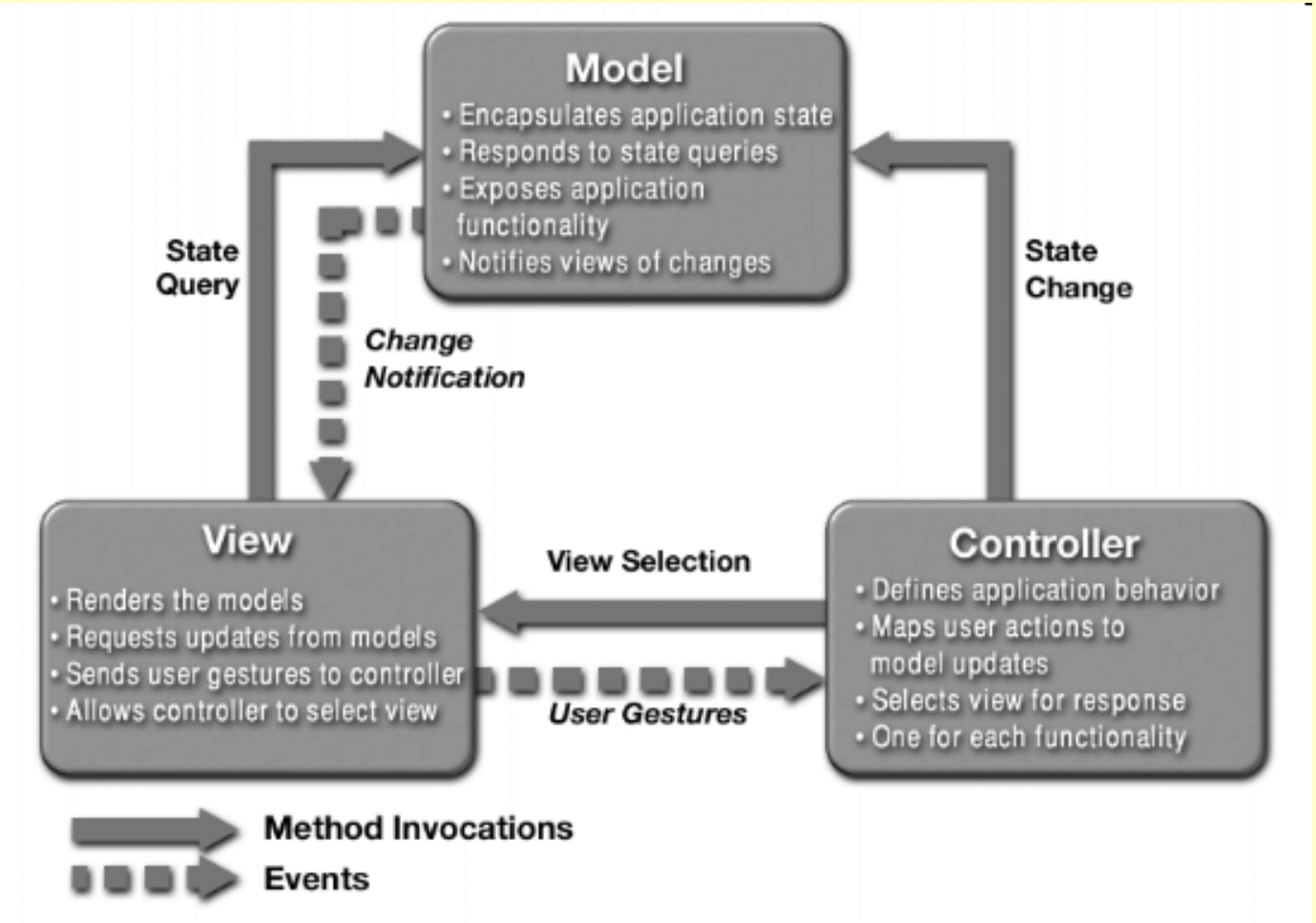
MVC Pattern


Model
• Encapsulates application state
• Responds to state queries
• Exposes application functionality
• Notifies views of changes
State Query
Change Notification
State Change
View
• Renders the models
• Requests updates from models
• Sends user gestures to controller
• Allows controller to select view
View Selection
User Gestures
Controller
• Defines application behavior
• Maps user actions to model updates
• Selects view for response
• One for each functionality
Method Invocations
Events

