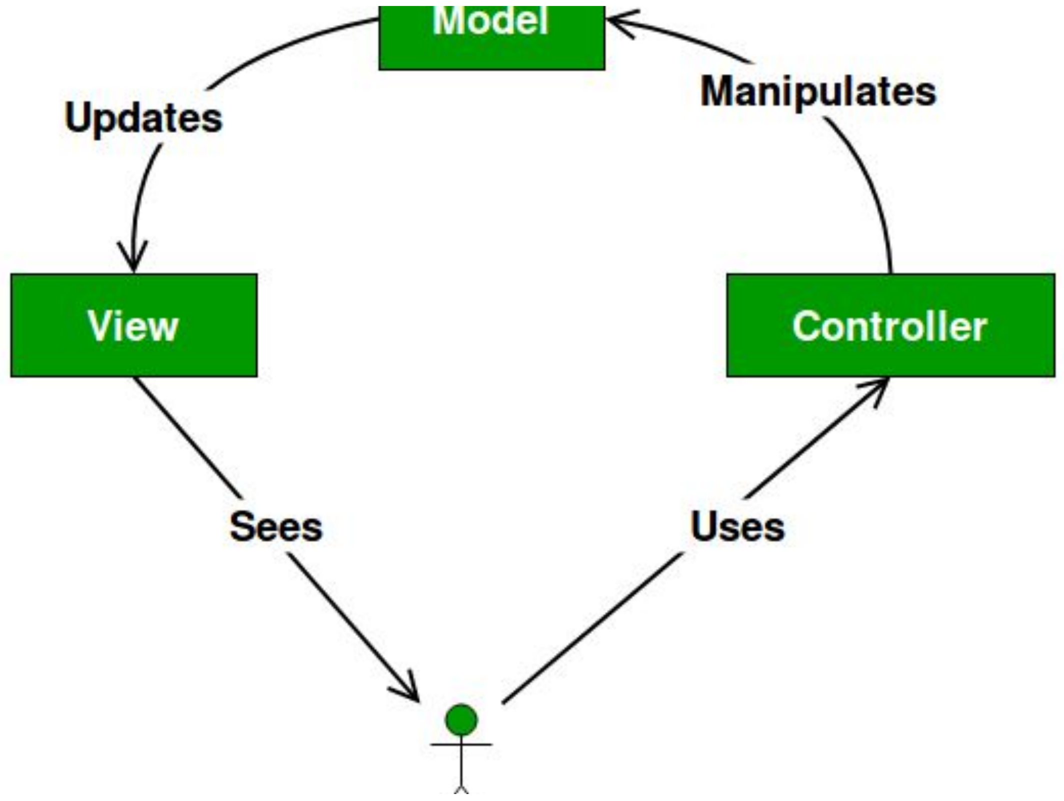
MVC Architecture

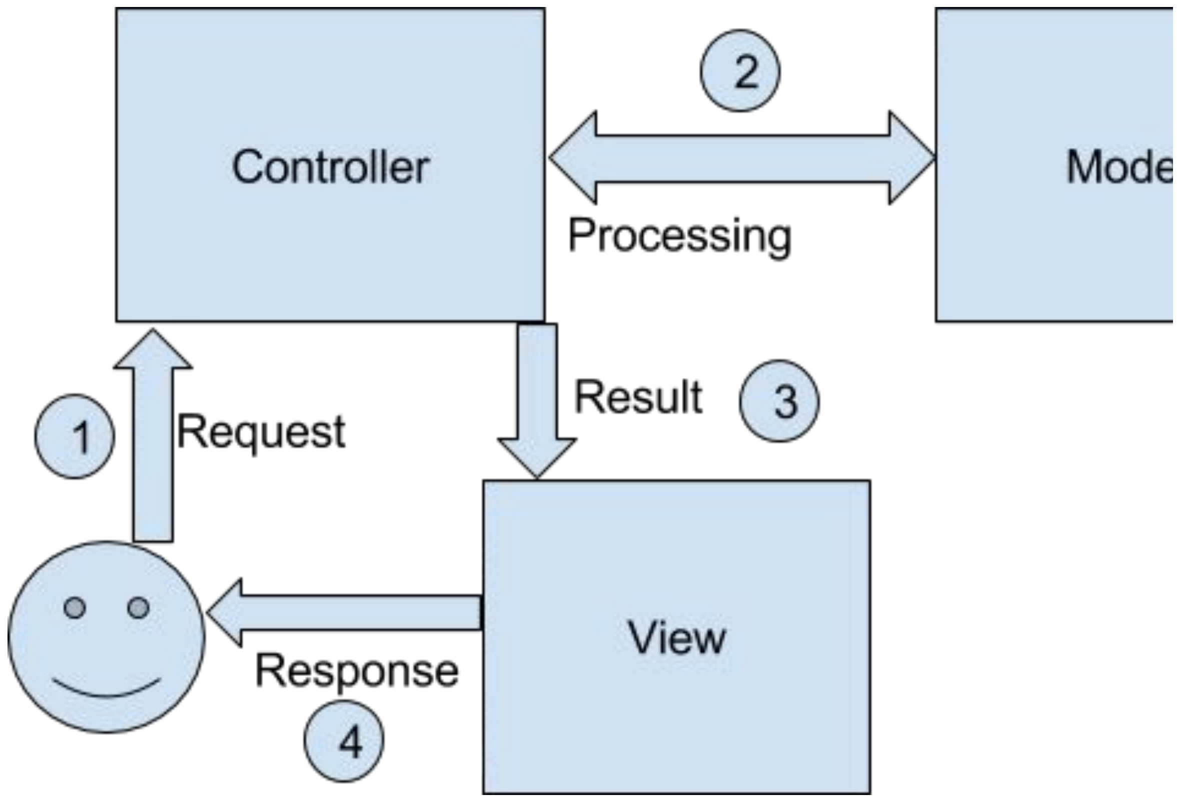
Unit V

Model-View-Controller (MVC)

- The **MVC** is an architectural pattern that separates an application into three main logical components: the **model**, the view, and the controller.
- Each of these components are built to handle specific development aspects of an application.
- MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects.

