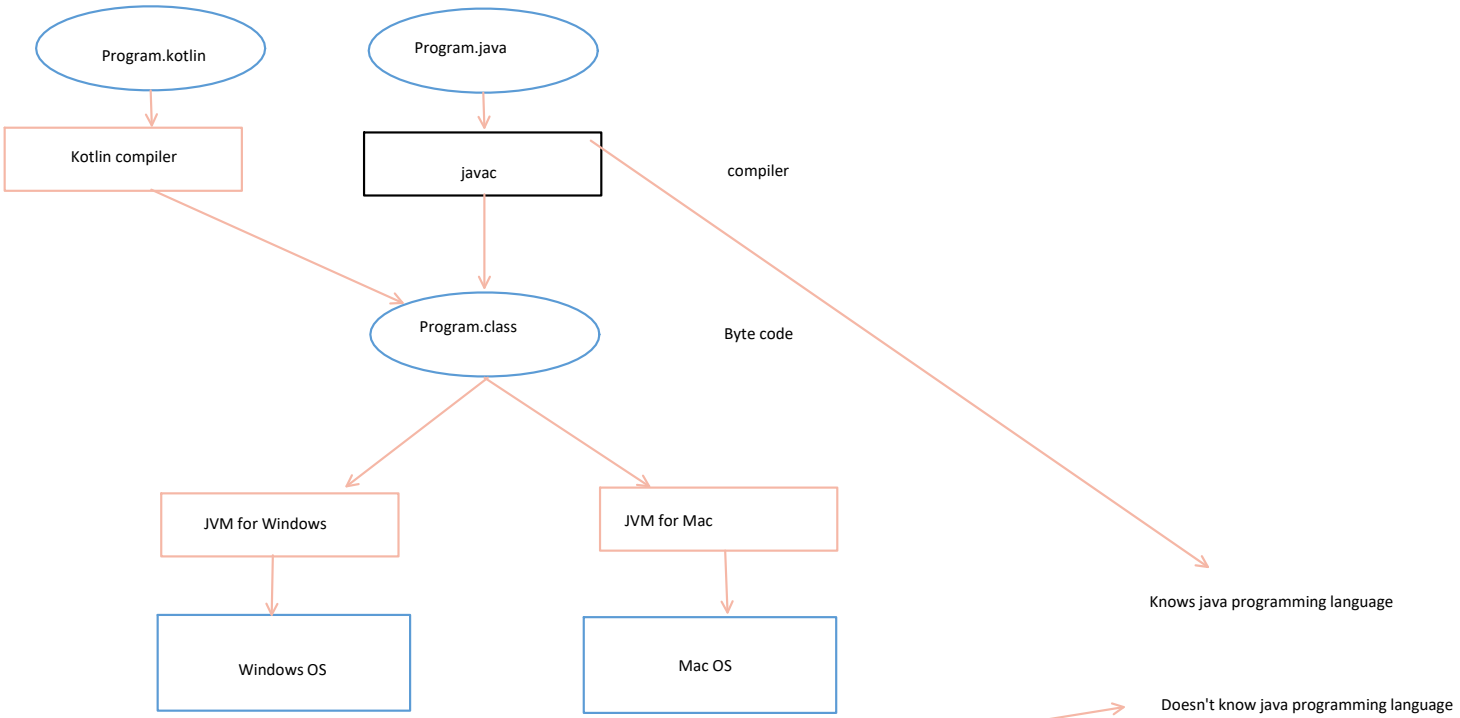
Java Architecture

Monday, 25 May 2020 3:24 PM



How do you run a java program?

C:\>java Program

- You are running a Program.class, **not program.java**
- **Is Program.class a java code?**
  - **No. It's a byte code**
- **That means 'Java' is not running Java Program. Why then we call it java?**

Java Terminology

Monday, 25 May 2020 3:37 PM

Java —> Represents to different and related ideas.

1. It's a … programming language.
   a. But
      - **Android Phone is Not a Java Phone but we using java language**
      - **There were older phone that were java Phone still we never wrote a java code in the phone.**
2. It's an Execution Platform t Execute can Application
   a. Android Phones use Java Programming language to write the program
   b. **But** Andorid Phones don't use Java Execution Platform
      i. They are not considered Java Phones
   c. You may write an Android application in a new programming language called Kotlin and can run it on Java Platform

Java compiler. Knows java

C:\> javac Program.java

C:\>**java Program**

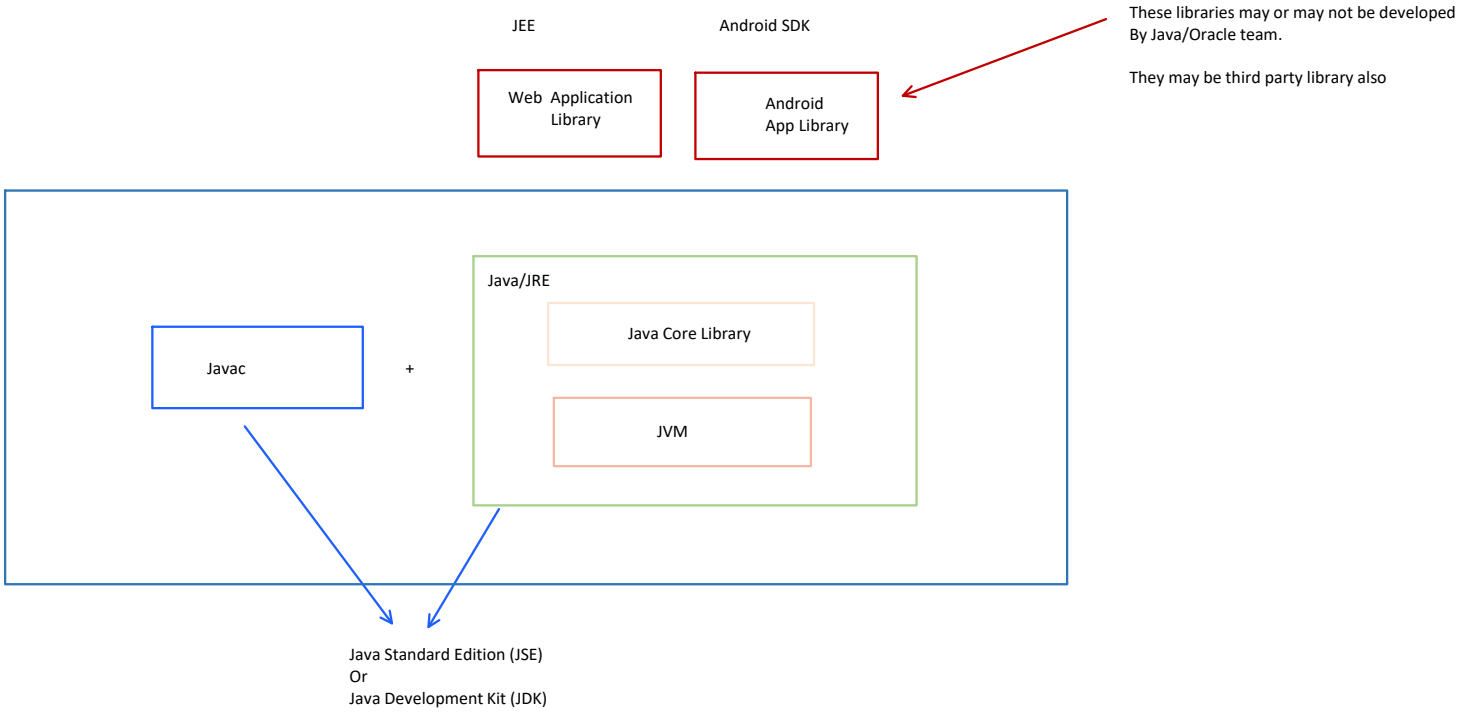
Executes the byte code that is not a java program

3. JRE — Just another word for 'Java Platform'
   a. When you want to run an application on an operating system you need Java installed -> you need JRE installed.
4. JVM — JVM is a component of Java Platform that interprets the entire byte code.

