```
    /*------ Display.java ------  */

package assignment_1;

import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Timer;
import java.util.TimerTask;
import java.awt.*;

public class Display extends JPanel implements ActionListener {
    JButton cook, door, cancel; // buttons
    JTextField screen, screen1, screen2; // display area
    Thread thread_stop; // thread to stop current execution
    static boolean open = false; // to check the status of door
    double counter = 60; // counter to keep track of time
    Timer timer = new Timer();

    public Display() {
        // Creation of TextArea
        screen = new JTextField(100);
        screen.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
        add(screen);
        screen1 = new JTextField(100);
        screen1.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
        add(screen1);
        screen2 = new JTextField(100);
        screen2.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
        add(screen2);
        // layout
        setLayout(new GridLayout(2, 1));
        // Creation of buttons
        // ------ cook BUTTON ----------//
        cook = new JButton("COOK");
        cook.setActionCommand("b_cook");
        add(cook);
        // ------ door BUTTON ----------//
        door = new JButton("DOOR");
        door.setActionCommand("b_door");
        add(door);
        // ------ cancel BUTTON ----------//
        cancel = new JButton("cancel");
        cancel.setActionCommand("b_cancel");
        add(cancel);
        // ------ CREATION OF ACTIONlISTENER TO BUTTONS ----------//
        cook.addActionListener(this);
        door.addActionListener(this);
        cancel.addActionListener(this);
    }
```

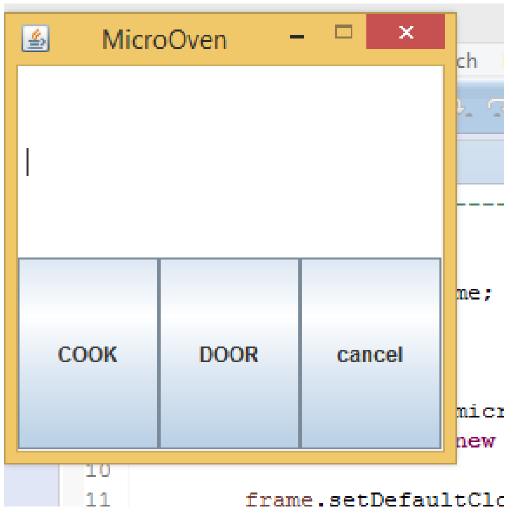
```
// assigning actions to the buttons
public void actionPerformed(ActionEvent e) {
     // method for button cook
     if ("b_cook".equals(e.getActionCommand())) {
          if (open == true) {
               screen.setText("Beep");
               screen1.setText("");
               screen2.setText("");
               counter = 60;
               timer.cancel();
          } else {
               Mytimer();
               cook.setActionCommand("b_CookAgain");
          }
     }
     // action to be performed when cook button is press again
     if ("b_CookAgain".equals(e.getActionCommand())) {
          // calling cook function in Thread
          if (open == true) {
               screen.setText("Beep");
               screen1.setText("");
               screen2.setText("");
               counter = 0;
               timer.cancel();
          } else {
               thread_stop = new Thread(() -> CookAgain());
               thread_stop.start();
          }
     }
     // method for button door
     if ("b_door".equals(e.getActionCommand())) {
          screen.removeAll();
          if (open == false) {
               open = true;
               screen.setText("Door open");
               screen1.setText("lights on");
               screen2.setText("beep");
          } else {
               open = false;
               screen.setText("Door closed");
               screen1.setText("lights off");
               screen2.setText("beep");
          }
     }
     // method for button cancel
     if ("b_cancel".equals(e.getActionCommand())) {
          timer.cancel();
          counter = 60;
          screen.removeAll();
          screen1.removeAll();
          screen2.removeAll();
```

