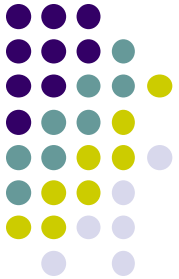
ADVANCED JAVA PROGRAMMING (AJP)

Teaching Scheme			Credit (L+T+P)	Examination Scheme													
				Theory						Practical							
L	T	P		Paper Hrs.	ESE		PA		Total		ESE		PA		Total		
					Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	
3	1	2	6	90 Min	70*#	28	30*	00	100	40	25#	10	25	10	50	20	

Couse Outcomes

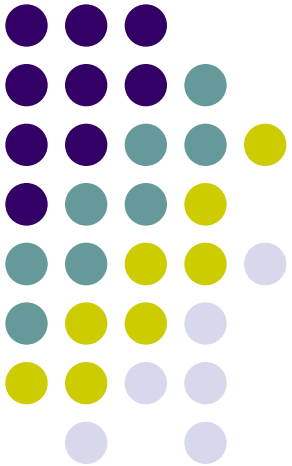


- **Develop programs using GUI Framework(AWT and Swing)**
- **Handle events of AWT and Swings Components.**
- **Develop programs to handle events in Java Programming.**
- **Develop Java Programs using networking concepts.**
- **Develop programs using database.**
- **Develop programs using servlets.**

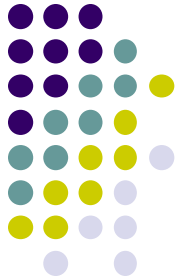
Vidyalankar Polytechnic

INTRODUCTION TO ABSTRACT WINDOW TOOLKIT (AWT)