JEE

Java Technology Stack

Monday, 25 May 2020 4:01 PM



Abstract Windowing Toolkit

Monday, 25 May 2020 4:36 PM

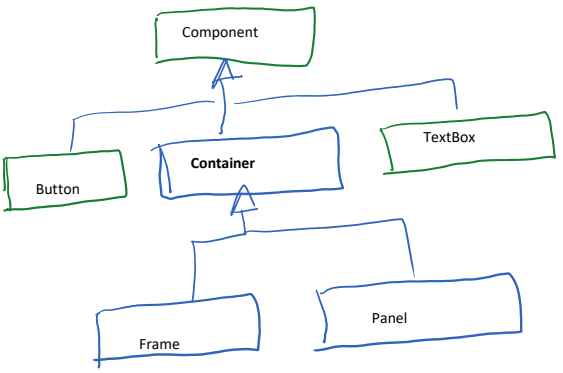
- Original Java Desktop (and Applet) Development Model
- You can develop Java Desktop Application using AWT
- These applications shall be platform Independent
  - On each platform JVM using native OS api to render the UI

AWT Model

AWT Components

- Each UI element such as **Button, TextBox, Checkbox** are individual Objects
- Each of these objects are subtypes of **Component class**

- They are gathered together on a **Frame**
- Frame is also a subclass of **Component**
  - The Exact hierarcy
    - Frame extends **Container** extends Component
- A **Container** is a component that can have childrens
  - Example
    - Frame
    - Panel
    - List
  - A Container can include another container



can be added to a container

outerlevel window
element with visible borders

Invisible container
that can have its
own children

Layout Manager

- AWT doesn't recommend (although allows) absolute positing or sizing of its components
  - The idea is when the UI size is changed the components should be proportionately changed in size and position to make it look good under new size
- This is managed by different Layout managers
- We can associate one (and only one) Layout Manager with a container
- The LayoutManager will decide how a component will be displayed within the container
- The LayoutManager may choose to ignore the dimension and location for individual components

LayoutManager

Monday, May 25, 2020 5:29 PM

- AWT comes with several Layout Managers

1. Flow Layout Manager

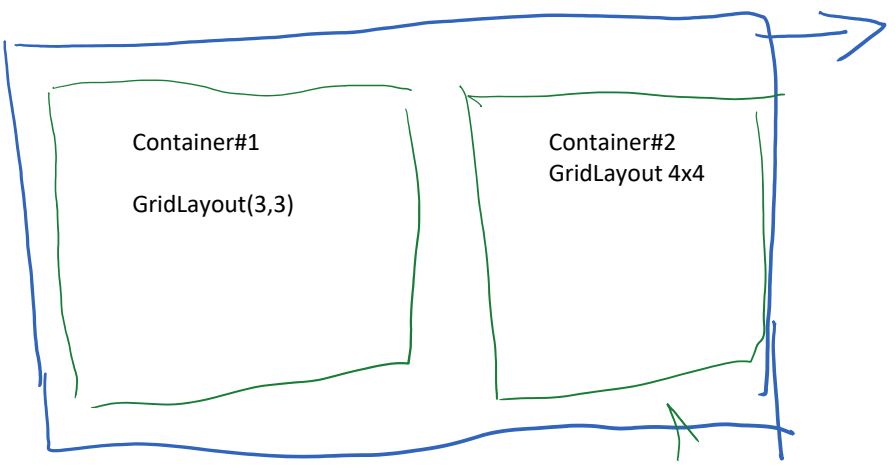
a. Simples of all the layout
b. Positions each component one after another left to right
c. When the line is full horizontally, moves next component to next row
d. The components keep chainging their position as container is resized
e. The Size of the component is not changed.

2. GridLayout

a. divides the container in equal size grids defined by rows and columns
b. Each component is first added row wise
    i. fill each column of row1 , then row 2 etc
c. Resize container will resize all components to maintain optimal space utilization and not breaking the grid formation
d. If the components are more than row X col,
    i. column is ignored,
    ii. number of row is maintained
    iii. number of column is recalculated
e. If number of controls are not a perfect multiple for row x col, some of columns of last row may remain empty.

Working with Multiple Layouts

- A container can have only one layout set at a time
- If you set again the new set call will replace the previous one
- To create complex UI we can add mulitple containers (Panel) over the frame
    - Each container can have its own layout separate from another



Event Handling

Monday, May 25, 2020 5:52 PM

Console Application vs GUI Application

Console Application	GUI Application
1. Program flow is sequential --- It is driven by developer. User defines flow from start to end	1. Program flow is **EventDriven.** Programmer sets the GUI and then what will happen depends on user action -- It is driven by user
2. Program starts with main() and ends with main()	2. Program starts with main where UI is created. End of main is just the beginning of actual program whose lifecycle is decided by user actions
Example **• you want to create a simple calculator** **1. you ask user number1** **2. you ask user number2** **3. you ask user operator (+,-,*,/)** **4. you print result** **5. you ask user if they want a second calculation** **6. end the program if they down want else loop to step 1** **• Problem** **○ You can't change the order of input -- number1, number2, operator** **○ if you want same number1 and number 2 for different operations, you need to enter again and again** **○ If you want to quit after entering number1, you cant unless you enter the other details and reach the prompt on step 5.**	**Example** **• you want to create a simple calculator** **1. Create the UI with two text box and 4 buttons** **2. respond to button clicks** **3. Main ends after intial setup** **• User may choose which number he would enter first and second** **• They can apply multiple operations on the same number** **• They may close application whenever they like** **• There is no nagging prompt if we want ot continue**
*	

Event Handling

- Most important aspect of any GUI application
- How the UI reacts when you interact with it

Import Event Handlng Elements

Event

- When you interact with the GUI, it generates an event
- Interaction may mean
  - clicking a button
  - typing in a text box
  - checking/unchecking a checkbox
  - moving your mouse
  - typing from keyboard
- Each of the interaction generates an Event
- In Awt **Event** is an Object that contains details related to what happened
  - Example
    - TextEvent
      - occurs when you type text in a textbox
      - The event must know these details
        - which text box you typed in (**source of event)**
        - what is typed
    - MouseEvent
      - Occurs when you interact with your mouse
        - MouseMove
        - MouseClick
        - The event must tell you
          - source of the event (component)
          - location where mouse has moved
          - button that is clicked.
  - Each event at least includes the **source**
- **To handle a particular Event** we have Event Listener
- Imortant events
  1. ActionEvent
  2. TextEvent
  3. MouseEvent

Event types

- Events may be **low leve or high level**

Low level event

- something happened with the hardware or divice the purpose is not clear
- Example
  - MouseEvent
  - KeyEvent

High level Event

- Event that gives meaning to action that just happened.
- different low level event may have some common goal
- same low level event may have different goal

4. KeyEvent
5. ComponentEvent
6. ItemEvent
7. WindowEvent

Event Listner

- for every Event **XEvent,** there is a listener interface called **Xlistner**
  - Example
    - for handling **ItemEvent** we have **ItemListener**
    - for handling **ActionEvent** we have **ActionListner**

Component

- A component may produce different events
- All components supports **MouseEvent and ComponentEvent**
- A listbox, checkbox etc also supports **ItemEvent (**when a item is selected or deselected)
- A TextBox also supports **TextEvent** and **ActionEvent**

Handling an Event

- To handle the event of a component we need to take following action
  1. Decide which event you are going to handle say **Xevent**
  2. Create a class that implments **Xlistner**
  3. Create an object of your class
  4. Select the component for which you are handling the event say **comp**
  5. Add the event by calling **addXListner()** method

Some Important Events

Action Event

- defines that user want to perform some action
  - caused by
    - ButtonClick
    - Hitting Enter in TextBox
    - Double Clicking a ListBox Item
- Tex

