```java
package project;

import java.awt.Color;

/* This Class is used to provide a visual interface for displaying Radio Buttons as Group
 * It used the direction and isEdit parameters to set the Radio buttons orientation and edit
 * property. */
public class RadioButtons extends JPanel implements ActionListener {

    private static final long serialVersionUID = 1L;
    private String selectedValue;
    @SuppressWarnings("unused")
    /* This is the variable hold the collection of String to populate the options */
    private String[] model;
    JRadioButton[] radioButtonArray;
    ButtonGroup buttonGroup;
    boolean isEdit;
    public RadioButtons(String[] model,int direction,boolean isEdit){
        super();
        this.isEdit=isEdit;
        GridLayout grid=null;
        this.model=model;
        if (direction==1){
             grid=new GridLayout(1, 0);
        }
        else grid=new GridLayout(0, 1);

        setLayout(grid);

        Border emptyBorder=BorderFactory.createEmptyBorder(5, 5, 5,5);
        Border etchedBorder=BorderFactory.createBevelBorder(BevelBorder.LOWERED );
        @SuppressWarnings("unused")
        Border compoundBorder=BorderFactory.createCompoundBorder(etchedBorder,emptyBorder);
        Border compoundBorder1=BorderFactory.createCompoundBorder( new CustomBorder(2, Color.blue),emptyBorder);
        @SuppressWarnings("unused")
        Border compoundBorder2=BorderFactory.createCompoundBorder( new CustomBorder(2, Color.blue),etchedBorder);
        setBorder(compoundBorder1);

        radioButtonArray=new JRadioButton[model.length];
        buttonGroup=new ButtonGroup();
        for(int i=0;i<model.length;i++){
            radioButtonArray[i]=new JRadioButton(model[i]);
            if (isEdit){
                radioButtonArray[i].setActionCommand(model[i]);
                radioButtonArray[i].addActionListener(this);
            }
            else radioButtonArray[i].setEnabled(false);
            buttonGroup.add(radioButtonArray[i]);
            add(radioButtonArray[i]);
```

```
57          }
58      }
59
60
61
62      /* This will get the selected value of the Radio Buttons */
63      public String getSelectedValue(){
64          for (int i=0;i<radioButtonArray.length;i++){
65              if (radioButtonArray[i].isSelected()){
66                  selectedValue=radioButtonArray[i].getText();
67                  return selectedValue;
68              }
69          }
70          return selectedValue;
71      }
72
73      /* This will get the selected value of the Radio Buttons */
74      public String getText(){
75          for (int i=0;i<radioButtonArray.length;i++){
76              if (radioButtonArray[i].isSelected()){
77                  selectedValue=radioButtonArray[i].getText();
78                  return selectedValue;
79              }
80          }
81          return selectedValue;
82      }
83
84      /* This will intialise the Radio Buttons to not selected state*/
85      public void initialiseButtons(){
86              buttonGroup.clearSelection();
87      }
88
89      /* This will set the given Radio button to selected state  */
90      public void setText(int i){
91           radioButtonArray[i].setSelected(true);
92      }
93      /* This will set the given Radio button to selected state  */
94      public void setText(String i){
95           radioButtonArray[Integer.parseInt(i)].setSelected(true);
96      }
97
98      /* This method is called whenever Radio Button is clicked*/
99      @Override
100     public void actionPerformed(ActionEvent e) {
101         selectedValue=e.getActionCommand();
102
103     }
104 }
105
```