TP - MVC for temperature sensor: the model

```
public class TemperatureModel {
// A simple model for storing and converting temperatures
private double minTempF = -100; // minimum limit
private double maxTempF = 200; // maximum limit
private double temperatureF = 32.0; // current temperature
private double minReachedF = temperatureF; // minimum reached since initial state
private double maxReachedF = temperatureF; // maximum reached since intitial state
public double getF(){return temperatureF;}
public double getC(){return (temperatureF - 32.0) * 5.0 / 9.0;}
public double getMaxF(){return maxTempF;};
public double getMinF(){return minTempF;};
public double getMaxReachedF(){return maxReachedF;};
public double getMinReachedF(){return minReachedF;};
```

TP - MVC for temperature sensor: the model

```
public void setF(double tempF){

        temperatureF = tempF;

         if (temperatureF> maxTempF) temperatureF = maxTempF;
        else if (temperatureF< minTempF) temperatureF = minTempF;

       if (temperatureF > maxReachedF) maxReachedF= temperatureF;
       else if (temperatureF < minReachedF) minReachedF= temperatureF;

            }



public void setC(double tempC){setF (tempC*9.0/5.0 + 32.0);}

}
```

TP- MVC for temperature sensor: make model observable

```
public class TemperatureModel extends java.util.Observable

…

public void setF(double tempF){

        temperatureF = tempF;

         if (temperatureF> maxTempF) temperatureF = maxTempF;
        else if (temperatureF< minTempF) temperatureF = minTempF;

       if (temperatureF > maxReachedF) maxReachedF= temperatureF;
       else if (temperatureF < minReachedF) minReachedF= temperatureF;

            setChanged();
            notifyObservers();

            } …
```

TP - MVC for temperature sensor: make model view

```
import javax.swing.*;
import javax.swing.event.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.text.DecimalFormat;

class TemperatureView1 implements java.util.Observer
{
        private TemperatureModel model;
        private Frame temperatureFrame;
        private TextField display;

    …
```

TP - MVC for temperature sensor: make model view

```
TemperatureView1(String label, TemperatureModel model, int h, int v){
        this.model = model;
        model.addObserver(this); // Connect the View to the Model

        display = new TextField();
        DecimalFormat df = new DecimalFormat("#.##"); // display to 2 decimal places
        display.setText("" + df.format(model().getF())); // display initial value
        temperatureFrame = new Frame(label);
        temperatureFrame.add("Center", display);
        temperatureFrame.addWindowListener(new CloseListener());
        temperatureFrame.setSize(200,150);
        temperatureFrame.setLocation(h, v);
        temperatureFrame.setVisible(true);
        addDisplayListener(new DisplayListener()); // listen for updates to text field
}
```