12 Marks

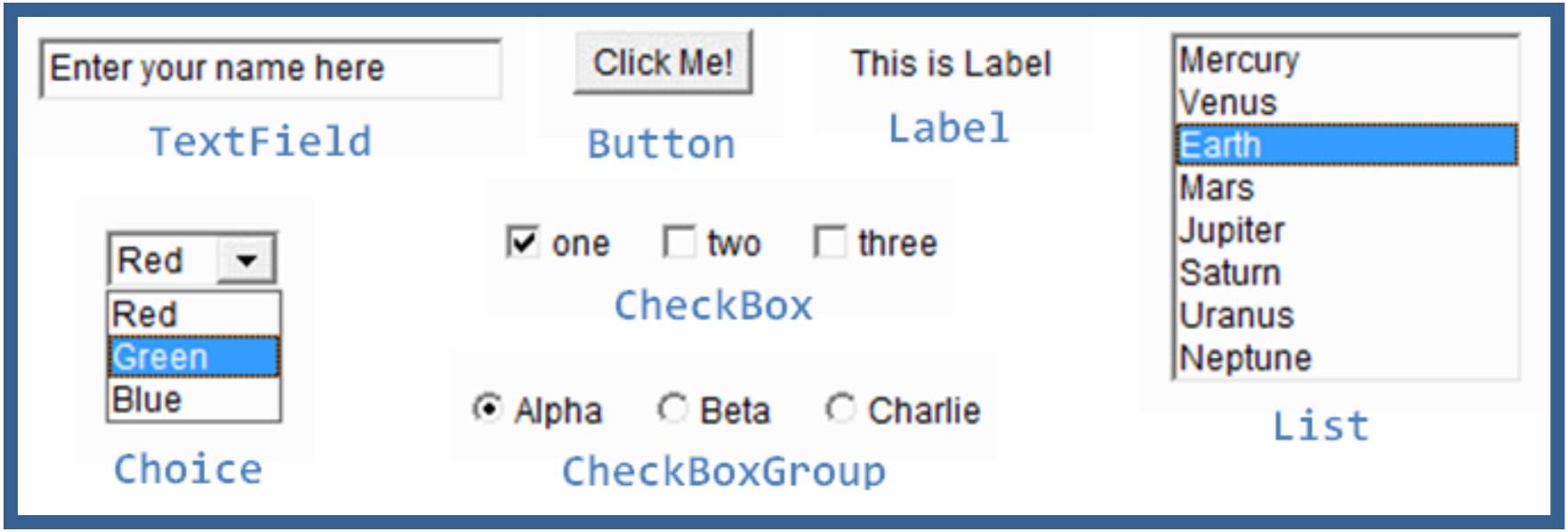


Unit Outcomes

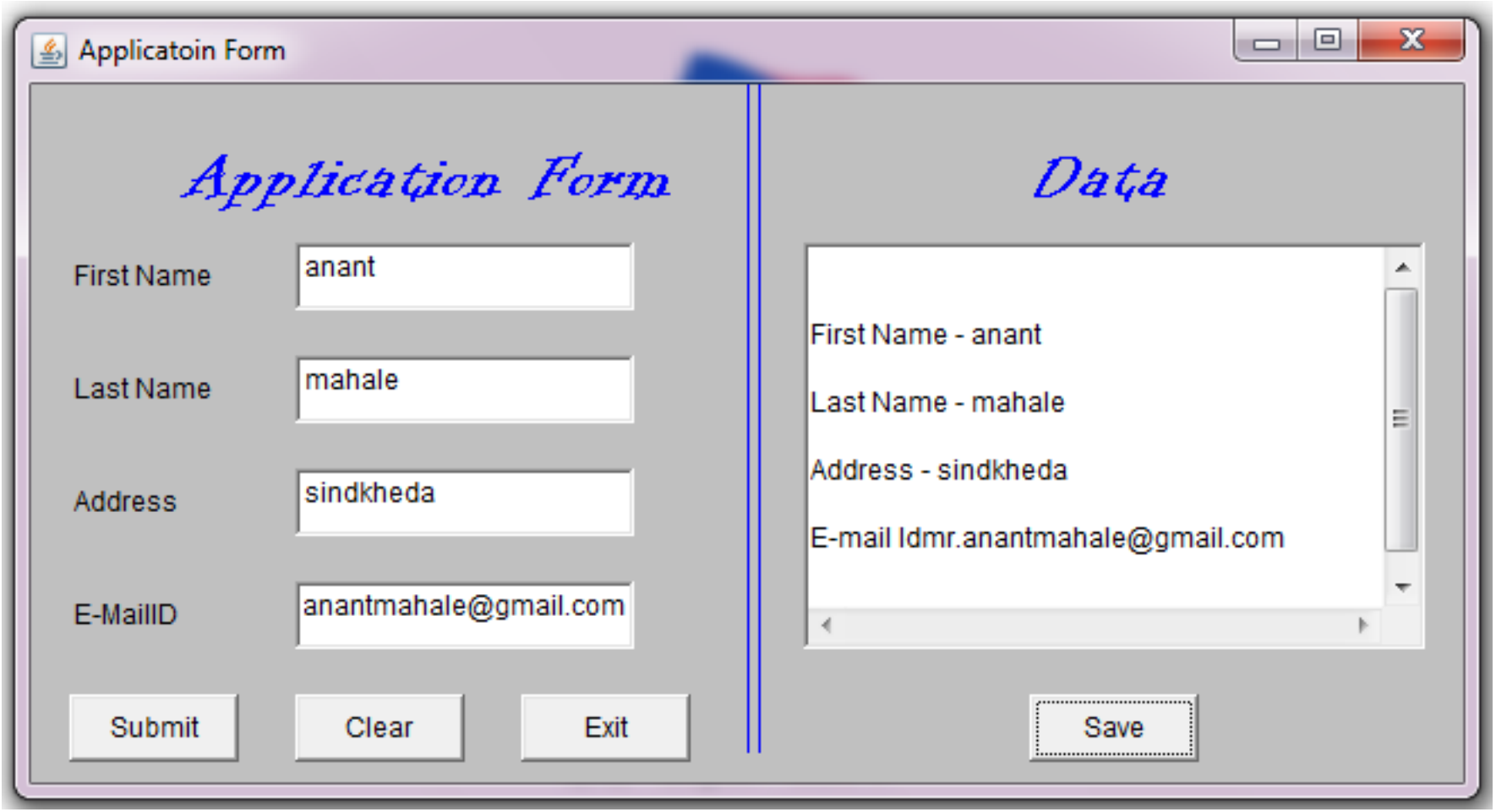


- Develop **GUI** using different AWT components for the given problem.
- Create Frame window with the specified AWT components.
- Arrange the GUI components using specified layout manager
- Develop a program using menu and Dialog Boxes for the given problem.

