

PROGRAMMING

Lecture 20

Sushil Paudel

PREVIOUS TOPIC

- Graphical User Interface (GUI)
- Border Layout
- Flow Layout

TODAY'S TOPIC

- GridLayout
- GridbagLayout

GRIDLAYOUT

- The GridLayout is used to arrange the components in rectangular grid. One component is displayed in each rectangle.
- Constructors of GridLayout class
- **GridLayout()**: creates a grid layout with one column per component in a row.
- **GridLayout(int rows, int columns)**: creates a grid layout with the given rows and columns but no gaps between the components.

EXAMPLE

```
import java.awt.*;
import javax.swing.*;
public class MyGridLayout{
    public MyGridLayout(){
        JFrame f=new JFrame();

        JButton b1=new JButton("1");
        JButton b2=new JButton("2");
        JButton b3=new JButton("3");
        JButton b4=new JButton("4");
        JButton b5=new JButton("5");
        JButton b6=new JButton("6");
        JButton b7=new JButton("7");
        JButton b8=new JButton("8");
        JButton b9=new JButton("9");
```

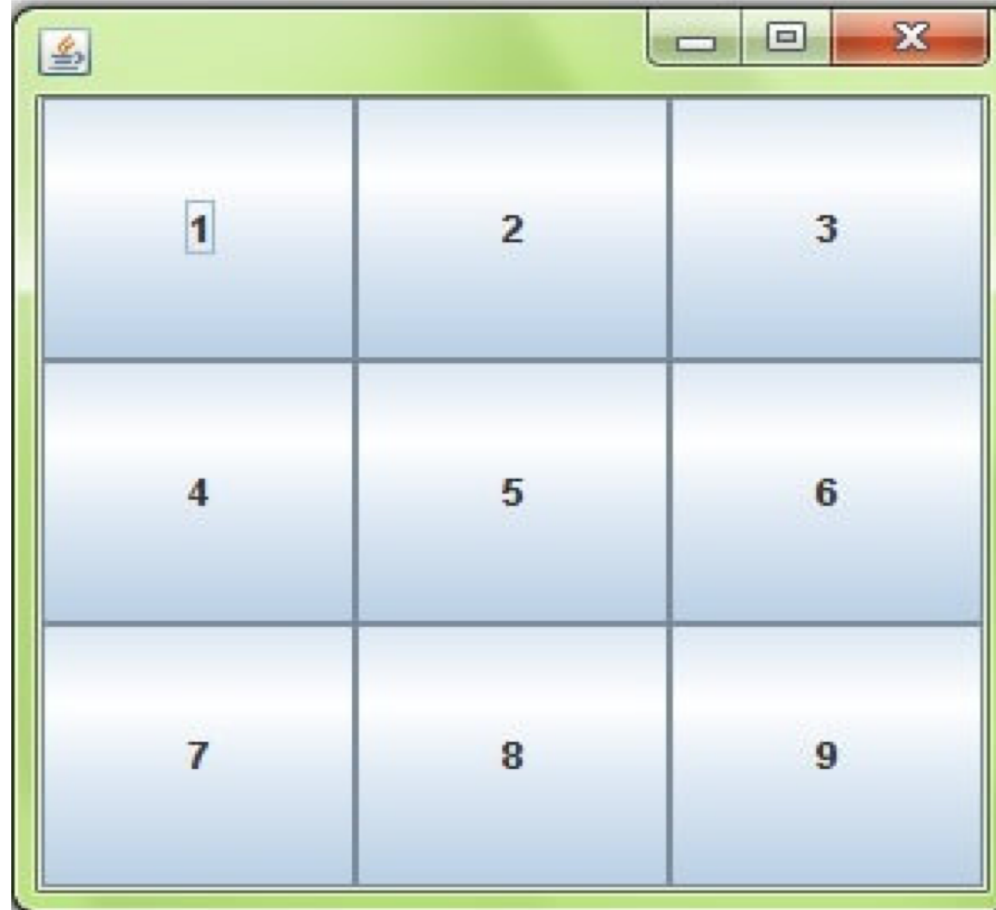
EXAMPLE

```
f.add(b1);
f.add(b2);
f.add(b3);
f.add(b4);
f.add(b5);
f.add(b6);
f.add(b7);
f.add(b8);
f.add(b9);

f.setLayout(new GridLayout(3,3));

f.setSize(300,300);
f.setVisible(true);
}
public static void main(String[] args) {
    new MyGridLayout();
}
}
```

