Lab No1

Title: GridBag Layout

Introduction:

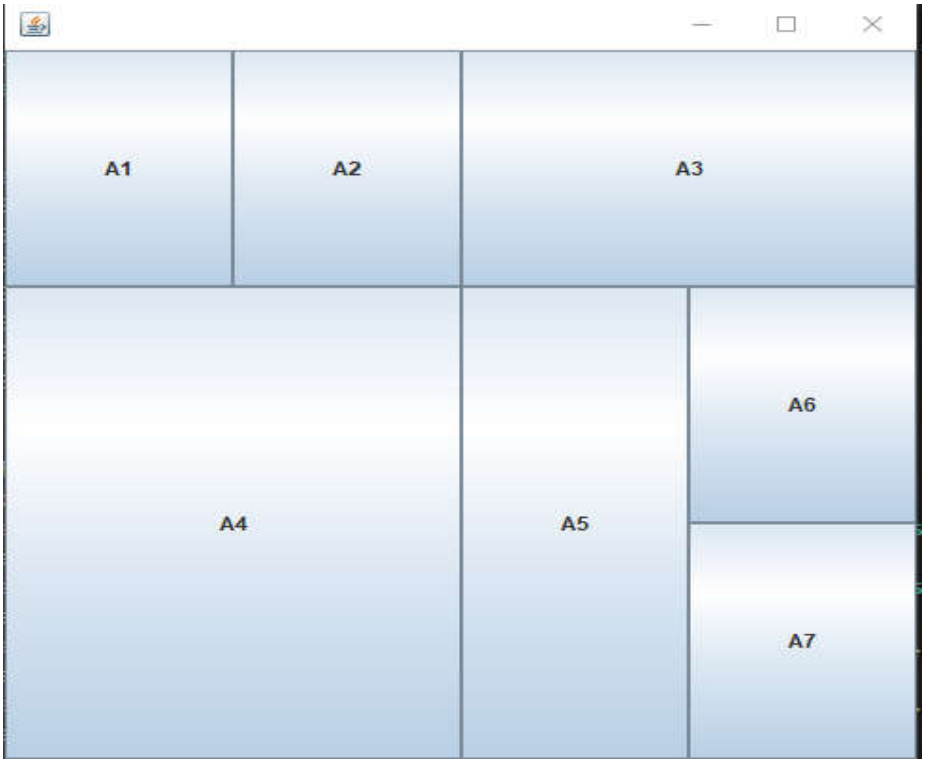
The GridBag Layout is a flexible layout manager that aligns component vertically and horizontally without requiring that the component be of same size. It maintains a dynamic rectangular grid of cell with each component occupying one or more cells. Each component managed by a Gridbag Layout associated with an instance of GridBagConstraints that specifies how the component is arranged within its display area.

Source Code:

```
import javax.swing.JButton;

import javax.swing.JFrame;

import java.awt.*;

public class B5 extends JFrame {

   JButton a1 = new JButton("A1");

   JButton a2 = new JButton("A2");

   JButton a3 = new JButton("A3");

   JButton a4 = new JButton("A4");

   JButton a5 = new JButton("A5");

   JButton a6 = new JButton("A6");

   JButton a7 = new JButton("A7");

   public B5() {

      GridBagLayout fl = new GridBagLayout();

      this.getContentPane().setLayout(fl);

      this.add(a1, new GridBagConstraints(0, 0, 1, 1, 0.25,0.33,GridBagConstraints.CENTER,
GridBagConstraints.BOTH new Insets(0, 0, 0, 0), 0, 0));

      this.add(a2, new GridBagConstraints(1, 0, 1, 1, 0.25, 0.33,
GridBagConstraints.CENTER, GridBagConstraints.BOTH new Insets(0, 0, 0, 0), 0, 0));

      this.add(a3, new GridBagConstraints(2, 0, 2, 1, 0.5, 0.33, GridBagConstraints.CENTER,
GridBagConstraints.BOTH,new Insets(0, 0, 0, 0), 0, 0));

      this.add(a4, new GridBagConstraints(0, 1, 2, 2, 0.5, 0.66, GridBagConstraints.CENTER,
GridBagConstraints.BOTH, new Insets(0, 0, 0, 0), 0, 0));

      this.add(a5, new GridBagConstraints(2, 1, 1, 2, 0.25,0.66,GridBagConstraints.CENTER,
GridBagConstraints.BOTH, new Insets(0, 0, 0, 0), 0, 0));
```

```
        this.add(a6, new GridBagConstraints(3, 1, 1, 1, 0.25,0.33,GridBagConstraints.CENTER,
GridBagConstraints.BOTH, new Insets(0, 0, 0, 0), 0, 0));

        this.add(a7, new GridBagConstraints(3, 2, 1, 1, 0.25,0.33,GridBagConstraints.CENTER,
GridBagConstraints.BOTH, new Insets(0, 0, 0, 0), 0, 0));

       setDefaultCloseOperation(EXIT_ON_CLOSE);

        setSize(500, 500);

        setVisible(true);

    }

    public static void main(String[] args) {

        B5 b5 = new B5();

    }

}
```

Output:



Result: The above program has been executed successfully and the output was verified.