Text Event

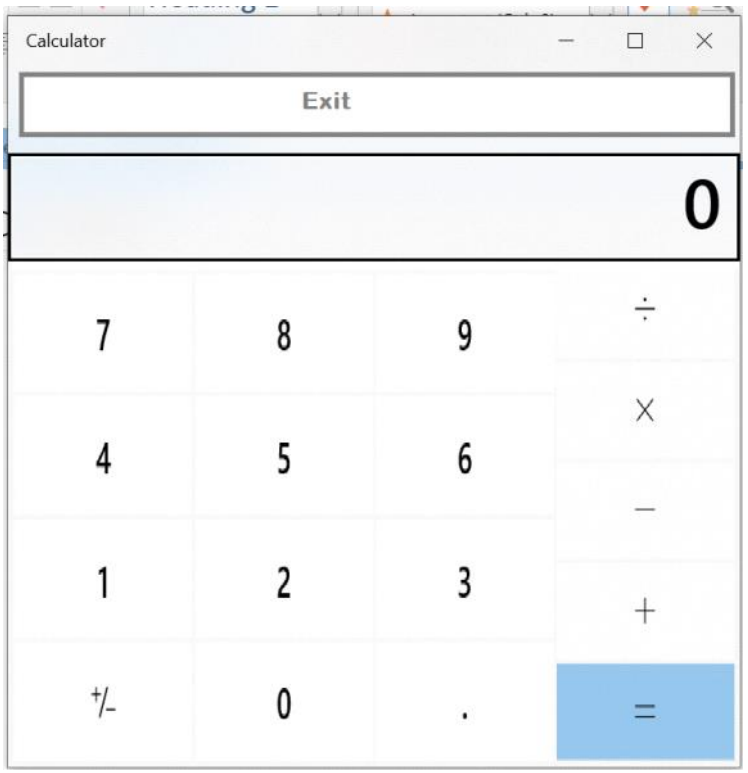
- when text values changes in a **TextField**
  - normal keyboard input causes TextEvent
  - enter key causes ActionEvent

Window Event

- Handles events related to the **Frame**
- The associated interface is **WindowListener**
- WindowListener has 7 methods
  - windowOpened() <— window opens for the first time
  - windowClosed() <— winow has been closed
  - windowActivated <— window gains focus
  - windowDeactivated <— window looses focus
  - windowIconfied<— window minimized
  - windowDeiconfied <— window restored
  - **windowClosing** <— somebody wants to close window (Close Button is clicked or Alt+F4 typed)
    - Window has not closed yet
    - This is a good place to ask if we need to close the window

Assignment -- Create a simple Calculator

Monday, May 25, 2020 7:09 PM

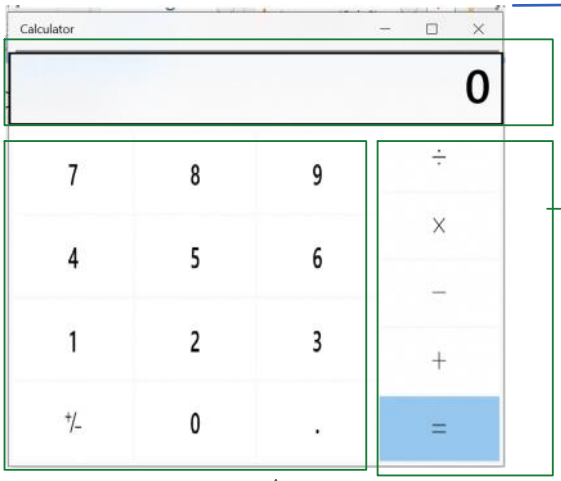


Write the logic to make the calculator work

- Create layout as close as you can get
- Define the event handlers to handle the button clicks
-

Assignment Solution

Tuesday, May 26, 2020 4:20 PM



calculator frame uses Border Layout

Screen to the **North** BorderLayout

A Panel added to the **East** of main Frame
This Pannel has 5 Buttons

What layout should use?

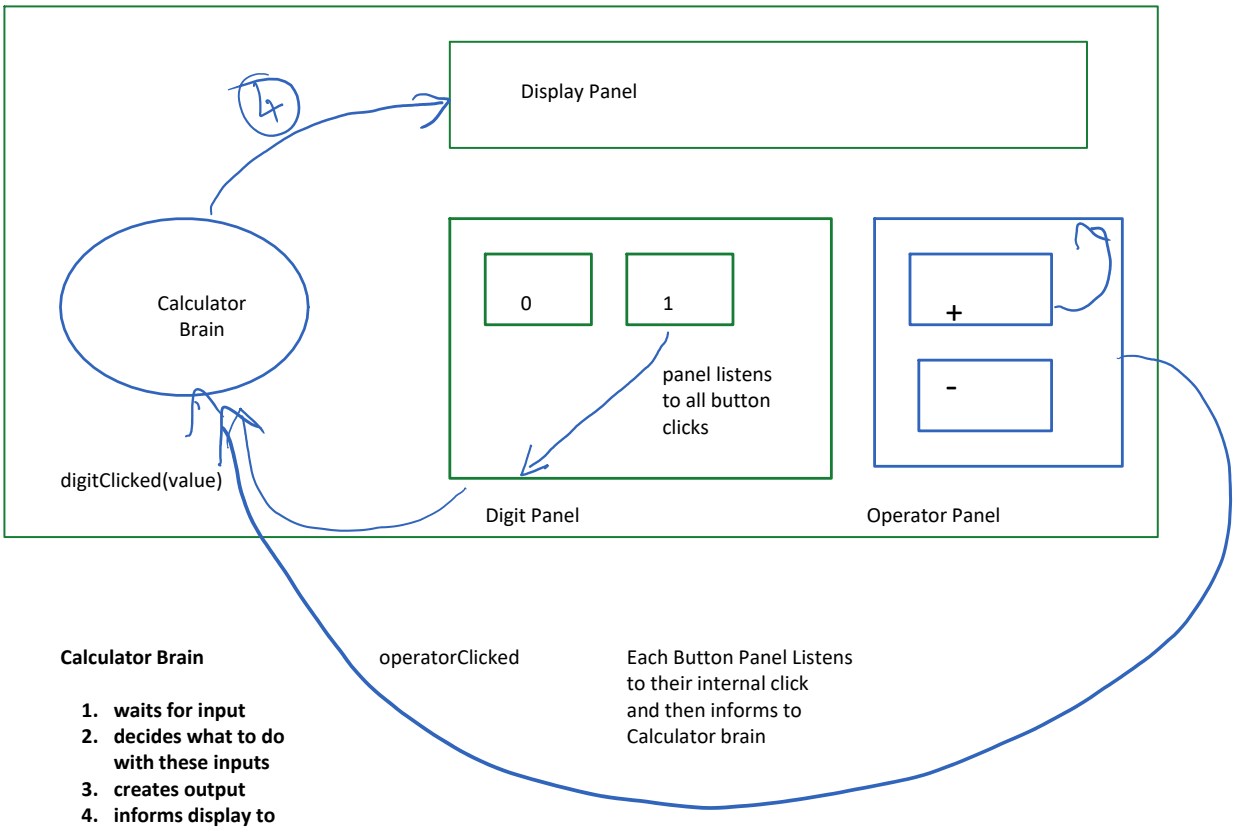
Another Panel with 12 Buttons
The Hight of 4 Buttons in this panel is equal to the Hight 5 buttons in another Panel

Where would you place this panel?

Program Flow

Tuesday, May 26, 2020 6:32 PM

CalculatorWindow

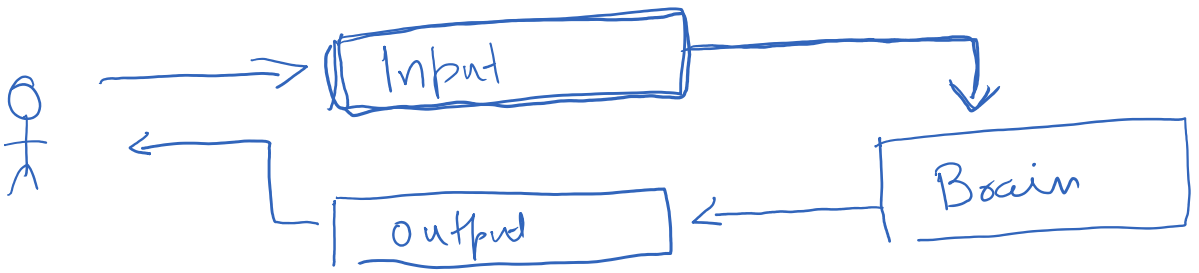


Calculator Brain

1. **waits for input**
2. **decides what to do with these inputs**
3. **creates output**
4. **informs display to display the value**