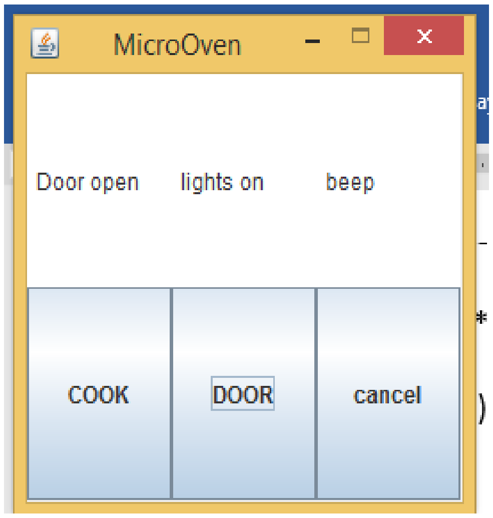
```
                screen.setText("");
                screen1.setText("");
                screen2.setText("");
            }
        }
        // method for ending call
        void CookAgain() {
            if (open == true) {
                screen.setText("Beep");
                screen1.setText("");
                screen2.setText("");
                timer.cancel();
                counter = 0;
            } else {
                counter += 60;
                Mytimer();
            }
        }
        // method for timer
        public void Mytimer() {
            TimerTask task = new TimerTask() {
                public void run() {
                    counter--;
                    if (counter == 0) {
                        screen.setText("light is off");
                        screen1.setText("beep beep");
                        screen2.setText("beep");
                        timer.cancel();
                    } else {
                        screen.setText("cooking");
                        screen2.setText(" light is on ");
                        screen1.setText(" " + counter);
                        if (open == true) {
                            timer.cancel();
                            counter = 60;
                            screen1.setText("door is open");
                        }
                    }
                }
            };
            timer.cancel();
            timer = new Timer();
            timer.scheduleAtFixedRate(task, 1000, 1000);
        }
        // method for enabling or disabling the buttons
        void buttonEnable(boolean a) {
            cook.setEnabled(a);
            door.setEnabled(a);
            cancel.setEnabled(a);
        }
}
```

```
	/*------ Oven.java---------    */
package assignment_1;

import javax.swing.JFrame;

public class Oven {

	public static void micro() {
		JFrame frame = new JFrame("MicroOven"); // using JFrame to display the
		                                                // applet
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		Display newContentPane = new Display();
		newContentPane.setOpaque(true); // content panes must be opaque
		frame.setContentPane(newContentPane);
		frame.setSize(250, 250); // applet frame size
		frame.setResizable(false); // applet frame size
		frame.setLocation(10, 10); // applet frame size
		frame.setVisible(true); // applet frame size
	}

	// main class to run thread and timer
	public static void main(String[] args) {
		javax.swing.SwingUtilities.invokeLater(new Runnable() {
			public void run() {
				micro(); // calling method micro
			}
		});
	}
}
```

/* Output

1) when the door is open display a message "Door is open"

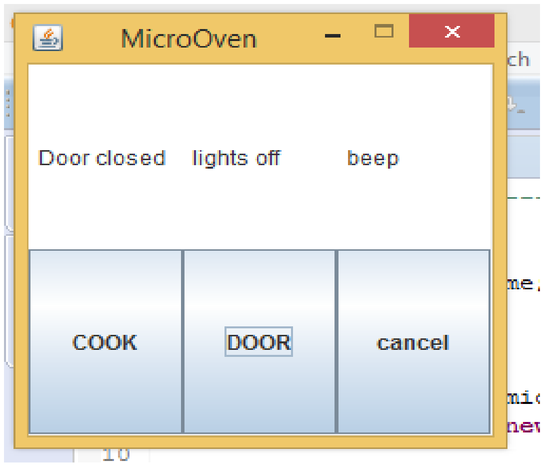




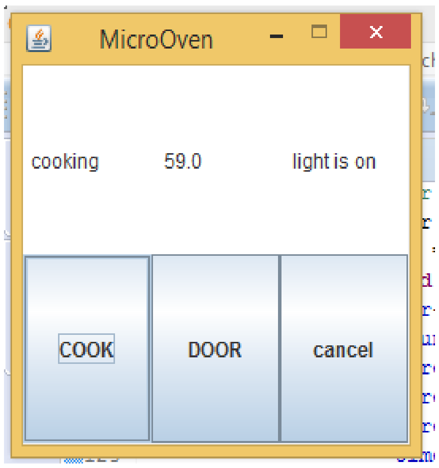
```
Door open      lights on      beep
```