Lab no2

Title: Multiple Threading by Implementing Runnable Interface.

Introduction:

Multi-threading is a programming concept in which multiple threads of execution run concurrently within a single program. In Java, multi-threading can be achieved by creating multiple threads within a single program using the java.lang.Thread class**.**

To create a new thread in Java, either extend the Thread class or implement the Runnable interface. Just need to give the definition of run() method.

Source code:

```
public class A12 implements Runnable {
   String name;
   public A12() {

  }
  public A12(String name) {
      this.name = name;
   }
   public void run() {
      while (true) {
         System.out.println(name);
         try {
            Thread.sleep(1000);
         } catch (InterruptedException ie) {

         }

      }
   }
   public static void main(String[] args) {
      A12 t1 = new A12("Thread 1");
      A12 t2 = new A12("Thread 2");
      Thread m1 = new Thread(t1);
      Thread m2 = new Thread(t2);
      m1.start();
      m2.start();
   }
}
```

Output:

```
C:\Program Files\Java\jdk1.8.0_351\bin>javac A12.java

C:\Program Files\Java\jdk1.8.0_351\bin>Java A12
Thread 1
Thread 2
Thread 1
Thread 2
Thread 2
Thread 1
Thread 1
Thread 2
Thread 2
Thread 1
Thread 1
Thread 2
Thread 2
Thread 1
Thread 1
Thread 2
Thread 2
Thread 1
Thread 1
```

Result: The above program has been executed successfully and the output was verified.

Lab no3

Title: Socket programming using TCP (Develop Chat Server)

Introduction: A Socket is an endpoint of a two way communication link between two programs running on the network.

Socket programming is a way of creating networked applications that communicate with each other using sockets, which are endpoints of a two-way communication link between two programs running on a network. TCP (Transmission Control Protocol) is a protocol used for reliable and ordered delivery of data between two endpoints.

Source Code:

 # For Server Side

```
import java.io.*;
import java.net.*;

public class ServerSide {
  public static void main(String[] args) throws IOException {
    ServerSocket serverSocket = new ServerSocket(4000);
    System.out.println("Server started");
    Socket socket = serverSocket.accept();
    System.out.println("Client connected");

    ObjectOutputStream outputStream = new ObjectOutputStream(socket.getOutputStream());
    ObjectInputStream inputStream = new ObjectInputStream(socket.getInputStream());

    BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
    String message = "";
    while (!message.equals("exit")) {
      try {
        message = (String) inputStream.readObject();
      } catch (Exception e) {
        e.printStackTrace();
      }
      System.out.println("Client: " + message);
      message = reader.readLine();
      outputStream.writeObject(message);
      outputStream.flush();
    }
    inputStream.close();
    outputStream.close();
    serverSocket.close();
    socket.close();
  }
}
```

 # For Client Side

```
import java.io.*;
import java.net.*;

public class ClientSide {
   public static void main(String[] args) throws IOException, ClassNotFoundException {
      Socket socket = new Socket("localhost", 4000);
      System.out.println("Connected to server");

      ObjectOutputStreamoutputStream=newObjectOutputStream(socket.getOutputStream());
      ObjectInputStream inputStream = new ObjectInputStream(socket.getInputStream());

      BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
      String message = "";
      while (!message.equals("exit")) {
         message = reader.readLine();
         outputStream.writeObject(message);
         outputStream.flush();
         message = (String) inputStream.readObject();
         System.out.println("Server: " + message);
      }

      inputStream.close();
      outputStream.close();
      socket.close();
   }
}
```

Output:

Server Side

```
Microsoft Windows [Version 10.0.19045.2728]
(c) Microsoft Corporation. All rights reserved.

C:\Program Files\Java\jdk1.8.0_351\bin>javac ServerSide.java

C:\Program Files\Java\jdk1.8.0_351\bin>java ServerSide
Server started
Client connected
Client: Hi
Hello,How can I help you?
Client: Please run this program ..
Sure, exit
Client: exit
exit

C:\Program Files\Java\jdk1.8.0_351\bin>
```