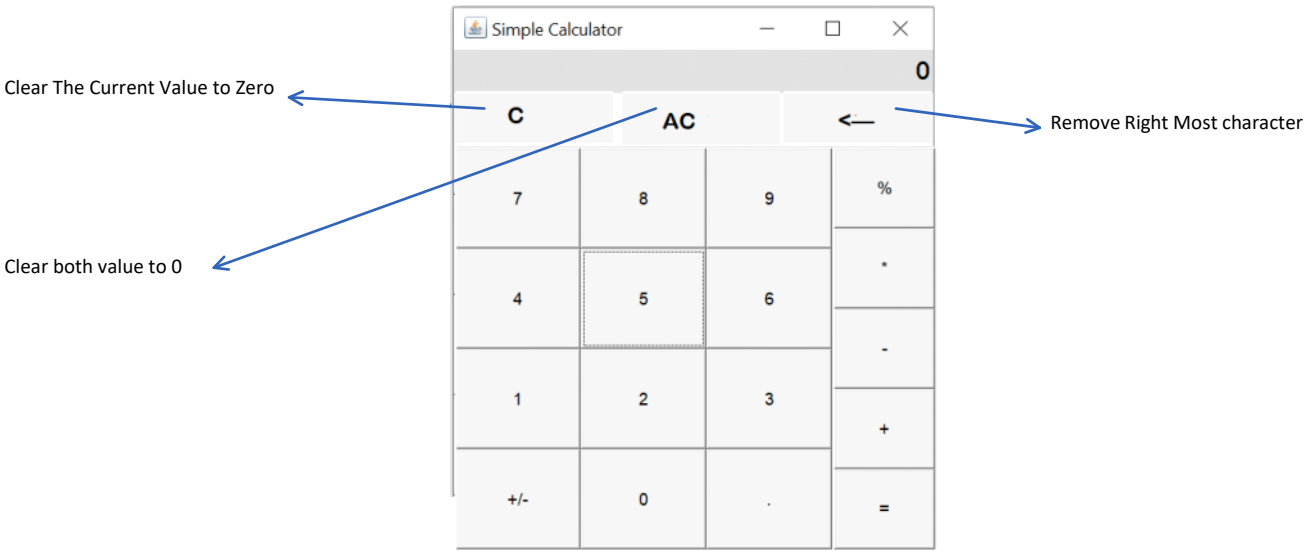
operatorClicked

Each Button Panel Listens to their internal click and then informs to Calculator brain



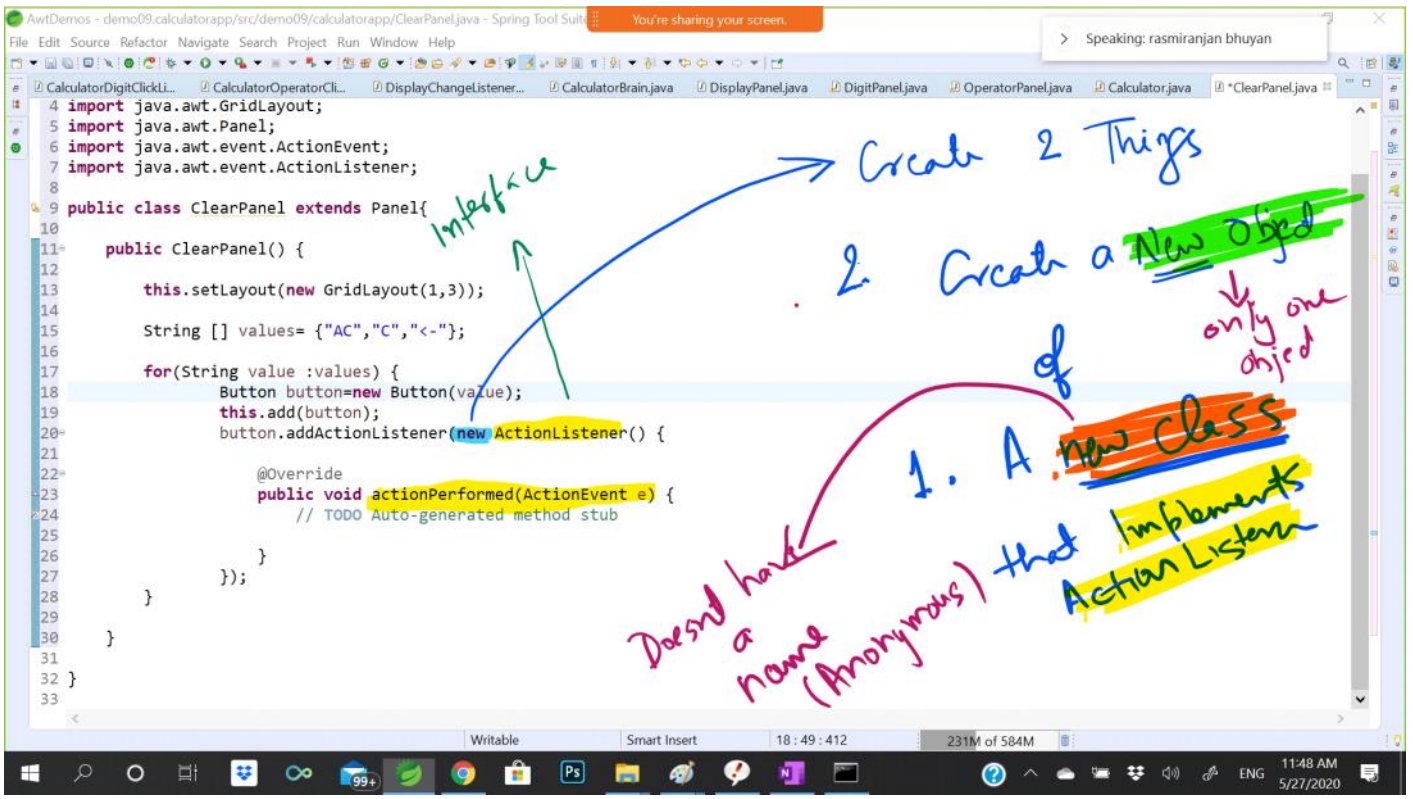
Assignment — New keys

Wednesday, May 27, 2020 10:20 AM



Anonymous Class

Wednesday, May 27, 2020 11:58 AM




```
4  import java.awt.GridLayout;
5  import java.awt.Panel;
6  import java.awt.event.ActionEvent;
7  import java.awt.event.ActionListener;
8
9  public class ClearPanel extends Panel{
10
11     public ClearPanel() {
12
13         this.setLayout(new GridLayout(1,3));
14
15         String [] values= {"AC","C","<-"};
16
17         for(String value :values) {
18                 Button button=new Button(value);
19                 this.add(button);
20                 button.addActionListener(new ActionListener() {
21
22                     @Override
23                     public void actionPerformed(ActionEvent e) {
24                         // TODO Auto-generated method stub
25
26                     }
27                 });
28         }
29
30     }
31
32 }
33
```

Evolution of Java Desktop

Wednesday, May 27, 2020 12:56 PM

1. Java originally shipped with **java.awt** package for desktop development (and browser based applet development)
    a. AWT interacted with underlying UI system of the operating system
    b. Each Awt component is internally translated to a control based on the operating
        i. Its look and feel will vary from one os to another
    c. Because each control must have an equivalent in all operating system AWT supports a very small number of components
        i. You can't have components not defined by all operating system.
    d. AWT is slow in performance.

2. **Java Swing**
    a. Another Java UI Framework
    b. Many concepts are similar in the two
        i. Layoutout
        ii. Event Handling
    c. Java swing doesn't show the native OS components
    d. **It draws all the components**
        i. Think each button and textbox is actually a nice drawing on the screen
    e. Swing is light weight, better performance.
    f. Swing can create more UI components which are not present in all OS
        i. It can even create component not present anywhere
    g. The UI can look the same on all OS

Sun or Oracle the original owner of Java didn't evolve the desktop model beyond their original attempt (almost 2 decades ago)

The core team focused on web and distributed development rather than desktop development

Third party organization developed their own library to create GUI applications.