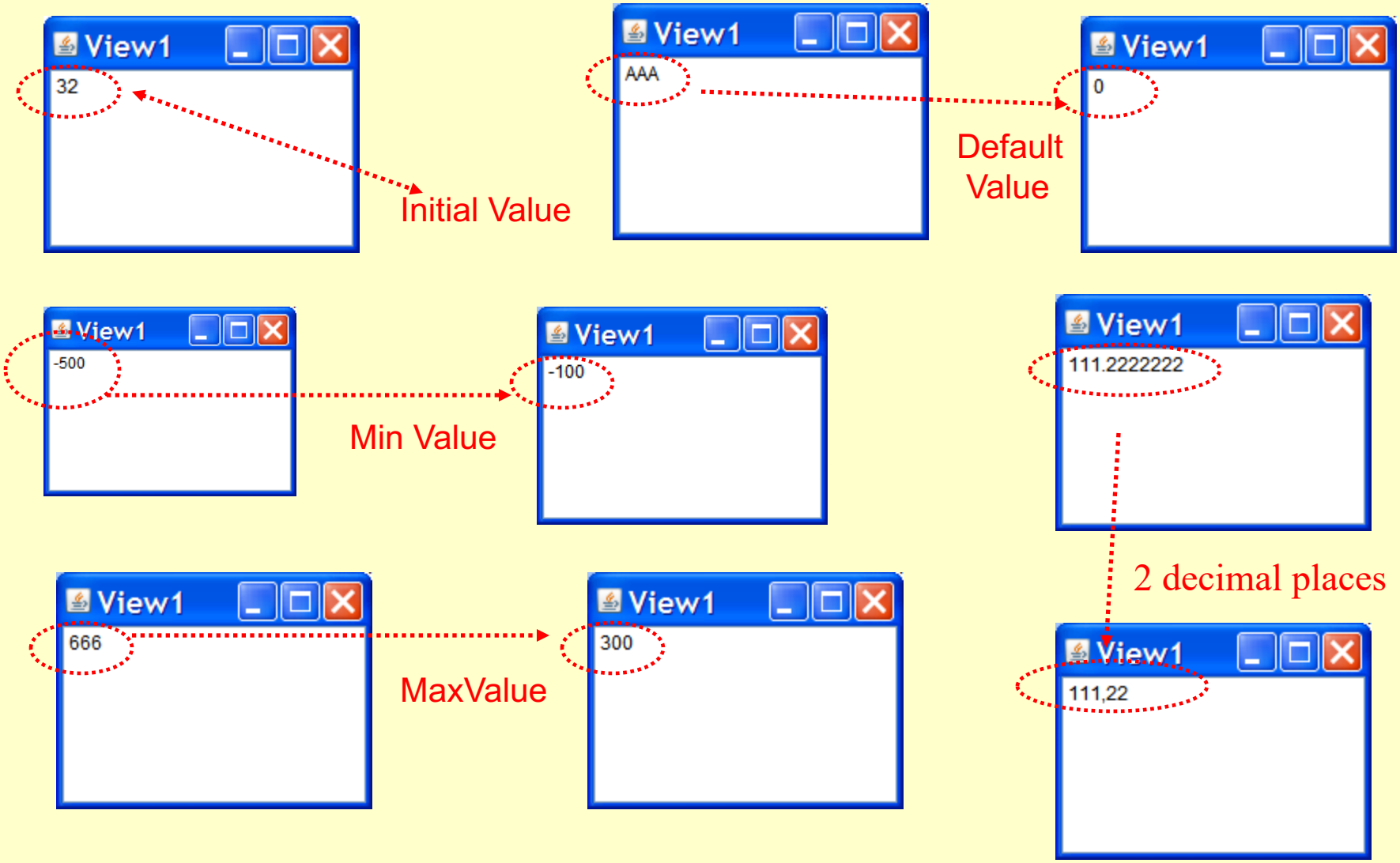
TP - MVC for temperature sensor: make model view

```
public void update(Observable t, Object o) {// when observers are notified by model

        DecimalFormat df = new DecimalFormat("#.##");
        display.setText("" + df.format(model.getF()));
            }

    public double getDisplay(){
                        double result = 0.0; // default when edit nonsensical
                        try{result = Double.valueOf(display.getText()).doubleValue();       }
                        catch (NumberFormatException e){}
                        return result;
            }

    public void addDisplayListener(ActionListener a){ display.addActionListener(a);}

    protected TemperatureModel model(){return model;}

    public static class CloseListener extends WindowAdapter{    // close all related windows
                            public void windowClosing(WindowEvent e)
                        {e.getWindow().setVisible(false);System.exit(0);}
            }

   class DisplayListener implements ActionListener
            {           public void actionPerformed(ActionEvent e){
                        double value = getDisplay(); model().setF(value);
                        }
            }
} // endclass TemperatureView1
```