Example 1


Enter your name here
TextField
Click Me!
Button
This is Label
Label
Mercury
Venus
Earth
Mars
Jupiter
Saturn
Uranus
Neptune
List
Red
Red
Green
Blue
Choice
one
two
three
CheckBox
Alpha
Beta
Charlie
CheckBoxGroup

Example 2


Applicatoin Form
Application Form
First Name
anant
Last Name
mahale
Address
sindkheda
E-MailID
anantmahale@gmail.com
Submit
Clear
Exit
Data
First Name - anant
Last Name - mahale
Address - sindkheda
E-mail Idmr.anantmahale@gmail.com
Save

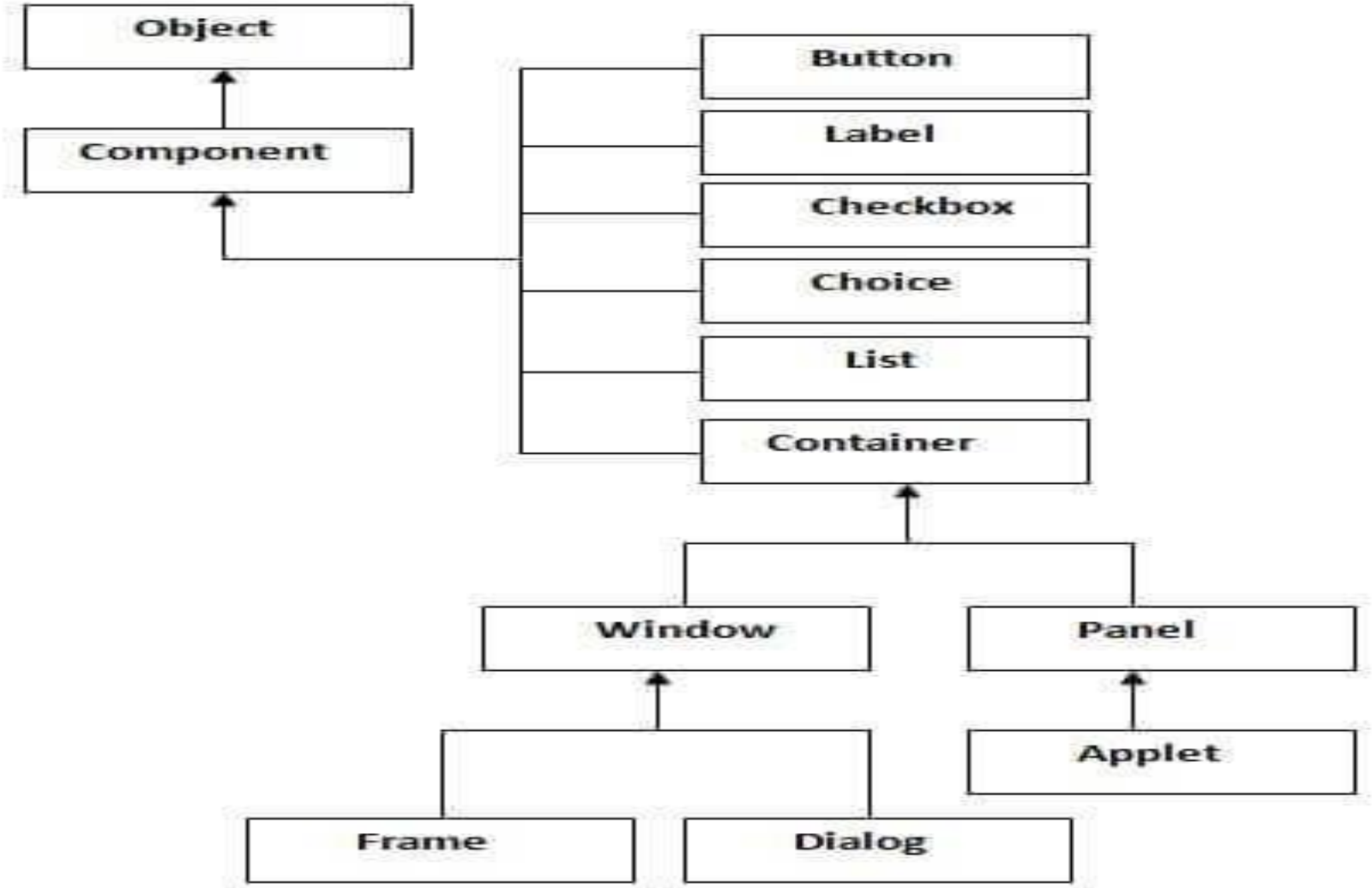
Introduction to AWT

- AWT Package
  - A complete set of UI
  - Support for UI containers
  - An event system
  - Techniques and code space for laying out UI components in such a way that permits platform independent UI design

Working with Windows and AWT

- An applet is a window based program.
- An applet waits until an event occurs. AWT notifies the applet about an event by calling an event handler.
- Applet must perform specific actions in response to events.

