Name: Akshay Subhash Gadhave

Gr No. : 21920090

Roll No : 221076

SY COMP A3

Assignment 5

Aim: Implement assignment 4 using Applet /Swing/AWT for UI.

Theory:

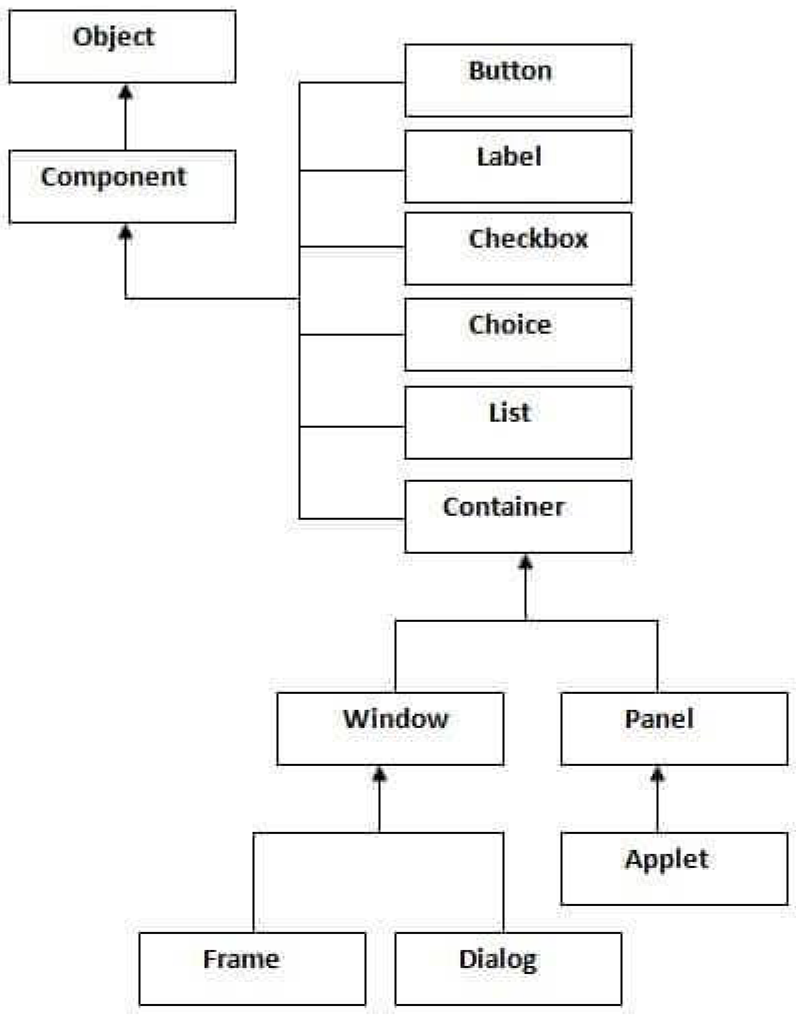
Java AWT (Abstract Window Toolkit) is *an API to develop GUI or window-based applications* in java.

Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavyweight i.e. its components are using the resources of OS.

The java.awt package provides classes for AWT api such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.

Java AWT Hierarchy

The hierarchy of Java AWT classes are given below.



Container

The Container is a component in AWT that can contain another components like buttons, textfields, labels etc. The classes that extends Container class are known as container such as Frame, Dialog and Panel.

Window

The window is the container that have no borders and menu bars. You must use frame, dialog or another window for creating a window.

Panel

The Panel is the container that doesn't contain title bar and menu bars. It can have other components like button, textfield etc.

Frame

The Frame is the container that contain title bar and can have menu bars. It can have other components like button, textfield etc.

Useful Methods of Component class

Method	Description
public void add(Component c)	inserts a component on this component.
public void setSize(int width,int height)	sets the size (width and height) of the component.
public void setLayout(LayoutManager m)	defines the layout manager for the component.
public void setVisible(boolean status)	changes the visibility of the component, by default false.