TP - MVC for temperature sensor: make MVC

```
import javax.swing.*;
import javax.swing.event.*;
import java.awt.*;
import java.awt.event.*;

public class TemperatureMVC1
{       public static void main(String args[])
        {       TemperatureModel temperature = new TemperatureModel();
                new TemperatureView1("View1",temperature, 100, 100);
        }
}
```

TP - MVC for temperature sensor: make model view



TP - MVC for temperature sensor: make model view2

Change Farenheit to Celsius

TO DO: create a second view class to show temperature in Celsius

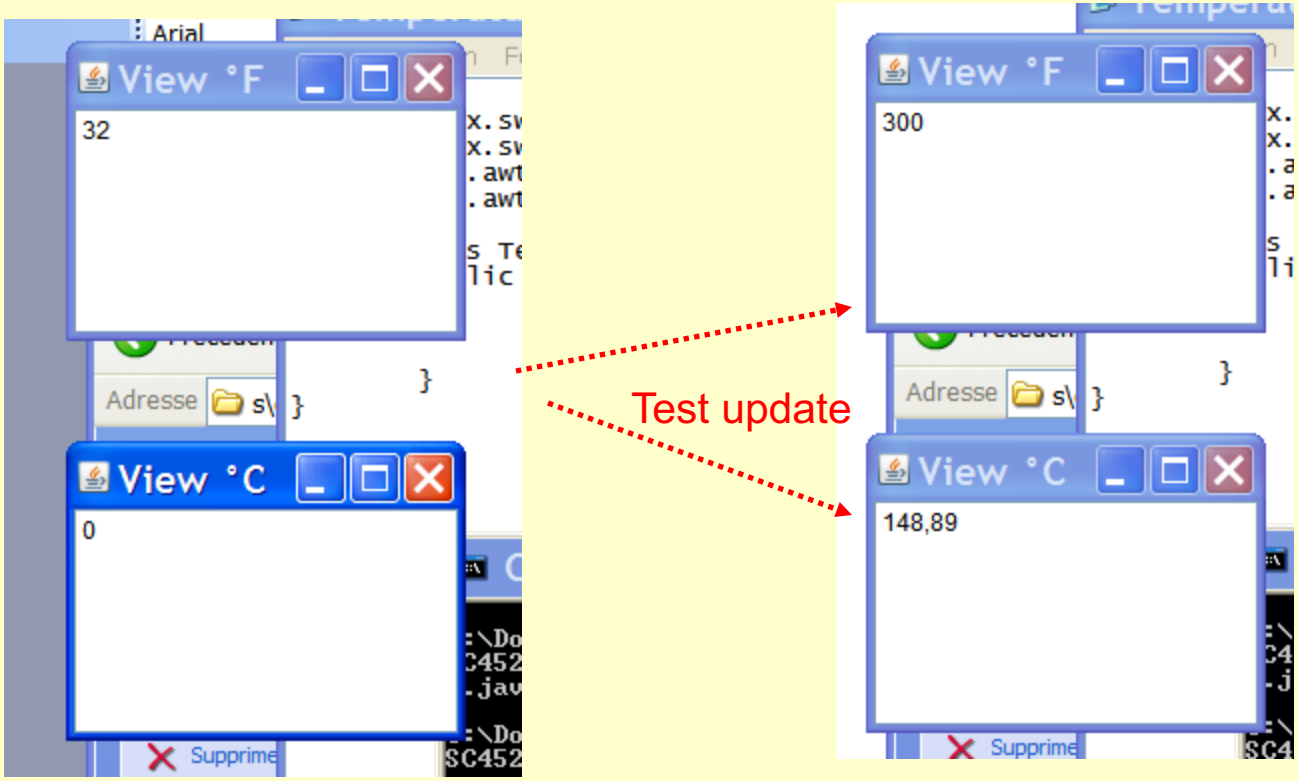
HINT:

getF -> getC

setF -> setC

TO DO: Give views meaningful names and execute together

TP - MVC for temperature sensor: make model view2




```
public class TemperatureMVC2
{		public static void main(String args[])
		{		TemperatureModel temperature = new TemperatureModel();
				new TemperatureView1("View °F",temperature, 100, 100);
				new TemperatureView2("View °C",temperature, 100, 300);
		}
}
```