Window Fundamentals



Working with Frame Window

- **Frame's constructors:**

  Frame( )

  Frame(String *title)*

- **Setting the Frame's Size**

  void setSize(int *newWidth*, *int newHeight)*

  void setSize(Dimension *newSize)*

  Dimension getSize( )

- **Hiding and Showing a Window**

  void setVisible(boolean *visibleFlag)*

Working with Frame Window (cont..)

- Setting a Window's Title

  void setTitle(String *newTitle)*

- Closing a Frame Window

  setVisible(false)

  To perform window-close event we must implement **windowsClosing()** method of **WindowListener**

Frame Example 1 Frame Example 2

Frame Window in an Applet

Example

HTML Page

Using AWT Controls, Layout Manager And Menus

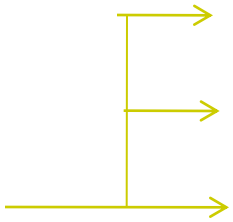
- Types of Control
  - TextFields
  - Push buttons
  - Check boxes
  - Choice lists
  - Lists
  - Scroll bars
  - Text editing
- Adding and Removing Controls

  Component add(Component Obj)

  void remove(Component Obj)

Labels

- A *Label is an object of type* ***Label, and it contains a*** string, which it displays.
- **Label defines the following constructors:**
  - Label( )
  - Label(String *str)*
  - Label(String *str, int how)*



  - Label.LEFT
  - Label.RIGHT
  - Label.CENTER
- These methods are shown here:
  - void setText(String *str)*
  - String getText( )
  - void setAlignment(int *how)*
  - int getAlignment( )

Program

HTML Page

Buttons

- A *push button is a component that* contains a label and that generates an event when it is pressed
- **Button defines the following constructors:**
  - Button( )
  - Button(String *s*
- Methods are shown here:
  - void setLabel(String *str)*
  - String getLabel( )

Program HTML Page

TextField

- Text fields allow the user to enter strings and to edit the text using the arrow keys, cut and paste keys, and mouse selections.
- **TextField defines the following constructors:**

  TextField( )

  TextField(int *numChars)*

  TextField(String str*)*

  TextField(String *str, int numChars)*

- These methods are shown here:

  void setText(String *str)*

  String getText( )

  String getSelectedText( )

  Void setEditable(boolean canEdit)

  int setEchoChar(char ch)

Program

HTML Page

TextArea

- Sometimes a single line of text input is not enough for a given task. To handle these situations, the AWT includes a simple multiline editor called **TextArea.**

  TextArea( )

  TextArea(int *numLines, int numChars)*

  TextArea(String *str)*

  TextArea(String *str, int numLines, int numChars)*

  TextArea(String *str, int numLines, int numChars, int sBars)*

  SCROLLBARS_BOTH SCROLLBARS_NONE

  SCROLLBARS_HORIZONTAL_ONLY
  SCROLLBARS_VERTICAL_ONLY

Check Boxes

- A *check box is a control that is used to turn an option on or off.*
- **Checkbox supports these constructors:**

  Checkbox( )

  Checkbox(String *str)*

  Checkbox(String *str, boolean on)*

- These methods are as follows:

  boolean getState( )

  void setState(boolean *on)*

  String getLabel( )

  void setLabel(String *str)*

Program

HTML Page

CheckBoxes Group

- It is possible to create a set of mutually exclusive check boxes in which one and only one check box in the group can be checked at any one time. These check boxes are often called *radio buttons*
- *Constructor*