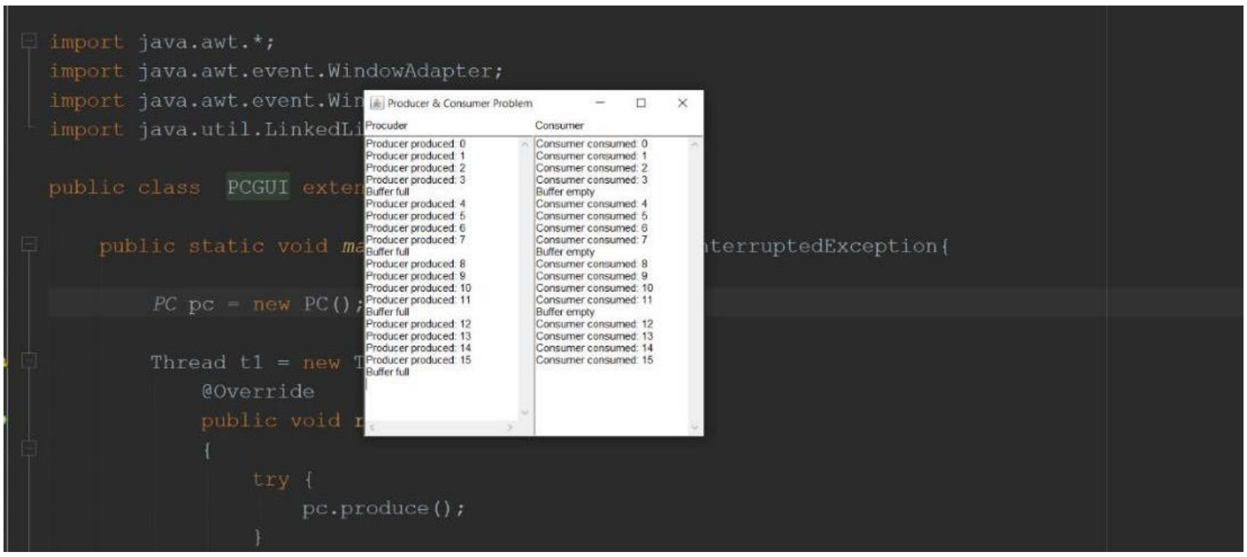
Code:

1. PCGUI.java file

```
import java.awt.*;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.util.LinkedList;

public class  PCGUI {

    public static void main(String[] args)
throws InterruptedException{

        PC pc = new PC();

        Thread t1 = new Thread(new Runnable() {
            @Override
        public void run()
        {
            try {
                pc.produce();
            }
            catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    });

    Thread t2 = new Thread(new Runnable()
        { @Override
        public void run()
        {
            try {
                pc.consume();
            }
            catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    });

    t1.start();
    t2.start();

    t1.join();
    t2.join();
}

public static class PC extends Frame{
    Label p,c;
    TextArea t1,t2;
    Panel p1,p2;

    PC(){
       super("Producer & Consumer
       Problem"); p = new Label("Procuder"); c
       = new Label("Consumer");
    p1 = new Panel();
    p1.setLayout(new GridLayout(1,2));
    p1.add(p);
    p1.add(c);
    t1 = new TextArea();
    t2 = new TextArea();
    t1.setSize(10,30);
    p2= new Panel();
    p2.setLayout(new GridLayout(1,2));
    p2.add(t1);
    p2.add(t2);
    setSize(400,400);
    setVisible(true);
    add(p1,BorderLayout.NORTH);
    add(p2,BorderLayout.CENTER);
    addWindowListener(new WindowAdapter()
        {
            @Override
            public void windowClosing(WindowEvent we)
                {
                    setVisible(false);
                    System.exit(0);
                }
        }
    );
}

LinkedList<Integer> list = new
LinkedList<>(); int capacity = 4;

public void produce() throws
InterruptedException {
    int value = 0;
    while (true) {
        synchronized (this)
        {
            while (list.size() == capacity){
                t1.append("Buffer full\n");
                wait();
            }
```

```
                // System.out.println("Producer produced-"+ value);
                 t1.append("Producer produced: 
"+Integer.toString(value)+"\n");
                 list.add(value++);

                 notify();

                 Thread.sleep(1000);
             }
         }
     }

     public void consume() throws
     InterruptedException {
         while (true) {
             synchronized (this)
             {
                 while (list.size() == 0){
                     t2.append("Buffer empty\n");
                     wait();
                 }

                 int val = list.removeFirst();
                 t2.append("Consumer consumed: 
"+Integer.toString(val)+"\n");

                // System.out.println("Consumer consumed-"+ val);

                 notify();

                 Thread.sleep(1000);
             }
         }
     }
   }
}
```

Output:



```
import java.awt.*;
import java.awt.event.WindowAdapter;
import java.awt.event.Win
import java.util.LinkedLi

public class  PCGUI  exten

    public static void ma                                        terruptedException{

        PC pc = new PC();

        Thread t1 = new T
            @Override
            public void r
            {
                try {
                    pc.produce();
                }
```

```
Procuder                        Consumer
Producer produced: 0            Consumer consumed: 0
Producer produced: 1            Consumer consumed: 1
Producer produced: 2            Consumer consumed: 2
Producer produced: 3            Consumer consumed: 3
Buffer full                     Buffer empty
Producer produced: 4            Consumer consumed: 4
Producer produced: 5            Consumer consumed: 5
Producer produced: 6            Consumer consumed: 6
Producer produced: 7            Consumer consumed: 7
Buffer full                     Buffer empty
Producer produced: 8            Consumer consumed: 8
Producer produced: 9            Consumer consumed: 9
Producer produced: 10           Consumer consumed: 10
Producer produced: 11           Consumer consumed: 11
Buffer full                     Buffer empty
Producer produced: 12           Consumer consumed: 12
Producer produced: 13           Consumer consumed: 13
Producer produced: 14           Consumer consumed: 14
Producer produced: 15           Consumer consumed: 15
Buffer full
```