TP - MVC for temperature sensor: *refactor* view1 and view2

TO DO:

•Define new abstract class TemperatureView that groups together common functionality of 2 different views

abstract class TemperatureView implements java.util.Observer

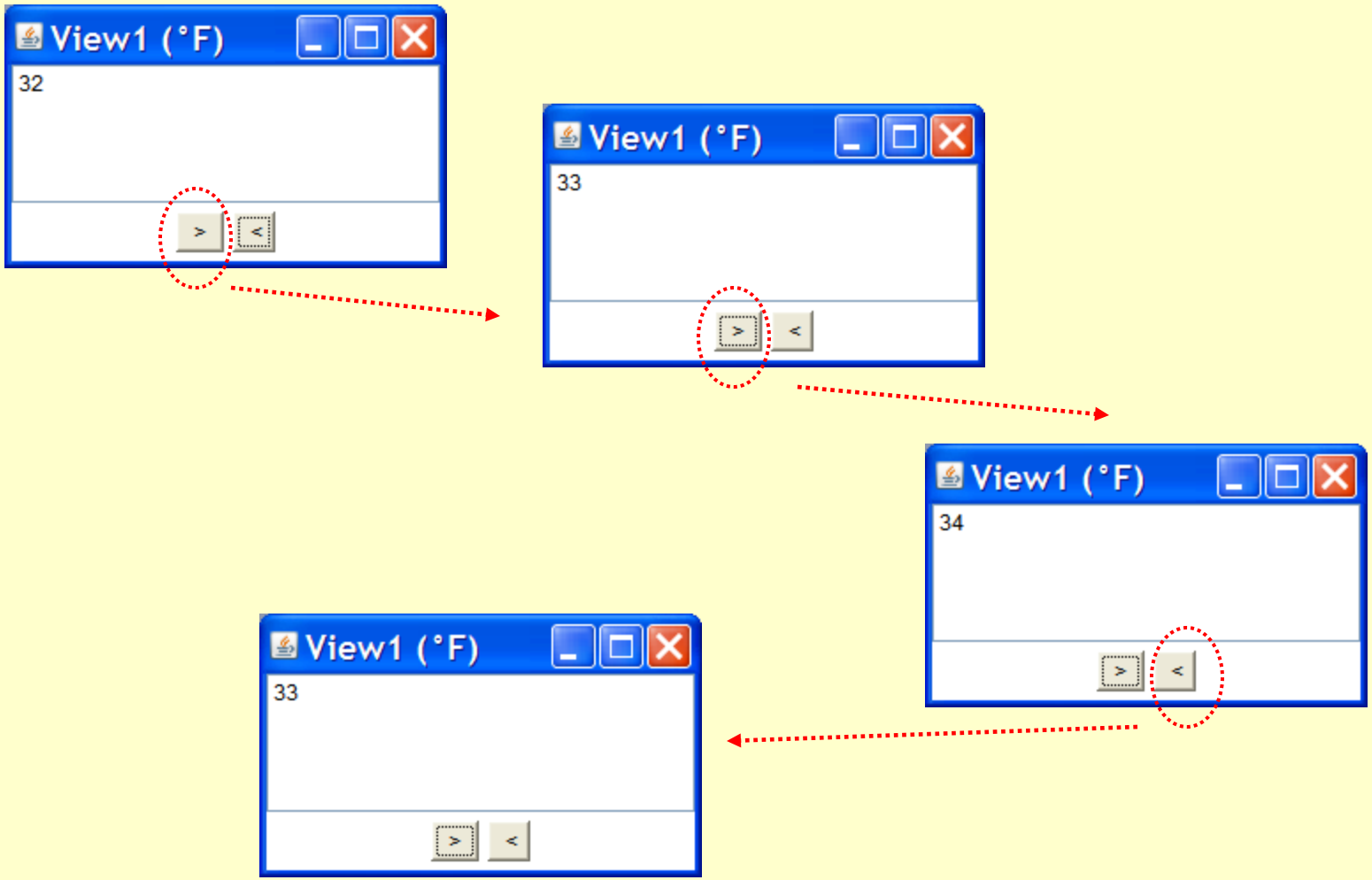
•Re-code 2 views as extensions of abstract class:

class TemperatureViewCelsius extends TemperatureView

class TemperatureViewFarenheit extends TemperatureView

TO DO: Check that functionality of system has not changed

TP - MVC for temperature sensor: *extend the views with buttons*



TP - MVC for temperature sensor: *extend the views with buttons*

```
import javax.swing.event.*;
import java.awt.*;
import java.awt.event.*;

public class TemperatureMVC12
{       public static void main(String args[])
        {       TemperatureModel temperature = new TemperatureModel();
                new TemperatureView12("View1 (°F)",temperature, 100, 100);
        }
}
```

TP - MVC for temperature sensor: *extend the views with buttons*

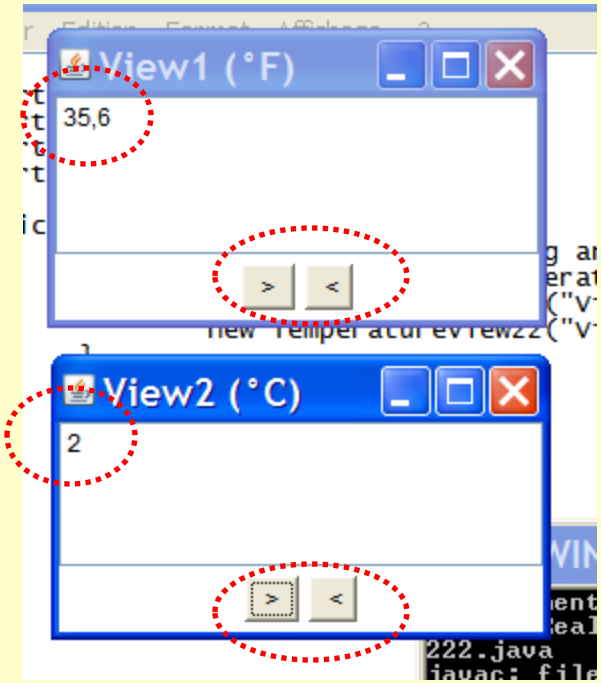
```
class TemperatureView12 extends TemperatureView1{

        private Button upButton;
              private Button downButton;

       public TemperatureView12(String label, TemperatureModel model, int h, int v){

              super(label,model,h,v);
              upButton = new Button(" > ");
              downButton = new Button(" <");
              Panel buttons = new Panel();
             buttons.add(upButton);
             buttons.add(downButton);
             temperatureFrame.add("South", buttons);
              addUpListener(new UpListener());
             addDownListener(new DownListener());


       }
```

TP - MVC for temperature sensor: *extend the views with buttons*

```
public void addUpListener(ActionListener a){
upButton.addActionListener(a);}

public void addDownListener(ActionListener a){
downButton.addActionListener(a);}

class UpListener implements ActionListener
{       public void actionPerformed(ActionEvent e)
        {       model.setF(model.getF() + 1.0);
        }
}

class DownListener implements ActionListener
{       public void actionPerformed(ActionEvent e)
        {       model.setF(model.getF() - 1.0);
        }
}


}
```

TP - MVC for temperature sensor: *extend the views with buttons*

TO DO: Extend the Celsius View with buttons in the same way and test both views together



Typical Example: Puzzle