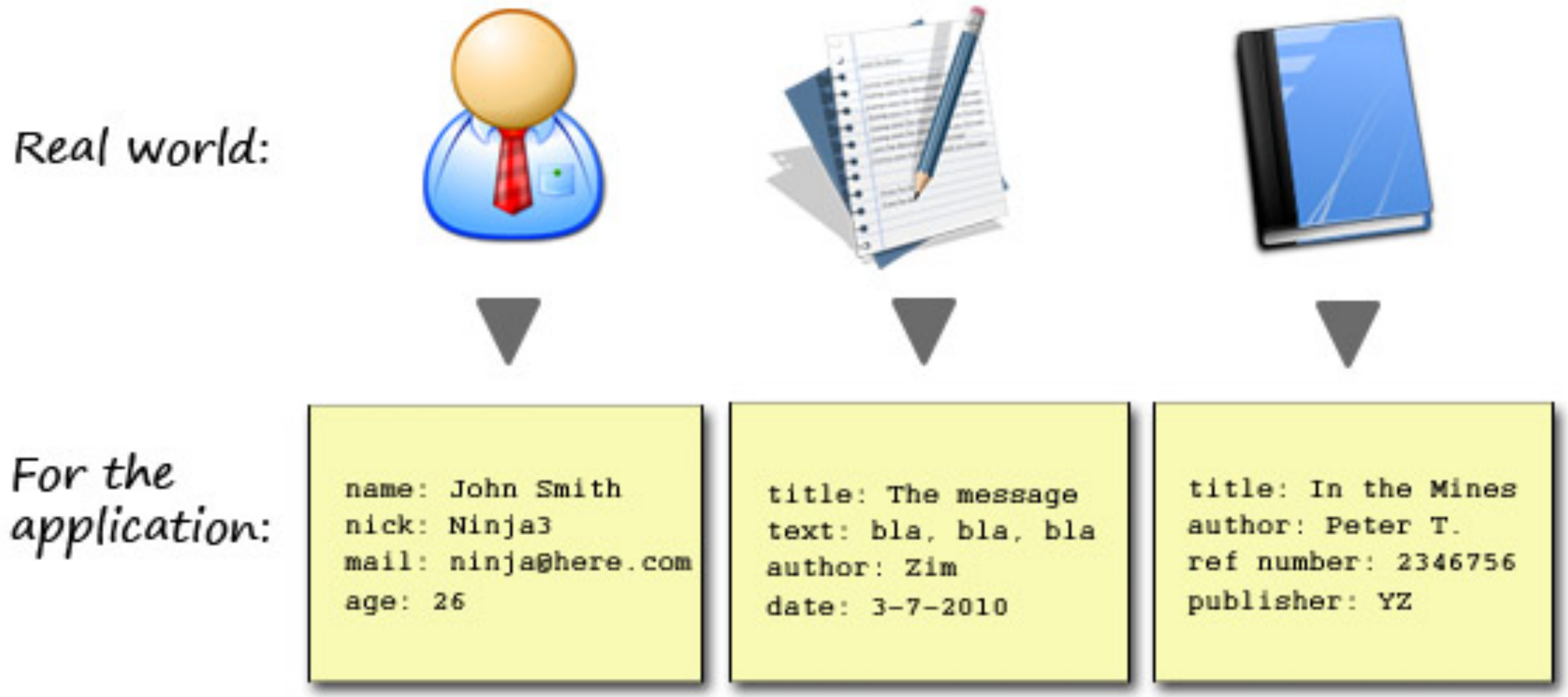
MVC Flow


Model
Updates
Manipulates
View
Controller
Sees
Uses


Controller
2
Mode
Processing
Result
3
1
Request
View
Response
4

Model

- The Model component corresponds to all the data-related logic that the user works with.
- This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data.
- The model is quite simply the data for our application.
- The data is “modelled” in a way it’s easy to store, retrieve, and edit.
- The model is how we apply rules to our data, which eventually represents the concepts our application manages.
- For example, a Customer object will retrieve the customer information from the database, manipulate it and update it data back to the database or use it to render data.


Real world:

For the application:

name: John Smith
nick: Ninja3
mail: ninja@here.com
age: 26

title: The message
text: bla, bla, bla
author: Zim
date: 3-7-2010

title: In the Mines
author: Peter T.
ref number: 2346756
publisher: YZ

View

- The View component is used for all the UI logic of the application.
- For example, the Customer view will include all the UI components such as text boxes, dropdowns, etc. that the final user interacts with.

Controller

- Controllers act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output.
- The primary function of a controller is to call and coordinate with the model to fetch any necessary resources required to act.
- Usually, on receiving a user request, the controller calls the appropriate model for the task at hand.
- For example, the Customer controller will handle all the interactions and inputs from the Customer View and update the database using the Customer Model.
- The same controller will be used to view the Customer data.

Advantages

- Multiple developers can work simultaneously on the model, controller and views.
- Offers improved scalability, that supplements the ability of the application to grow
- As components have a low dependency on each other, they are easy to maintain
- A model can be reused by multiple views which provides reusability of code
- Adoption of MVC makes an application more expressive and easy to understand
- Extending and testing of the application becomes easy

Java Web components - **AWT in Java**

- Abstract Window Toolkit acronymed as AWT is a toolkit of classes in Java which helps a programmer to develop Graphics and Graphical User Interface components.
- The AWT API in Java primarily consists of a comprehensive set of classes and methods that are required for creating and managing the Graphical User Interface(GUI) in a simplified manner.
- AWT is a platform dependent API for creating Graphical User Interface (GUI) for java programs.

Why AWT is platform dependent?

- Java AWT calls native platform (Operating systems) subroutine for creating components such as textbox, checkbox, button etc.
- For example an AWT GUI having a button would have a different look and feel across platforms like windows, Mac OS & Unix, this is because these platforms have different look and feel for their native buttons and AWT directly calls their native subroutine that creates the button.
- For example if you are instantiating a text box in AWT that means you are actually asking OS to create a text box for you.