Client Side

```
Microsoft Windows [Version 10.0.19045.2728]
(c) Microsoft Corporation. All rights reserved.

C:\Program Files\Java\jdk1.8.0_351\bin>javac ClientSide.java

C:\Program Files\Java\jdk1.8.0_351\bin>java ClientSide
Connected to server
Hi
Server: Hello,How can I help you?
Please run this program ..
Server: Sure, exit
exit
Server: exit

C:\Program Files\Java\jdk1.8.0_351\bin>
```

Result: The above program has been executed successfully and the output was verified.

Lab no4

Title: Write a Java program to perform addition or subtraction of two numbers using swing component. Use text fields for inputs and output. The program should display output if the user clicks a key. (Handling Key Events)

Introduction: Key event handling is the process of detecting when a user interacts with a keyboard by pressing, releasing, or typing a key, and responding to that event in some way.

We can handle key events by implementing the KeyListener interface and adding an instance of your class as a key listener to a component using the addKeyListener

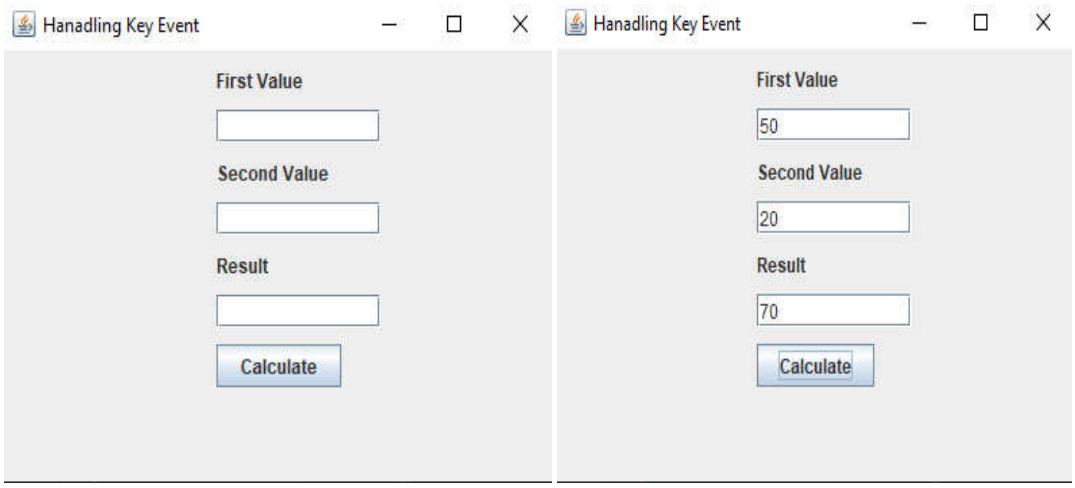
Source Code:

```
import java.awt.*;
import javax.swing.*;
import java.awt.FlowLayout;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;

public class EventTest extends JFrame implements KeyListener {
   private JTextField t1, t2, t3;
   JLabel l1, l2, l3;
   JButton b;

   public EventTest() {
      super("Hanadling Key Event");
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      l1 = new JLabel("First Value");
      l2 = new JLabel("Second Value");
      l3 = new JLabel("Result");
      t1 = new JTextField(10);
      t2 = new JTextField(10);
      t3 = new JTextField(10);
      b = new JButton("Calculate");
      b.addKeyListener(this);
      setLayout(new FlowLayout(FlowLayout.LEFT, 150, 10));
      add(l1);
      add(t1);
      add(l2);
      add(t2);
      add(l3);
      add(t3);
      add(b);
```

```
        setSize(400, 300);
        setVisible(true);
    }

    public void keyPressed(KeyEvent ke) {
        int x, y, z;
        x = Integer.parseInt(t1.getText());
        y = Integer.parseInt(t2.getText());
        if (ke.getKeyChar() == 'a')
            z = x + y;
        else if (ke.getKeyChar() == 's')
            z = x - y;
        else {
            t3.setText("Press a or s");
            return;
        }
        t3.setText(String.valueOf(z));
    }

    public void keyTyped(KeyEvent ke) {
    }

    public void keyReleased(KeyEvent ke) {
    }

    public static void main(String[] args) {
        new EventTest();
    }
}
```

Output:

If key ‘a’ is pressed



Result: The above program has been executed successfully and the output was verified.