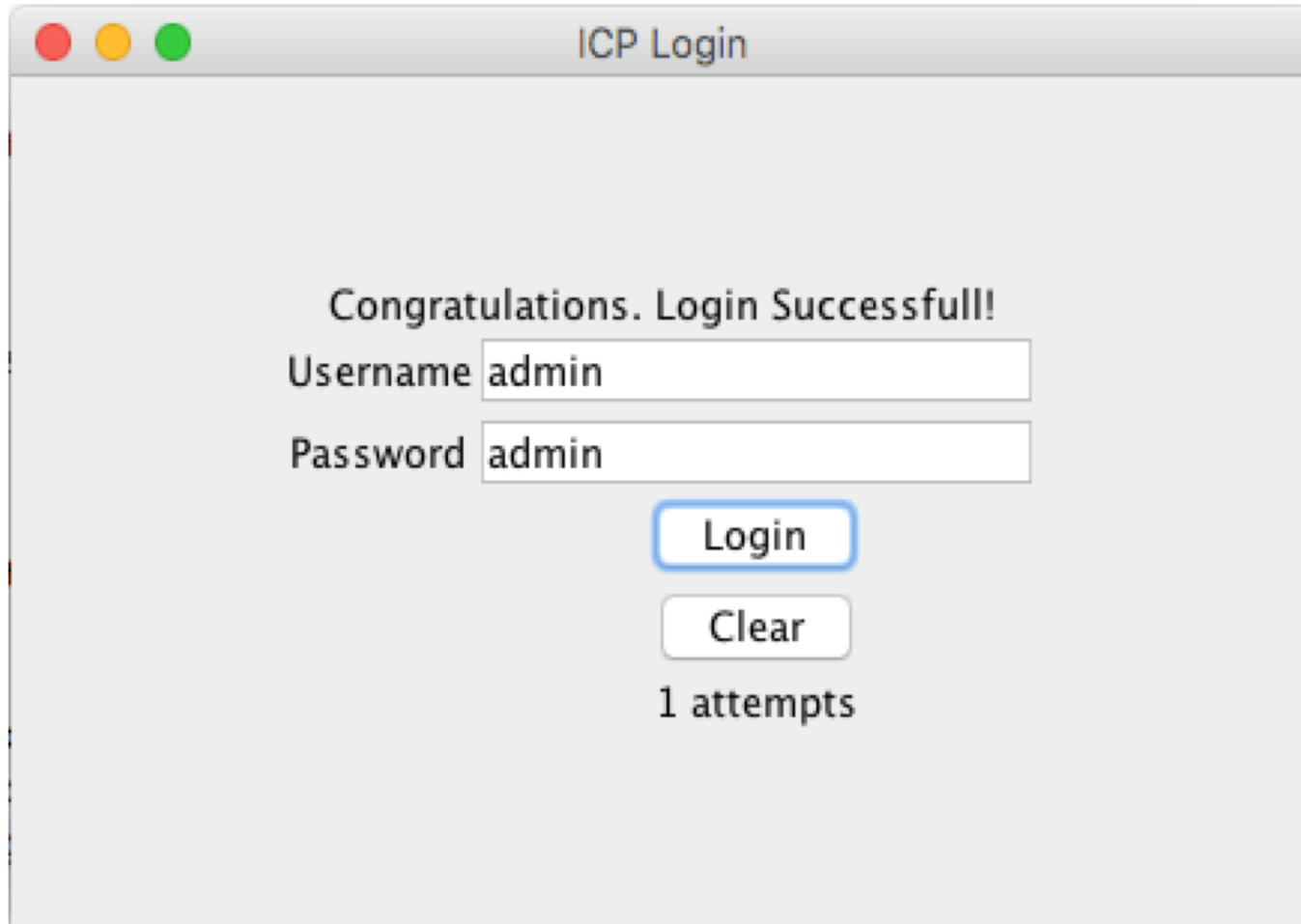
OUTPUT



GRIDBAGLAYOUT

- The GridBagLayout class is a flexible layout manager that aligns components vertically, horizontally without requiring that the components be of the same size.
- Each GridBagLayout object maintains a dynamic, rectangular grid of cells, with each component occupying one or more cells, called its *display area*.
- Each component managed by a GridBagLayout is associated with an instance of GridBagConstraints.
- The GridBagConstraints object specifies where a component's display area should be located on the grid and how the component should be positioned within its display area.

CREATING A SAMPLE LOGIN



ICP Login

Congratulations. Login Successfull!

Username

Password

1 attempts

COMPONENT POSITION

	X=0	X=1
Y=0	COMPONENT (0,0)	COMPONENT (1,0)
Y=1	COMPONENT (0,1)	COMPONENT (1,1)
Y=2	COMPONENT (0,2)	COMPONENT (1,2)

EXAMPLE

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class MyGridBagLayout {
    private JFrame frame;
    private int counter;
    public MyGridBagLayout() {
        frame = new JFrame("ICP Login");
        frame.setSize(400, 300);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);