  Checkbox(String *str, boolean on, CheckboxGroup cbGroup)*

  Checkbox(String *str, CheckboxGroup cbGroup, boolean on)*
- Methods

  Checkbox getSelectedCheckbox( )

  void setSelectedCheckbox(Checkbox *which)*

Program

HTML Page

Choice Controls

- The **Choice class is used to create a *pop-up list of items from which the user may*** choose. Thus, a **Choice control is a form of menu.**
- Methods

  void add(String *name)*

  String getSelectedItem( )

  int getSelectedIndex( )

  int getItemCount( )

  void select(int *index)*

  void select(String *name)*

  String getItem(int *index)*

Program

HTML Page

Lists

- The **List class provides a compact, multiple-choice, scrolling selection list.**
- **List provides these constructors:**

  List( )

  List(int *numRows)*

  List(int *numRows, boolean multipleSelect)*

- Methods

  void add(String *name)*

  void add(String *name, int index)*

  String getSelectedItem( )

  int getSelectedIndex( )

String[ ] getSelectedItems( )

int[ ] getSelectedIndexes( )

int getItemCount( )

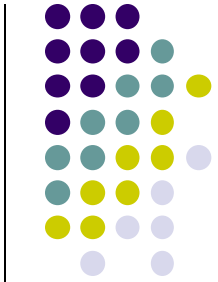
void select(int *index)*

String getItem(int *index)*

Program

HTML Demo

Layout Manager



- Layout Manager is set by

  setLayout()

- Default Layout Manager

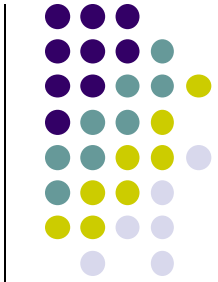
  Panel, Applet → Flow Layout

  Frame → Border Layout

- General Form

  void setLayout(LayoutManager layObj)

Flow Layout

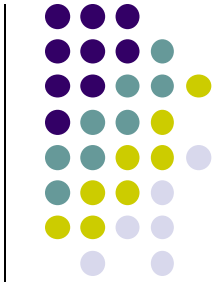


- Constructors
  - FlowLayout()
  - FlowLayout(int how) i.e. LEFT, CENTER, RIGHT
  - FlowLayout(int how, int horz, int vert)

Program

HTML Page

Border Layout



- Constructors defined by **BorderLayout:**

  BorderLayout( )

  BorderLayout(int *horz, int vert)*

- **BorderLayout defines the following constants that specify the regions:**

BorderLayout.CENTER BorderLayout.SOUTH

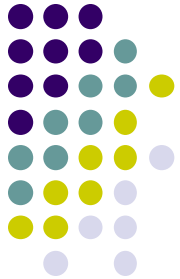
BorderLayout.EAST BorderLayout.WEST

BorderLayout.NORTH

void add(Component *compObj, Object region);*

Program *HTML Page*

Grid Layout



- The constructors supported by **GridLayout are shown here:**

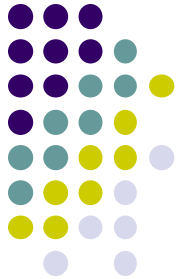
  GridLayout( )

  GridLayout(int *numRows, int numColumns)*

  GridLayout(int *numRows, int numColumns, int horz, int vert)*

Program *HTML Page*

Card Layout



- **CardLayout provides these two constructors:**

  CardLayout( )