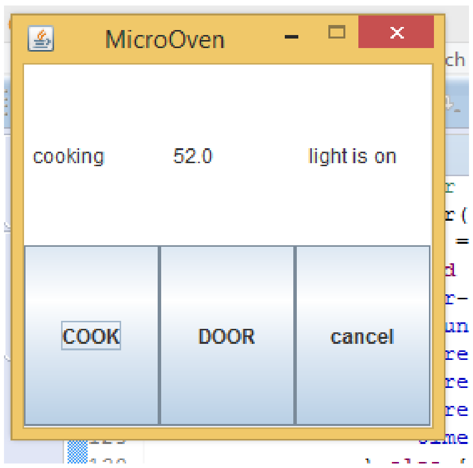
2) when the door is closed

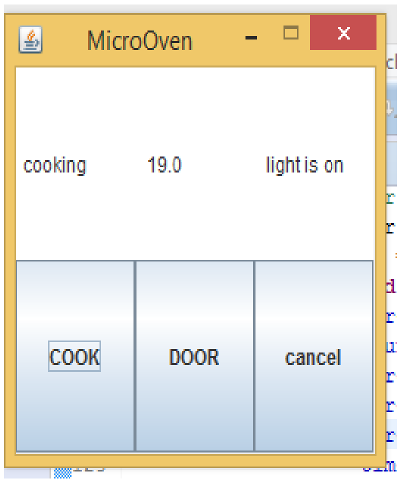
Brin Pereira Assignment 1 Batch : 12:30 -1:45

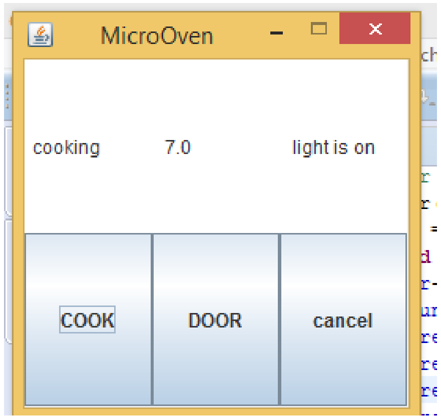


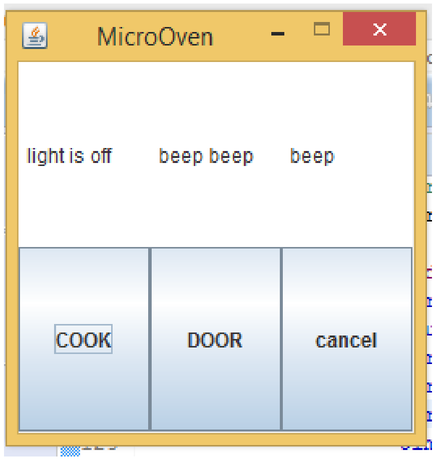
3) When the cook button is pressed, timer starts from 60 seconds to 0. (Timer is displayed, for documentation purpose I have taken only few screenshots)





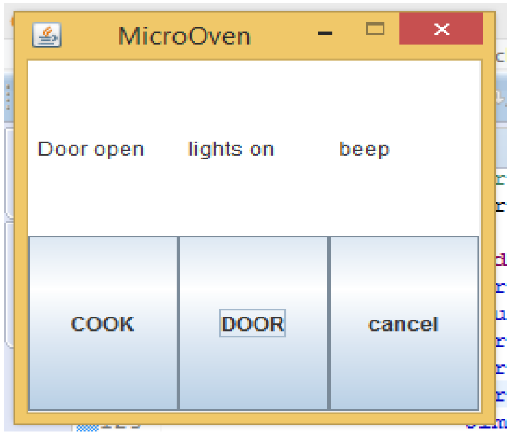


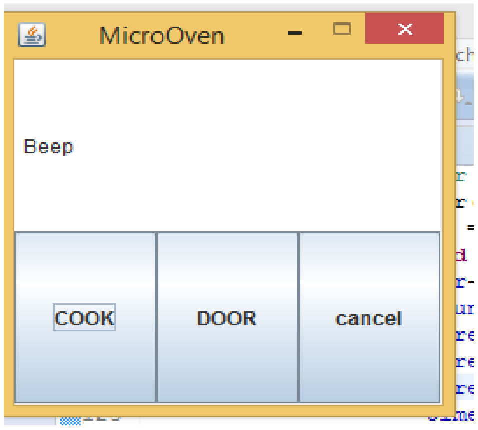




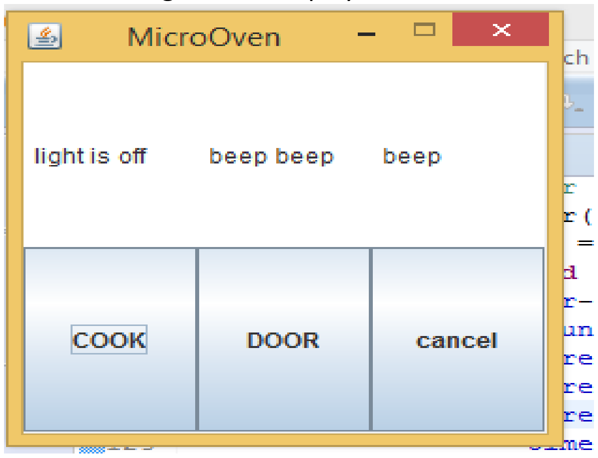
4) If the door is open and the cook button pressed just beep and do nothing else