        JPanel panel = new JPanel();
        panel.setSize(400, 300);

        JLabel usernameLabel = new JLabel("Username");
        JLabel passwordLabel = new JLabel("Password");
        JLabel messageLabel = new JLabel();
        JLabel attemptsLabel = new JLabel("0 attempts");

        JTextField usernameTf = new JTextField(15);
        JTextField passwordTf = new JTextField(15);

        JButton loginButton = new JButton("Login");
        JButton clearButton = new JButton("Clear");
    }
}
```

```
GridBagLayout layout = new GridBagLayout();
GridBagConstraints gbc = new GridBagConstraints();

panel.setLayout(layout);

gbc.gridx = 0;
gbc.gridy = 0;
gbc.gridwidth = 2;
panel.add(messageLabel, gbc);

gbc.gridwidth = 1;
gbc.gridx = 0;
gbc.gridy = 1;
panel.add(usernameLabel, gbc);

gbc.gridx = 1;
gbc.gridy = 1;
panel.add(usernameTf, gbc);

gbc.gridx = 0;
gbc.gridy = 2;
panel.add(passwordLabel, gbc);

gbc.gridx = 1;
gbc.gridy = 2;
panel.add(passwordTf, gbc);
```

EXAMPLE

```
gbc.gridx = 1;
gbc.gridy = 3;

panel.add(loginButton, gbc);

gbc.gridx = 1;
gbc.gridy = 4;
panel.add(clearButton, gbc);

gbc.gridx = 1;
gbc.gridy = 5;
panel.add(attemptsLabel, gbc);