  CardLayout(int *horz, int vert)*

  void first(Container *deck)*

  void last(Container *deck)*

  void next(Container *deck)*

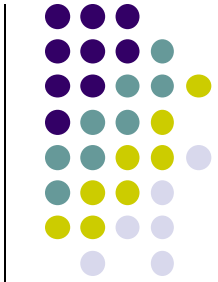
  void previous(Container *deck)*

  void show(Container *deck, String cardName)*

Program

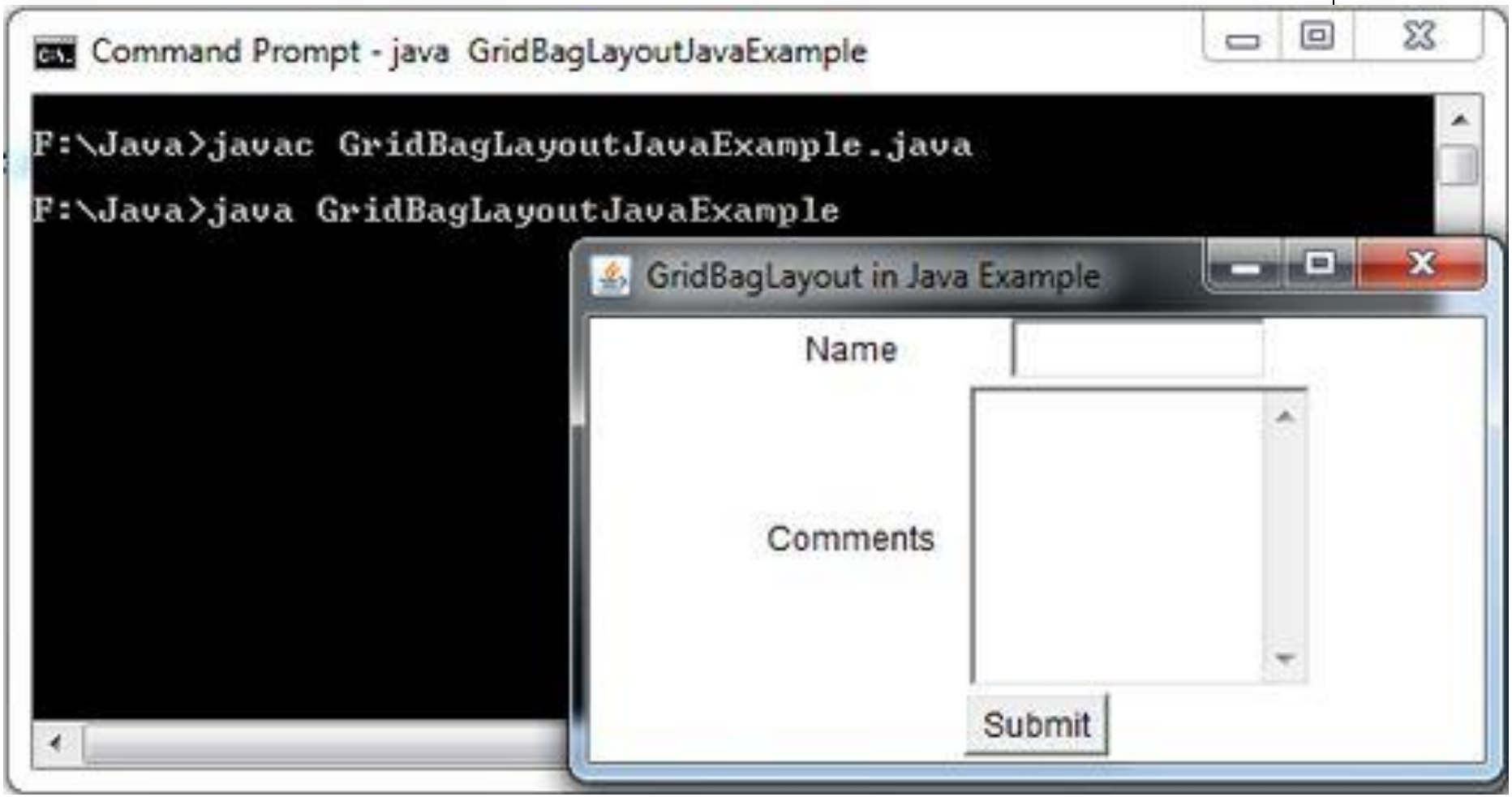
HTML Page

GridBagLayout



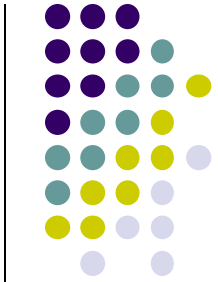
- GridBagLayout layout manager is the most powerful and flexible of all the predefined layout managers but more complicated to use
- Unlike GridLayout where the component are arranged in a rectangular grid and each component in the container is forced to be the same size, in GridBagLayout, components are also arranged in rectangular grid but can have different sizes and can occupy multiple rows or columns.
- A GridBagLayout layout manager requires a lot of information to know where to put a component in a container.
- A helper class called GridBagConstraints provides all this information. It specifies constraints on how to position a component, how to distribute the component and how to resize and align them.