Brin Pereira Assignment 1 Batch : 12:30 -1:45

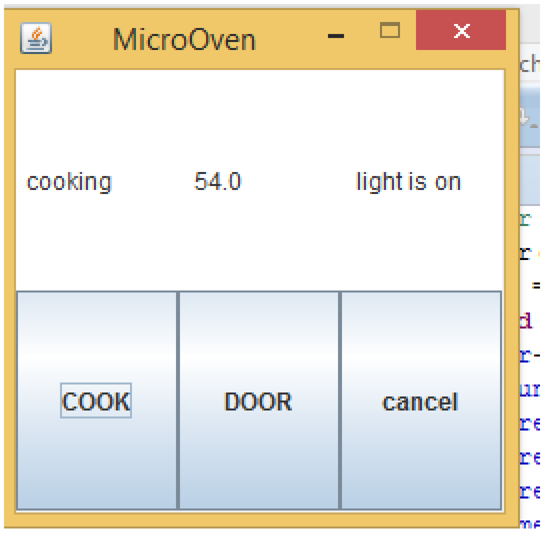


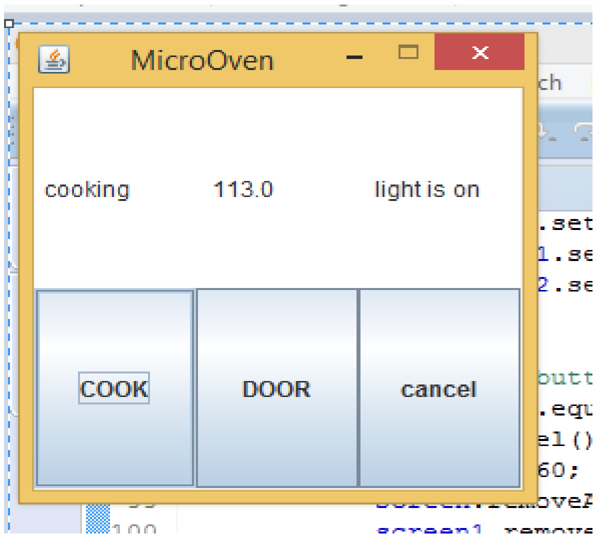


5) When cooking is done display



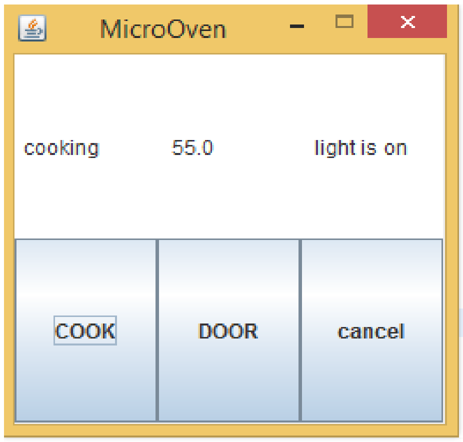
6) If cook is pressed while the oven is running, add 60 seconds to cook time. The number of seconds remaining should be equal to 60 plus whatever amount of time was at the point of pressing cook

