservers();// notify all observer when changed;

reset();

} else if (check\_draw()) {

win = false;

setChanged();

notifyObservers();

reset();

}

}

**View class**: JFrameView.java

**Strategy**: buttonstyle.java (interface for styles of the view)

**Concrete strategies:** ConcreteButtonStyle.java and ConcreteButtonStyle2.java

Here we apply the Concrete Strategy

@Override

public void actionPerformed(ActionEvent e) {

JButton jb = (JButton) e.getSource();

if (jb.getText() == "style1") {

for (int i = 0; i < jp.getComponents().length; i++) {

MyButton mb = (MyButton) jp.getComponents()[i];

buttonstyle style = new ConcreteButtonStyle();

style.setborderline(mb);

style.setfont(mb);

style.settextcolor(mb);

}

} else {

for (int i = 0; i < jp.getComponents().length; i++) {

buttonstyle style2 = new ConcreteButtonStyle2();

MyButton mb = (MyButton) jp.getComponents()[i];

style2.setborderline(mb);

style2.setfont(mb);

style2.settextcolor(mb);

}

}