Variable	Role
gridx and gridy	These contain the coordinates of the origin of the grid. They allow a at a specific position of a component positioning. By default, they have GrigBagConstraint.RELATIVE value which indicates that a component can be stored to the right of previous
gridwidth, gridheight	Define how many cells will occupy component (height and width). by The default is 1. The indication is relative to the other components of the line or the column. TheGridBagConstraints.REMAINDER value specifies that the nextcomponent inserted will be the last of the line or the current column. the value GridBagConstraints.RELATIVE up the component after the last component of a row or column.
fill	Defines the fate of a component smaller than the grid cell. GridBagConstraints.NONE retains the original size: Default GridBagConstraints.HORIZONTAL expanded horizontally GridBagConstraints.VERTICAL GridBagConstraints.BOTH expanded vertically expanded to the dimensions of the cell
ipadx, ipady	Used to define the horizontal and vertical expansion ofcomponents. not works if expansion is required by fill. The default value is (0,0).
anchor	When a component is smaller than the cell in which it is inserted, it can be positioned using this variable to define theside from which the control should be aligned within the cell.Possible variables NORTH, NORTHWEST, NORTHEAST, SOUTH, SOUTHWEST, SOUTHEAST, WEST and EAST
weightx, weighty	Used to define the distribution of space in case of change of dimension


Command Prompt - java GridBagLayoutJavaExample
F:\Java>javac GridBagLayoutJavaExample.java
F:\Java>java GridBagLayoutJavaExample
GridBagLayout in Java Example
Name
Comments
Submit

Program

Dialog Boxes



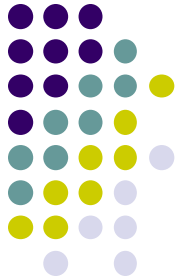
- Dialog(Frame *parentWindow*, *boolean mode)*
- Dialog(Frame *parentWindow*, *String title*, *boolean mode)*

Program

Frame Example

HTML File

Menu Bars and Menu



Menu(String *optionName)*

Menu(String *optionName, boolean removable)*

MenuItem(String *itemName)*

CheckboxMenuItem(String *itemName)*

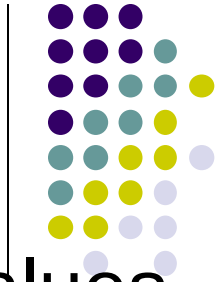
CheckboxMenuItem(String *itemName, boolean on)*

boolean getState( )

void setState(boolean *checked)*

Program

Scroll Bars



- *Scroll bars* are used to select continuous values between a specified minimum and maximum.
- Scroll bars may be oriented horizontally or vertically.

Scrollbar( ) → Scrollbar.HORIZONTAL

Scrollbar( ) → Scrollbar.VERTICAL

Scrollbar(int *style*)

Scrollbar(int *style,* int *initialValue*, int *thumbSize*, int *min*, int *max*)

Methods

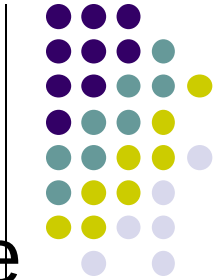
int getValue( ), void setValue(int *newValue*), int getMinimum( )

int getMaximum( ), void setUnitIncrement(int *newIncr*)

void setBlockIncrement(int *newIncr*)

Program

FileDialog



- FileDialog a built-in dialog box that lets the user specify a file.

FileDialog(Frame *parent*, String *boxName*)

FileDialog(Frame *parent*, String *boxName*, int *how*)

FileDialog(Frame *parent*)

- FileDialog.LOAD
- FileDialog.SAVE

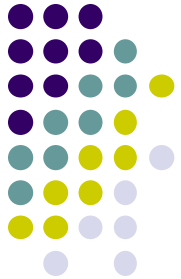
FileDialog( ) provides methods that allow you to determine the name of the file and its path as selected by the user.

String getDirectory( )

String getFile( )

Program

Difference between AWT & Swing



No.	Java AWT	Java Swing
1)	AWT components are **platform-dependent**.	Java swing components are **platform-independent**.
2)	AWT components are **heavyweight**.	Swing components are **lightweight**.
3)	AWT **doesn't support pluggable look and feel**.	Swing **supports pluggable look and feel**.
4)	AWT provides **less components** than Swing.	Swing provides **more powerful components** such as tables, lists, scrollpanes, colorchooser, tabbedpane etc.
5)	AWT **doesn't follows MVC** (Model View Controller) where model represents data, view represents presentation and controller acts as an interface between model and view.	Swing **follows MVC**.

- https://forms.office.com/Pages/ResponsePage.aspx?id=QEUap70T20yz690UXwrJwOOKzgKlZI9Fg6Oncam09LtUMVhEUlpMMEtZVUMyS1JBQVpIRE5RRjAxUi4u