SWT (Standard Window Toolkit)

Wednesday, May 27, 2020 1:15 PM

- It is a desktop application development library from **Eclipse**
- It is not from **Core Java Team.**
- Eclipse uses this library to develop the **Eclipse IDE**
- Eclipse has released it SWT library in public domain so that you can use this library to develop your own desktop application.
- **You need to download SWT library to use in your project.**
- You can develop desktop application with SWT
- Just like AWT, SWT also maps its UI to native OS components
- But for some advanced components SWT also draws them like SWING making them avaialble to all operating systems
- There is a huge number of components available in SWT (much more than AWT/SWING)

SWT vs AWT

- Both are desktop development models
- Both have several common concepts
    - Button, TextBox etc
    - Layout
    - Event Handling
    - Both tries to create
- SWT and AWT differs in following ways
    - Both have different components
        - AWT Button and SWT Button is not same
        - AWT Layout and SWT Layout is not same
        - They have different Design Models
    - SWT may be a bit more complex for begineers

Important SWT Elements

Wednesday, May 27, 2020 1:23 PM

1. Widget a.k.a Control

- SWT calls its GUI elements (Button, Label etc) as **widgets**.
- All **widgets** are sub class or **Control** (Awt uses **Component** as the super class)
  - we can use the term **widget** and **Control** interchangabley
    - This is simlar to **java.awt.Component**

2. Shell

- **Shell** is equivalent to **Frame** of AWT
- It represents the MainWindow

3. Display

- Display is an invible element which acts a controller for all Controls and widgets
- It helps you
  - handle events
  - draw on the widgets
  - handle color font etc

Bare minimum SWT Application requires

1. Shell
2. Display
3. **Event Loop**

Event Loop (a.k.a Message Loop)

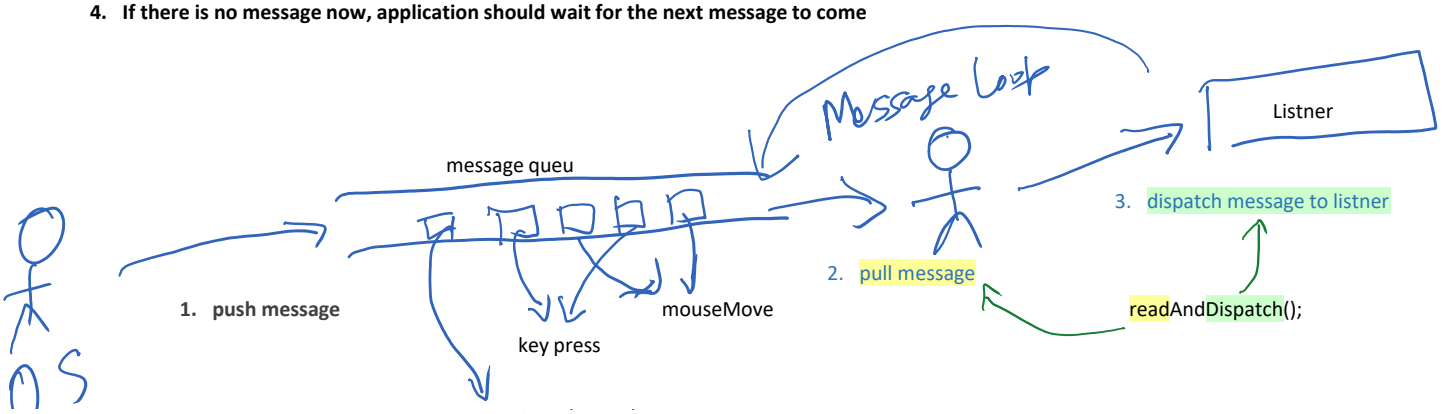
How GUI applications handle events (All GUI Application in any OS and language)

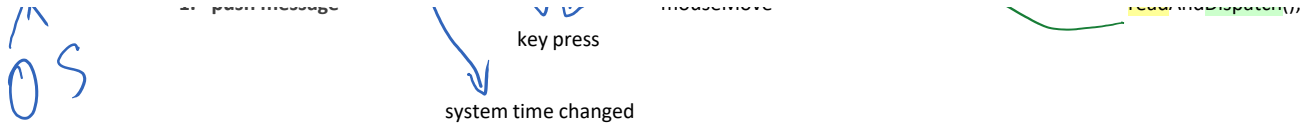
- Everytime User interacts with UI (mouse, keyboard, touch etc) this event is intercepted by the operating system
  - Remember you may
    - Click outside the current application
    - Use special key combos such as **Win, Win+R, Alt+Tab, Win+L**
      - These keys are handled in a different way by the OS and not your app

- The operating system looks at the event and the context (What key is typed, where it is typed etc) and decides following points
  1. Which application it is associated with. (based on key or location)
  2. What does user want **intent (also known as High Level Events)**
     i. Is the current Mouse Click
        1) A Button Click
        2) Check box selection
        3) Attempt to close window
  3. OS then sends a **message** to the application. Message tells what has happened
  4. Now your application should do whatever it needs to do.

Message Queue

- If user moves your mouse very quickly it may generate 100s of message in a second.
- OS can keep sending all the message
- While OS is sending a new message your application may still be handling the previous one.
- So every application **must maintain a message loop.**
  1. **Every application will have a message Queue**
  2. **OS will add the new message to the Queue**
  3. **The application should have loop to read the message and process it4.** 4. loop back to get next message
  4. **If there is no message now, application should wait for the next message to come**





AWT vs SWT

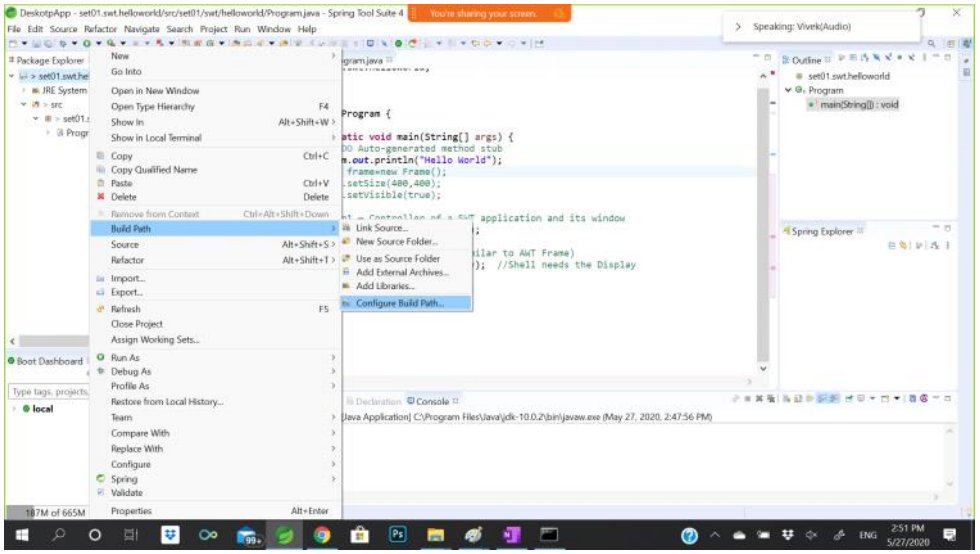
- AWT has an internal message handling loop
- **SWT expects programmer to create the message loop. Its not out of box**
- **Since there we don't wait for message to come application quits immediately**

How long should message loop run?