frame.add(panel);
```

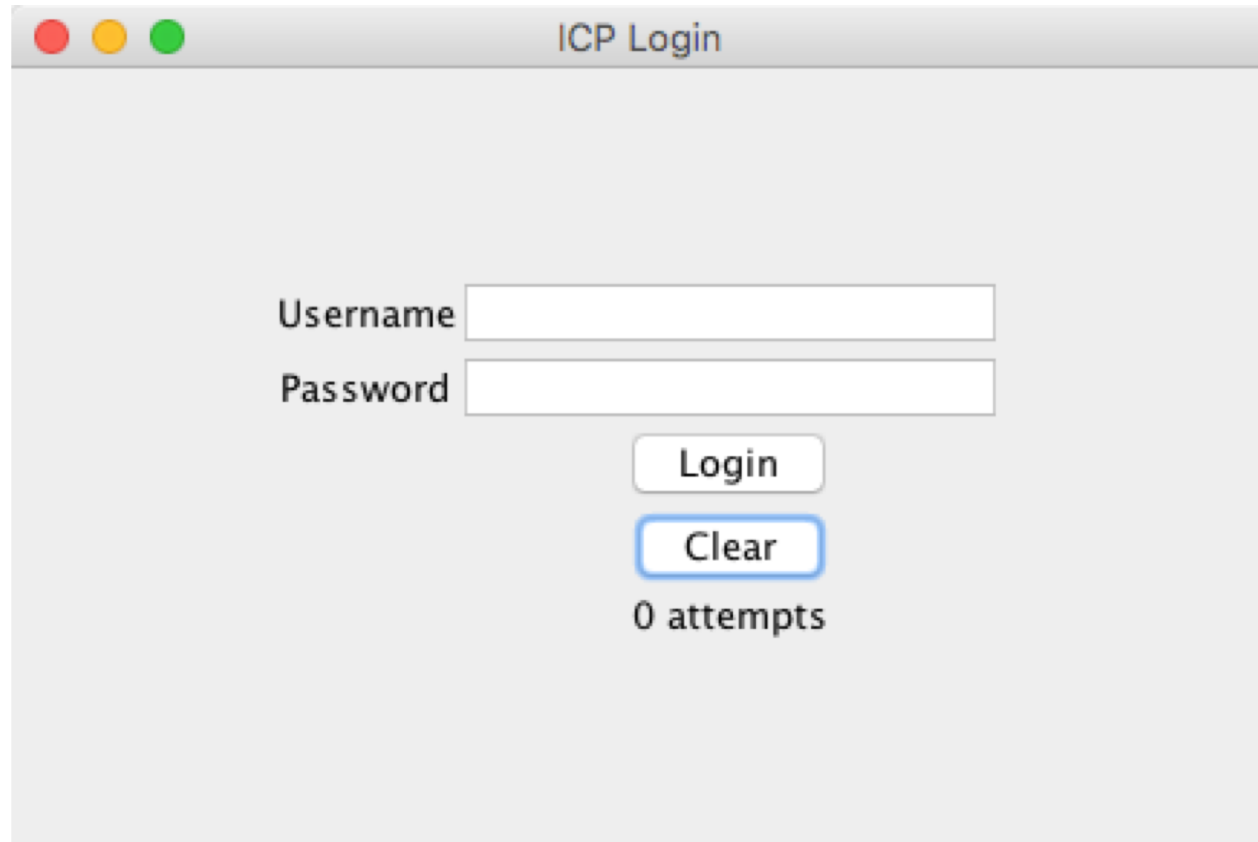
EXAMPLE

```
loginButton.addActionListener(new ActionListener() {  
  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        if (usernameTf.getText().equals("") || passwordTf.getText().equals("")) {  
            messageLabel.setText("Empty username or password.");  
        } else if (usernameTf.getText().equals("admin") &&  
            passwordTf.getText().equals("admin")) {  
            messageLabel.setText("Congratulations. Login Successful!");  
        } else {  
            messageLabel.setText("Username or password is incorrect.");  
            attemptsLabel.setText(++counter + " attempts");  
        }  
    }  
});
```

EXAMPLE

```
clearButton.addActionListener(new ActionListener() {  
  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        usernameTf.setText("");  
        passwordTf.setText("");  
        counter = 0;  
        messageLabel.setText("");  
        attemptsLabel.setText("0 attempts");  
    }  
});  
  