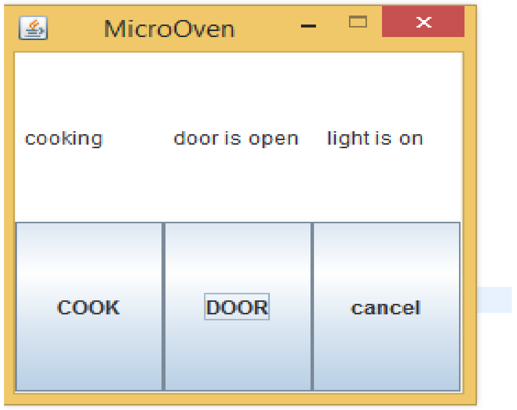




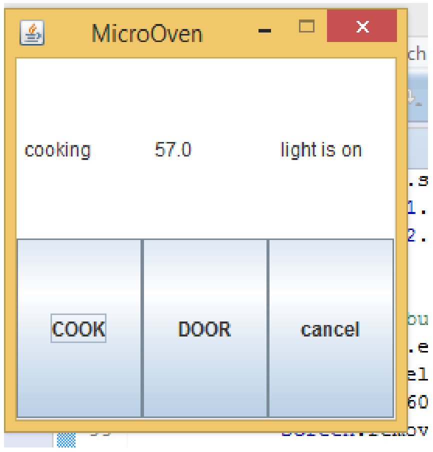
7) Whenever the oven is cooking, the light inside the oven must be on to allow the cook to see the food. The light should also go on when the oven door is opened.

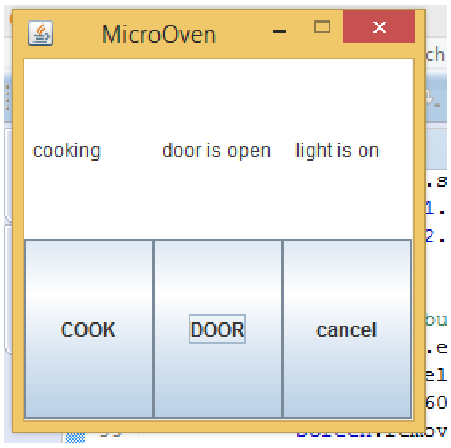
Brin Pereira Assignment 1 Batch : 12:30 -1:45

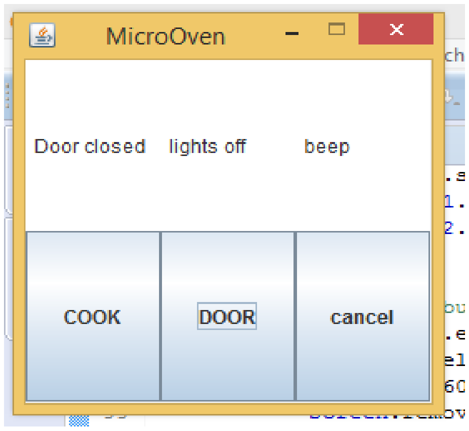




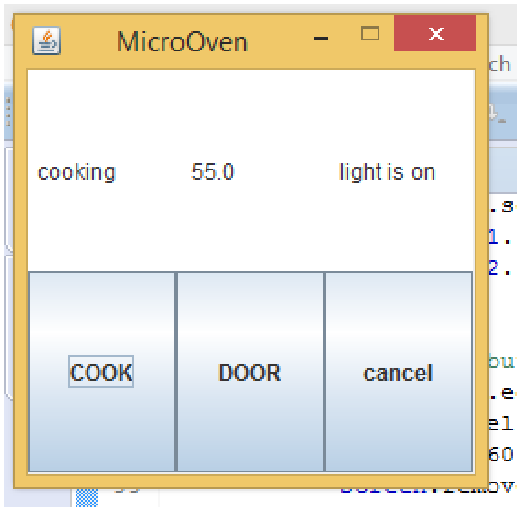
8) While the oven is cooking, opening the door will interrupt cooking. Any remaining cooking time is cleared and the oven will not beep.

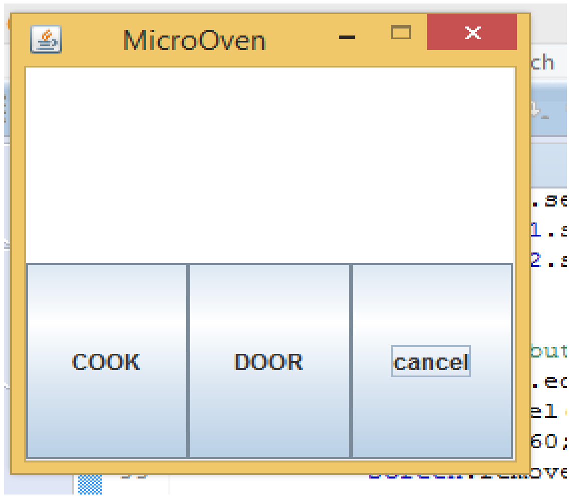






9) Pressing the cancel button while the oven is cooking will cancel cooking. The light is turned off and any remaining cooking time is cleared. The oven does not beep three times for this cooking interruption.





*/