Features of AWT in Java

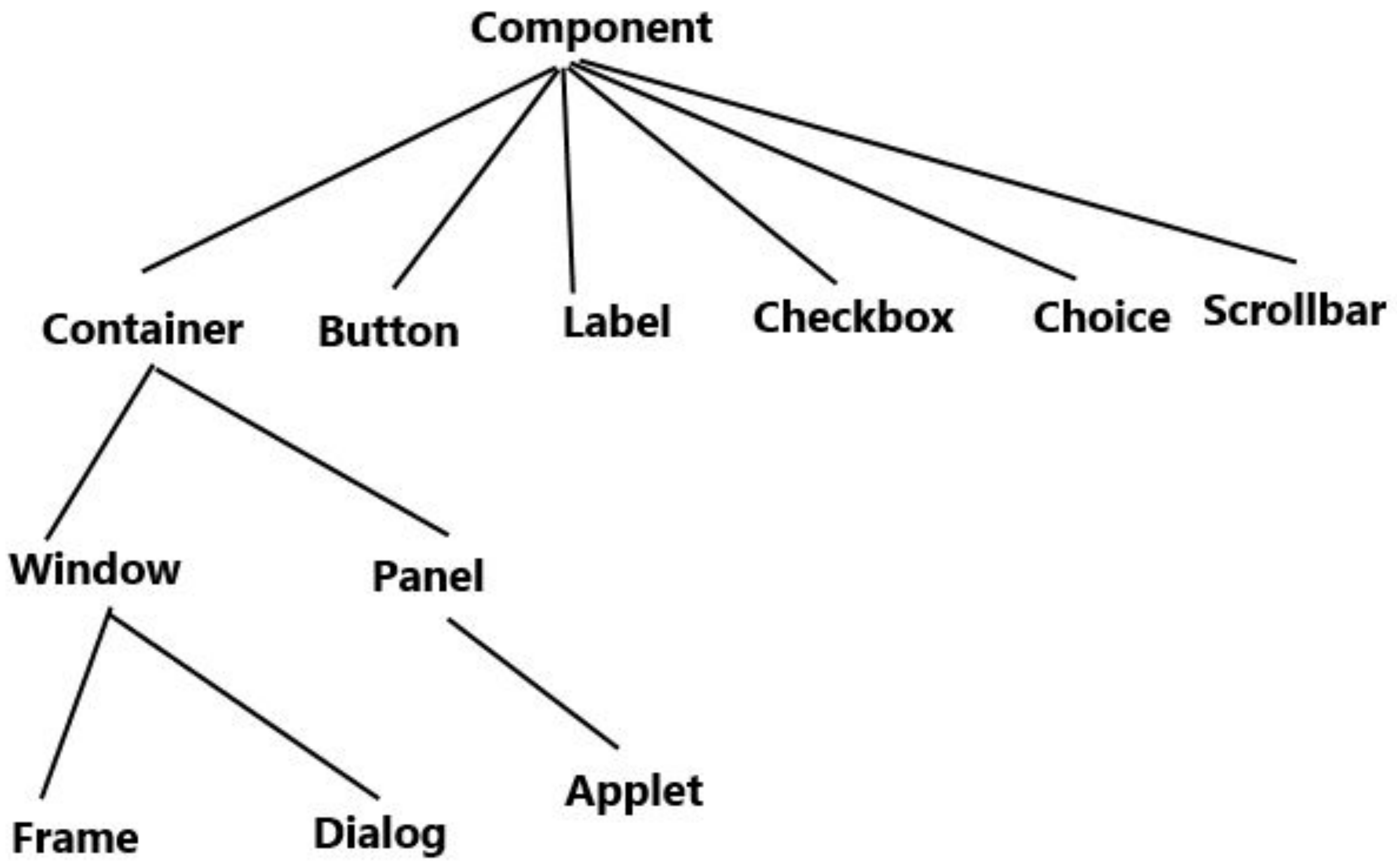
- AWT is a set of native user interface components
- It is based upon a robust event-handling model
- It provides Graphics and imaging tools, such as shape, color, and font classes
- AWT also avails layout managers which helps in increasing the flexibility of the window layouts
- Data transfer classes are also a part of AWT that helps in cut-and-paste through the native platform clipboard
- Supports a wide range of libraries that are necessary for creating graphics for gaming products, banking services, educational purposes, etc.