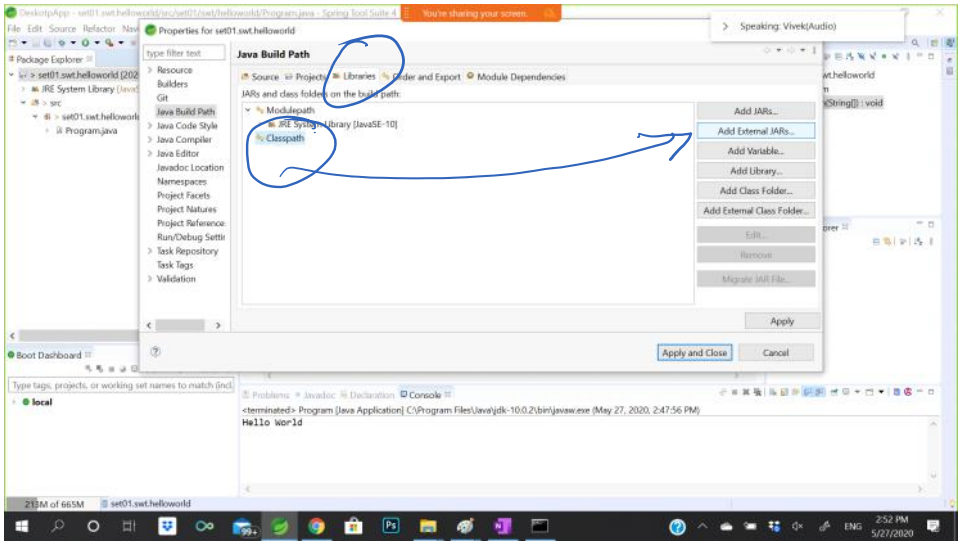
- Till my window is not closed.
- Once the window is closed, no message can arrive for you as OS will not now what should be sent you!

Include the SWT jar file

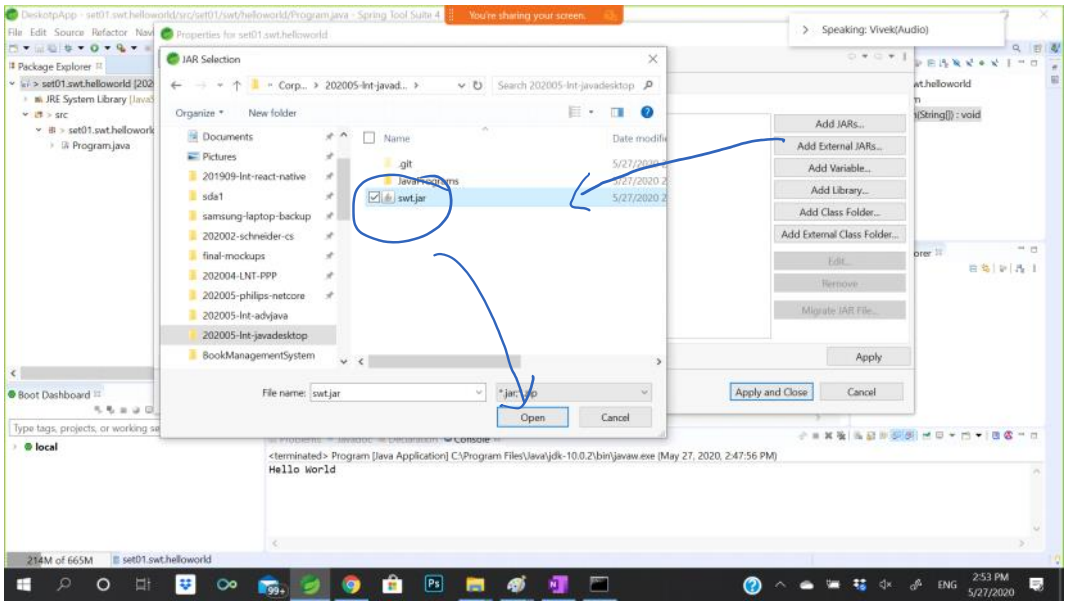
Wednesday, May 27, 2020 2:51 PM



1. Right click project in project exploerer
2. BuildPath —> Configure BuildPath

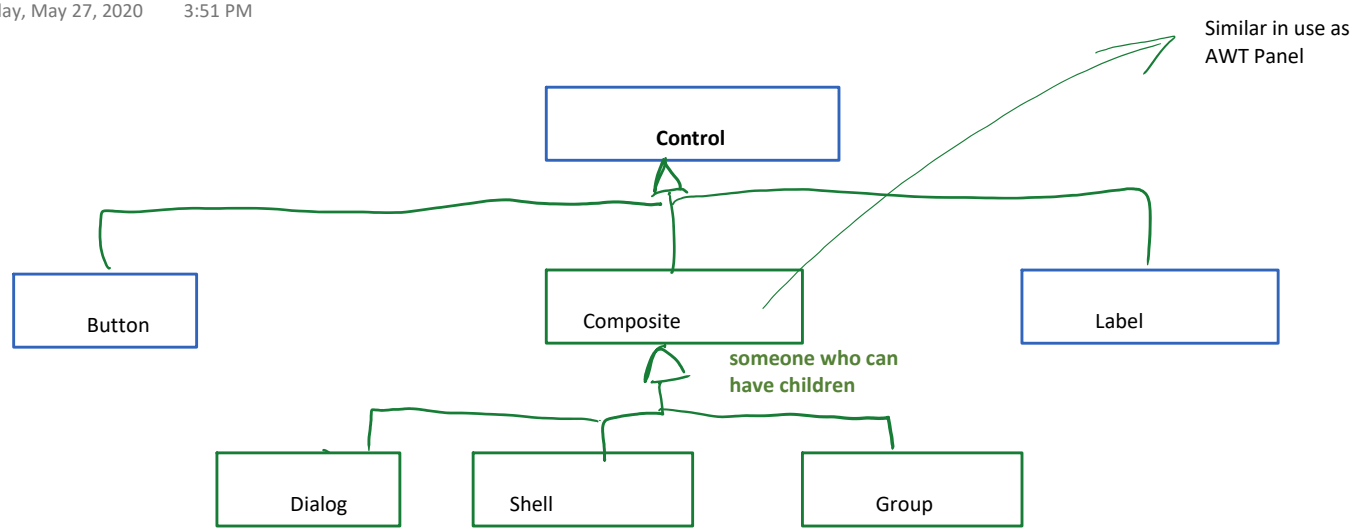


Library —> Classpath —> Add External Jar



SWT Controls

Wednesday, May 27, 2020 3:51 PM



Similar to AWT Frame

How to you Add Control to a Container

Action	AWT	SWT
create main window	**Frame frame=new Frame();**	Display display=new Display(); **Shell shell=new Shell(display);**
create a container withing frame	Panel panel=new Panel(); **frame.add(panel); //this line connects parent-child**	Composite panel=new Composite(**shell**); //you pass the parent to add new component to parent
Add a Button to container	Button button=new Button(); panel.add(button);	Button button=new Button(panel);

Programming Practice

Wednesday, May 27, 2020 4:31 PM

- As a modern Object Oriented Practice, Inhertiance should be avoided
- Keeping in line, unlike AWT Frame, SWT Shell can't be unherited.
- Shell is not an abstractclass, But throws runtime exception, while trying to inherit
- You application should have a class that can build the UI over the shell

SWT Layoutout

Wednesday, May 27, 2020 4:41 PM

- Like AWT, SWT supports Layouts
- The layout is slightly different between SWT and AWT in their design and working
- Important SWT Layout
    - RowLayout <— similar to the Flow Layout of AWT
    - GridLayout <— Similar to GridBagLayout of AWT
    - FormLayout <— n/a in AWT
    - n/a in SWT <— BorderLayout of AWT

Assignment Project

Wednesday, May 27, 2020 5:27 PM

Task Management

- You can add a Task (Todo)
- You can define its details
- Task can have states completed/inprogres/notstarted
- We can save and load task
- We can search our task
- We can delete or edit our task details

Task Info

- Task Heading
- Task Details
- Task Completed
- Date Created
- Expected Completion Date
- Actual Completion Date
- Task Notes

UI Requirement

1. Add New Task
2. Task List and Navigator
3. Task Details
4. Add Notes to the Task
5. Mark Task complete

SWT GridLayout

Thursday, May 28, 2020 11:12 AM

- SWT GridLayout divides the composite into Grids or Row and Columns (Like AWT)
- However we have a greater degree of flexiblity in GridLayout of SWT than AWT
- It has high customization Options to add components
- Each Component
    - Can choose
        - Their desired size (may be ingored)
        - if they want to be stretched
        - or stay in optimal size
    - May take one or more adjust horizontal or vertical area
    - May grab the remaining Area horizontally or vertically