}  
  
public static void main(String[] args) {  
    new MyGridBagLayout();  
}  
  
}
```

OUTPUT



A screenshot of a macOS-style window titled "ICP Login". The window has a light gray background and a title bar with three colored window control buttons (red, yellow, green) on the top left. The main content area contains two text input fields, one for "Username" and one for "Password", stacked vertically. Below the password field are two buttons: "Login" and "Clear". The "Clear" button is highlighted with a blue border. Below the buttons, the text "0 attempts" is displayed.

ICP Login

Username

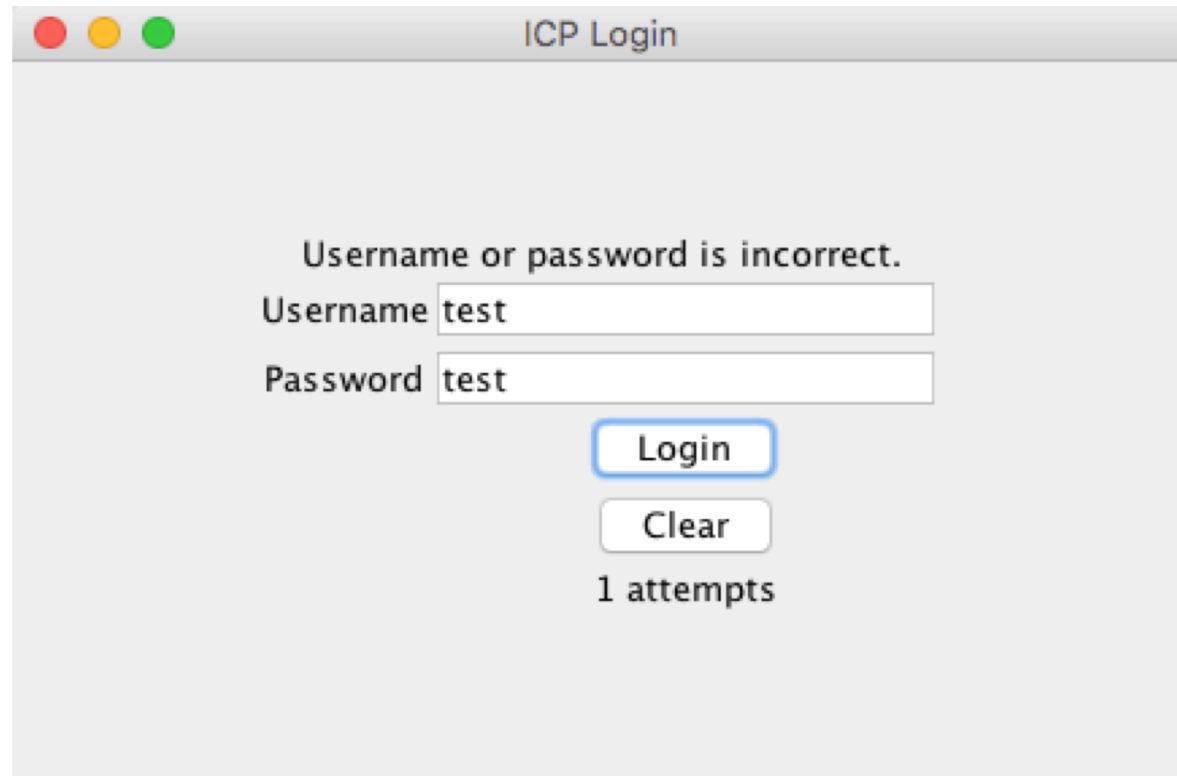
Password

Login

Clear

0 attempts

OUTPUT



ICP Login

Username or password is incorrect.

Username

Password

1 attempts

The image shows a graphical user interface for an 'ICP Login' system. It features a title bar with standard window controls (red, yellow, green buttons) and the text 'ICP Login'. The main area has a light gray background. A message 'Username or password is incorrect.' is displayed in a dark font. Below this message are two text input fields: 'Username' containing 'test' and 'Password' containing 'test'. Under the input fields are two buttons: 'Login' (highlighted with a blue border) and 'Clear'. At the bottom, the text '1 attempts' is shown.

OUTPUT



A screenshot of a web application window titled "ICP Login". The window has a light gray background and a standard macOS-style title bar with red, yellow, and green window control buttons. The main content area displays a success message: "Congratulations. Login Successfull!". Below this message are two input fields: "Username" with the value "admin" and "Password" with the value "admin". Below the input fields are two buttons: "Login" (highlighted with a blue border) and "Clear". At the bottom, it says "1 attempts".

ICP Login

Congratulations. Login Successfull!

Username admin

Password admin

Login

Clear

1 attempts

GRIDBAGCONSTRAINTS

gridx,gridy	Specify the row and column from top to bottom and from left to right starting from zero. For example gridx = 0, gridy = 0 is the top left cell of the grid.
gridwidth, gridheight	Specify number of rows (gridheight) and columns (gridwidth) to which a component can span. The default value of <i>gridwidth</i> and <i>gridheight</i> is 1.
fill	This property is used to resolve whether and how to resize the component when the component's display region is larger than the component's requested size. The values for fill property are: NONE, VERTICAL, HORIZONTAL, VERTICAL and BOTH.
ipadx, ipady	The ipadx, ipady property are used to set the internal padding of the component.
insets	The insets property is used to set the external padding of the component.
anchor	The anchor property is used to determine where to place the component when the component is smaller than display area in the container. Valid values (defined as GridBagConstraints constants) are CENTER (the default), PAGE_START, PAGE_END, LINE_START, LINE_END, FIRST_LINE_START, FIRST_LINE_END, LAST_LINE_END, and LAST_LINE_START.

GRIDBAGLAYOUT

- Unlike the grid layout manager, the gridbag layout manager does not automatically set every cell to exactly the same size.
- The width of any given column is determined by the width of the widest component in that column.
- Similarly, the height of any given row is determined by the height of the tallest component in that row.
- Another way in which the gridbag layout differs from the grid layout, is that a single component can span two or more columns and/or two or more rows.

ARRAYLIST

- ArrayList in Java is used to store dynamically sized collection of elements.
- Contrary to Arrays that are fixed in size, an ArrayList grows its size automatically when new elements are added to it.

ADDING VALUES IN ARRAYLIST