AWT UI Aspects

Any UI will be made of three entities:

- **UI elements**: These refers to the core visual elements which are visible to the user and used for interacting with the application. AWT in Java provides a comprehensive list of widely used and common elements.
- **Layouts**: These define how UI elements will be organized on the screen and provide the final look and feel to the GUI.
- **Behavior**: These define the events which should occur when a user interacts with UI elements.

AWT hierarchy



Components and containers

- All the elements like buttons, text fields, scrollbars etc are known as components.
- In AWT we have classes for each component as shown in the above diagram.
- To have everything placed on a screen to a particular position, we have to add them to a container.
- A container is like a screen wherein we are placing components like buttons, text fields, checkbox etc.
- In short a container contains and controls the layout of components.
- A container itself is a component (shown in the above hierarchy diagram) thus we can add a container inside container.

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.

AWT Components - Containers

- Container in Java AWT is a component that is used to hold other components such as text fields, buttons, etc.
- It is a subclass of java.awt.Component and is responsible for keeping a track of components being added.
- There are four types of containers provided by AWT in Java.

Types of containers

- **Window:**
  - An instance of the Window class has no border and no title

- **Dialog:**
  - Dialog class is also a subclass of Window.
  - Dialog class has border and title.
  - An instance of the Dialog class cannot exist without an associated instance of the Frame class.

- **Panel:**
  - Panel does not contain title bar, menu bar or border.
  - It is a generic container for holding components.
  - An instance of the Panel class provides a container to which to add components.

- **Frame:**
  - Frame is a subclass of Window.
  - A frame has title, border and menu bars.
  - It can contain several components like buttons, text fields, scrollbars etc.
  - This is most widely used container while developing an application in AWT. It comes with a resizing canvas.
  - You can create a Java AWT Frame in two ways:
  - By Instantiating Frame class
  - By extending Frame class

Java AWT Button

- A button is basically a control component with a label that generates an event when pushed.
- The **Button** class is used to create a labeled button that has platform independent implementation.
- The application result in some action when the button is pushed.

Sr. no.	Constructor	Description
1.	Button( )	It constructs a new button with an empty string i.e. it has no label.
2.	Button (String text)	It constructs a new button with given string as its label.

Java AWT Label

- The object of the Label class is a component for placing text in a container.
- It is used to display a single line of **read only text**.
- The text can be changed by a programmer but a user cannot edit it directly.
- The java.awt.Component class has following fields:

1. **static int LEFT:** It specifies that the label should be left justified.
2. **static int RIGHT:** It specifies that the label should be right justified.
3. **static int CENTER:** It specifies that the label should be placed in center.

Sr. no.	Constructor	Description
1.	Label()	It constructs an empty label.
2.	Label(String text)	It constructs a label with the given string (left justified by default).
3.	Label(String text, int alignement)	It constructs a label with the specified string and the specified alignment.

Java AWT TextField

- The object of a **TextField** class is a text component that allows a user to enter a single line text and edit it.
- It inherits **TextComponent** class, which further inherits **Component** class.

Sr. no.	Constructor	Description
1.	TextField()	It constructs a new text field component.
2.	TextField(String text)	It constructs a new text field initialized with the given string text to be displayed.
3.	TextField(int columns)	It constructs a new textfield (empty) with given number of columns.
4.	TextField(String text, int columns)	It constructs a new text field with the given text and given number of columns (width).

