

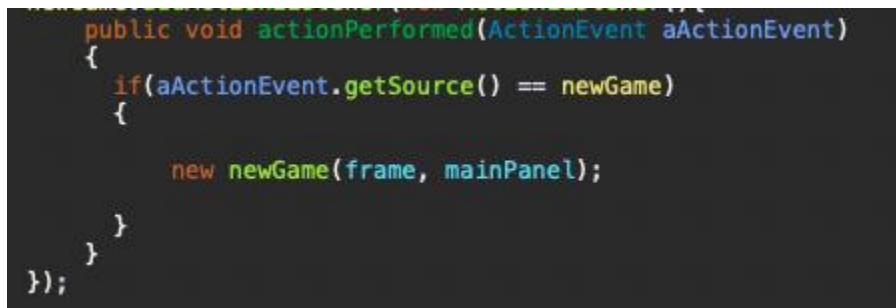
Design Patterns

The code we have designed works on many action listeners. In each of the classes we have designed, we have an action listener for conditional handling. This leads to large incohesive classes in the design. One way of implementing this could have been using the Command Interface.

The usage of Command Interface includes creating the interface called Command. The Command Interface can be implemented as follows:

```
public interface Command  
{  
    public void execute();  
}
```

The current design handles the conditional handling in this way.



```
public void actionPerformed(ActionEvent aActionEvent)  
{  
    if(aActionEvent.getSource() == newGame)  
    {  
        new newGame(frame, mainPanel);  
    }  
}  
});
```

In this current design, an action listener is added to each and every button in the following manner. This makes the code exceptionally big.

```
public void ActionPerformed( Event event)  
{  
    ((Command) event.getSource()).execute();  
}
```

```
public loadButton extends JButton implements Command
{
    public void execute()
    {
        new newGame(frame,mainPanel);
    }
}
```

The implementation above is better approach to handle action events. In this we are creating a command interface. We have a method loadButton in the Command interface that implements JButton. Each time an actionlistener is invoked the common interface is going to look for the execution using the execute().