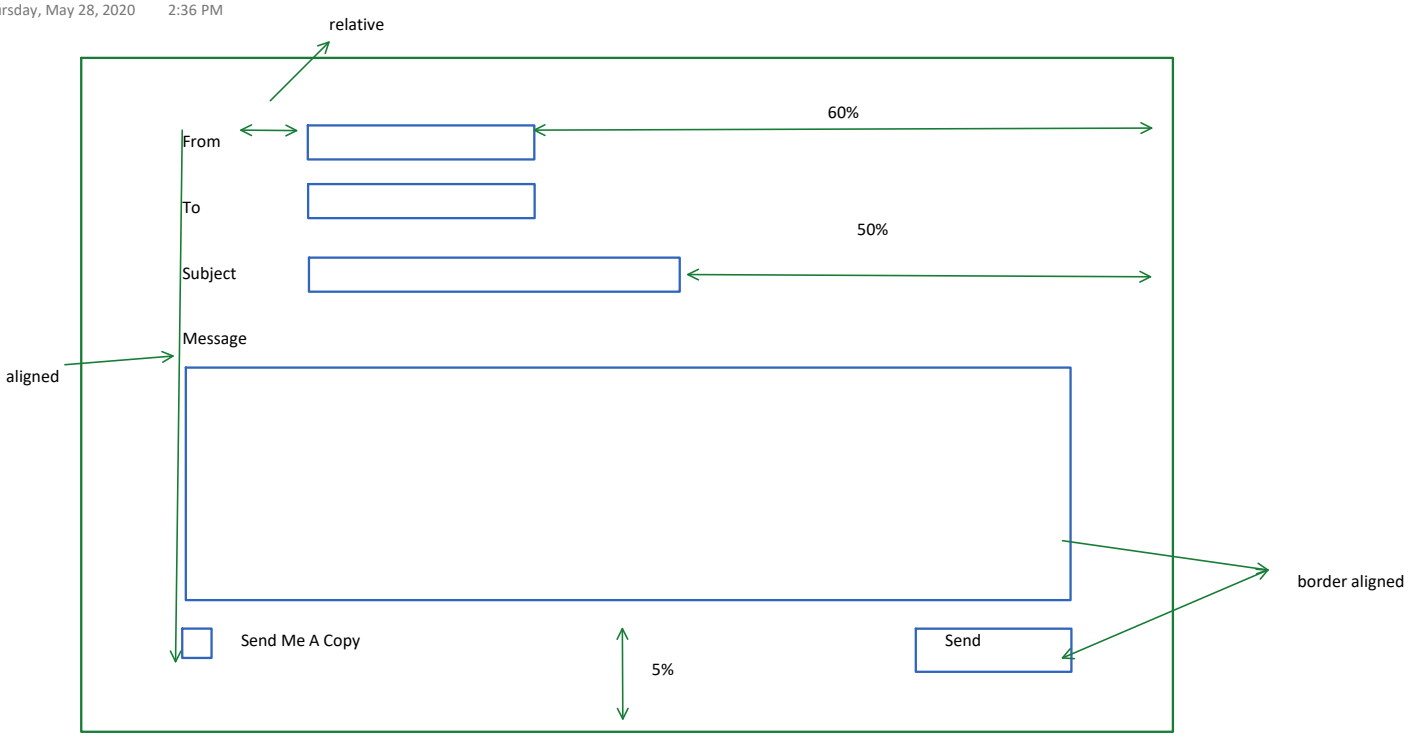
Assignment — SWT Calculator

Thursday, May 28, 2020 12:39 PM

Create the given UI design

FormLayout

Thursday, May 28, 2020 2:36 PM



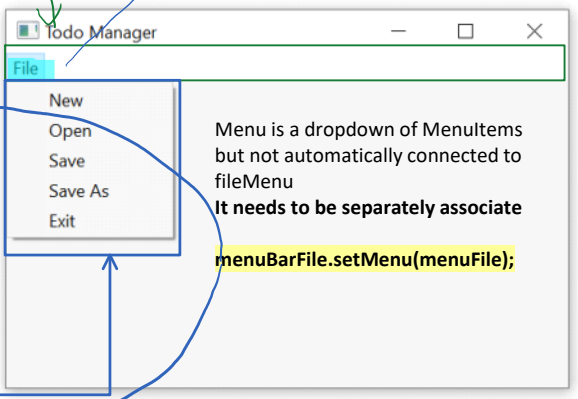
Menu

Friday, May 29, 2020 11:09 AM

2. This line only adds File section on the bar. Currently there is no dropdown menu here

```
private void buildMenu(Composite parent) {
    // TODO Auto-generated method stub
    Shell shell=(Shell) parent;

    //STEP 1. menuBar can be connected only to a Shell
    //  It creates the large bar
    Menu menuBar=new Menu(shell,SWT.BAR);


    //STEP 2. create MenuItem to display on the MenuBar.
    //This item MUST be MenuItem with CASCADE Design
    MenuItem menuBarFile= new MenuItem(menuBar, SWT.CASCADE);
    menuBarFile.setText("&File");

    //STEP 3. Menu need to have dropdown list of Menus
    Menu menuFile=new Menu(shell, SWT.DROP_DOWN);
    MenuItem menuFileNew=new MenuItem(menuFile,SWT.PUSH);
    menuFileNew.setText("&New");
    MenuItem menuFileOpen=new MenuItem(menuFile,SWT.PUSH);
    menuFileOpen.setText("&Open");
    MenuItem menuFileSave=new MenuItem(menuFile,SWT.PUSH);
    menuFileSave.setText("&Save");
    MenuItem menuFileSaveAs=new MenuItem(menuFile,SWT.PUSH);
    menuFileSaveAs.setText("Save &As");
    MenuItem menuFileExit=new MenuItem(menuFile,SWT.PUSH);
    menuFileExit.setText("E&xit");
    menuFileExit.addListener(SWT.Selection, new Listener() {

        @Override
        public void handleEvent(Event arg0) {
            // TODO Auto-generated method stub
            Application.instance.getShell().dispose();
        }
    });
```

1 Bar can be created only with a Shell or Decorated Component



Menu is a dropdown of MenuItems but not automatically connected to fileMenu

It needs to be separately associate

menuBarFile.setMenu(menuFile);

1

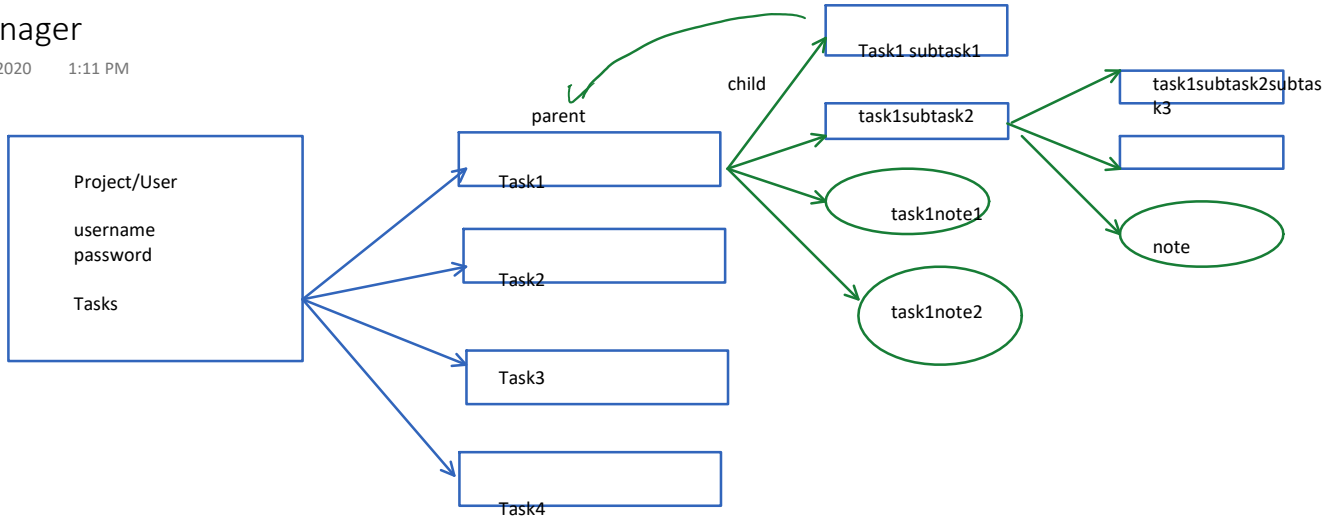
MenuBar should also be explicitly associated with the Shell.