Java AWT TextArea

- The object of a TextArea class is a multiline region that displays text.
- It allows the editing of multiple line text.
- The text area allows us to type as much text as we want.
- When the text in the text area becomes larger than the viewable area, the scroll bar appears automatically which helps us to scroll the text up and down, or right and left.
- **Fields of TextArea Class**

static int SCROLLBARS_BOTH - It creates and displays both horizontal and vertical scrollbars.

static int SCROLLBARS_HORIZONTAL_ONLY - It creates and displays only the horizontal scrollbar.

static int SCROLLBARS_VERTICAL_ONLY - It creates and displays only the vertical scrollbar.

static int SCROLLBARS_NONE - It doesn't create or display any scrollbar in the text area.

Class constructors:

Sr. no.	Constructor	Description
1.	TextArea()	It constructs a new and empty text area with no text in it.
2.	TextArea (int row, int column)	It constructs a new text area with specified number of rows and columns and empty string as text.
3.	TextArea (String text)	It constructs a new text area and displays the specified text in it.
4.	TextArea (String text, int row, int column)	It constructs a new text area with the specified text in the text area and specified number of rows and columns.
5.	TextArea (String text, int row, int column, int scrollbars)	It construcst a new text area with specified text in text area and specified number of rows and columns and visibility.

Java AWT Checkbox

- The Checkbox class is used to create a checkbox. It is used to turn an option on (true) or off (false).
- Clicking on a Checkbox changes its state from "on" to "off" or from "off" to "on".
- **Checkbox Class Constructors**

Sr. no.	Constructor	Description
1.	Checkbox()	It constructs a checkbox with no string as the label.
2.	Checkbox(String label)	It constructs a checkbox with the given label.
3.	Checkbox(String label, boolean state)	It constructs a checkbox with the given label and sets the given state.
4.	Checkbox(String label, boolean state, CheckboxGroup group)	It constructs a checkbox with the given label, set the given state in the specified checkbox group.
5.	Checkbox(String label, CheckboxGroup group, boolean state)	It constructs a checkbox with the given label, in the given checkbox group and set to the specified state.

Java AWT CheckboxGroup

- The object of CheckboxGroup class is used to group together a set of Checkbox.
- At a time only one check box button is allowed to be in "on" state and remaining check box button in "off" state

Java AWT Choice

- The object of Choice class is used to show popup menu of choices.
- Choice selected by user is shown on the top of a menu.

Sr. no.	Constructor	Description
1.	Choice()	It constructs a new choice menu.

Java AWT List

- The object of List class represents a list of text items.
- With the help of the List class, user can choose either one item or multiple items.
- **AWT List Class Constructors**

Sr. no.	Constructor	Description
1.	List()	It constructs a new scrolling list.
2.	List(int row_num)	It constructs a new scrolling list initialized with the given number of rows visible.
3.	List(int row_num, Boolean multipleMode)	It constructs a new scrolling list initialized which displays the given number of rows.

Java AWT Canvas

- The Canvas class controls and represents a blank rectangular area where the application can draw or trap input events from the user.

Sr. no.	Constructor	Description
1.	Canvas()	It constructs a new Canvas.
2.	Canvas(GraphicConfiguration config)	It constructs a new Canvas with the given Graphic Configuration object.

Java AWT Dialog

- The Dialog control represents a top level window with a border and a title used to take some form of input from the user
- Frame and Dialog both inherits Window class.
- Frame has maximize and minimize buttons but Dialog doesn't have.

Java AWT Panel

- The Panel is a simplest container class. It provides space in which an application can attach any other component.

Advantages of GUI over CUI

- GUI provides graphical icons to interact while the CUI (Character User Interface) offers the simple text-based interfaces.
- GUI makes the application more entertaining and interesting on the other hand CUI does not.
- GUI offers click and execute environment while in CUI every time we have to enter the command for a task.
- New user can easily interact with graphical user interface by the visual indicators but it is difficult in Character user interface.
- GUI offers a lot of controls of file system and the operating system while in CUI you have to use commands which is difficult to remember.
- Windows concept in GUI allow the user to view, manipulate and control the multiple applications at once while in CUI user can control one task at a time.
- GUI provides multitasking environment so as the CUI also does but CUI does not provide same ease as the GUI do.
- Using GUI it is easier to control and navigate the operating system which becomes very slow in command user interface. GUI can be easily customized.