```
public class College {  
    private ArrayList<String> students;  
  
    public College() {  
        this.students = new ArrayList<>();  
    }  
  
    public void addStudent(String name) {  
        this.students.add(name);  
    }  
  
    public void printStudents() {  
        for (String student : students) {  
            System.out.println("Name: " + student);  
        }  
    }  
}
```

ADDING VALUES IN ARRAYLIST

```
public static void main(String[] args) {  
    College college = new College();  
    Scanner scanner = new Scanner(System.in);  
  
    for (int i = 0; i < 5; i++) {  
  
        System.out.println("Enter name");  
        String name = scanner.nextLine();  
  
        college.addStudent(name);  
    }  
    college.printStudents();  
}
```

STUDENT CLASS

```
public class Student {  
    private String name;  
    private int rollNo;  
  
    public String getName() {  
        return name;  
    }  
  
    public void setName(String name) {  
        this.name = name;  
    }  
  
    public int getRollNo() {  
        return rollNo;  
    }  
  
    public void setRollNo(int rollNo) {  
        this.rollNo = rollNo;  
    }  
}
```


COLLEGE CLASS

```
public class College {  
    private ArrayList<Student> students;  
  
    public College() {  
        this.students = new ArrayList<>();  
    }  
  
    public void addStudent(Student student) {  
        this.students.add(student);  
    }  
  
    public void printStudents() {  
        for (Student student : students) {  
            System.out.println("Name: " + student.getName() + ", Roll No: " +  
student.getRollNo());  
        }  
    }  
}
```

COLLEGE CLASS

```
public static void main(String[] args) {  
    College college = new College();  
    Scanner scanner = new Scanner(System.in);  
  
    for (int i = 0; i < 5; i++) {  
  
        System.out.println("Enter name");  
        String name = scanner.nextLine();  
        System.out.println("Enter roll no");  
        int rollNo = scanner.nextInt();  
        scanner.nextLine();  
  
        Student student = new Student();  
        student.setName(name);  
        student.setRollNo(rollNo);  
  
        college.addStudent(student);  
    }  
    college.printStudents();  
}
```

THANK YOU!

Any questions?