shell.setMenuBar(menuBar);

A bar can be associated only with Shell and not with other controls

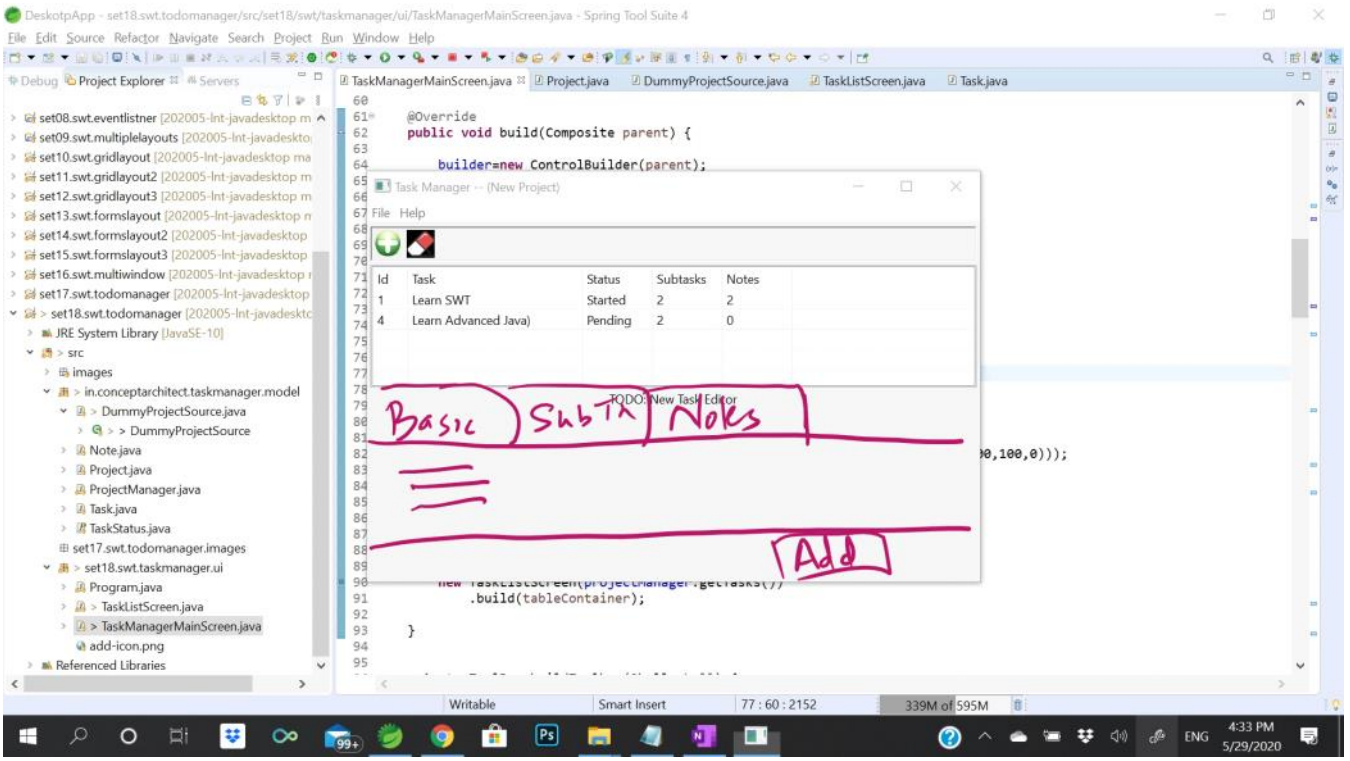
Task Manager

Friday, May 29, 2020 1:11 PM



Assignment -- Task Manager

Friday, May 29, 2020 4:35 PM



JFace

Wednesday, May 27, 2020 12:57 PM

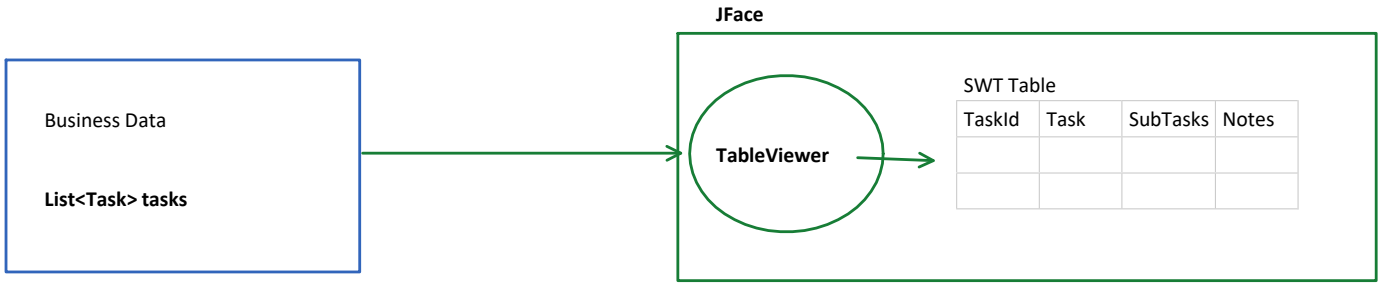
- Extension of SWT
- Based on SWT Library
- Adds Additional Components and Features to SWT Library
- JFace is not about adding more controls to SWT. It is about making SWT better and easier to use
- JFace Provides same basic helpers
    - **MessageDialog —>** Similar to our MessageDialog in design and purpose.
        - Wrapper around MessageBox api
        - Message Box API requires multiple lines of redundant code
            - Create Box with a shell and style
            - decide title and message
            - decide icon and button set
            - open the dialog
            - **This is handled by a single line of code**
    - **GridDataFactory** --> Similar to our Grid class
        - Creates GridData with required element using chained functions()

Most Important Elements of JFace

1. Resource Manager
    a. Makes working with Fonts, Colors, Images easier.
2. Viewer
    a. Viewers are the decorators (wrappers) around SWT components to make it work easier with user data
    b. Separates User Data and SWT Component
    c. Reduces montonous code and effort

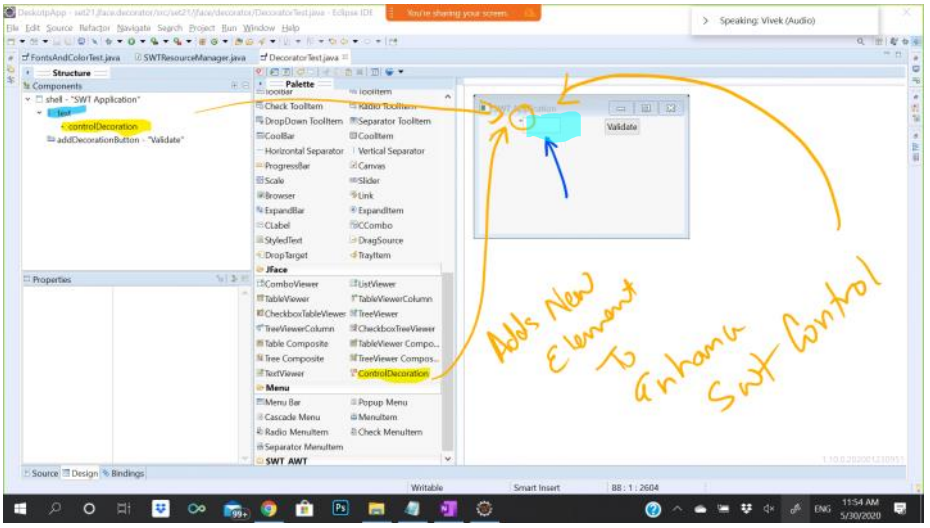
2.1 swt.Table and jface.TableViewer

- Working with SWT Table Involves
    - Creating Individual Columns
    - Adding Data
    - Editing Data
    - Refreshing the table on every change
- There are more complex elements such as Tree or TreeTable
    - Working with plain swt may proof to be pain swt.
- JFace acts as a mediator between Our Business Data and SWT data controls (Table,Tree etc)
    - It automatically generates rows column and nodes based on our needs.



JFace Decorator

Saturday, May 30, 2020 11:56 AM



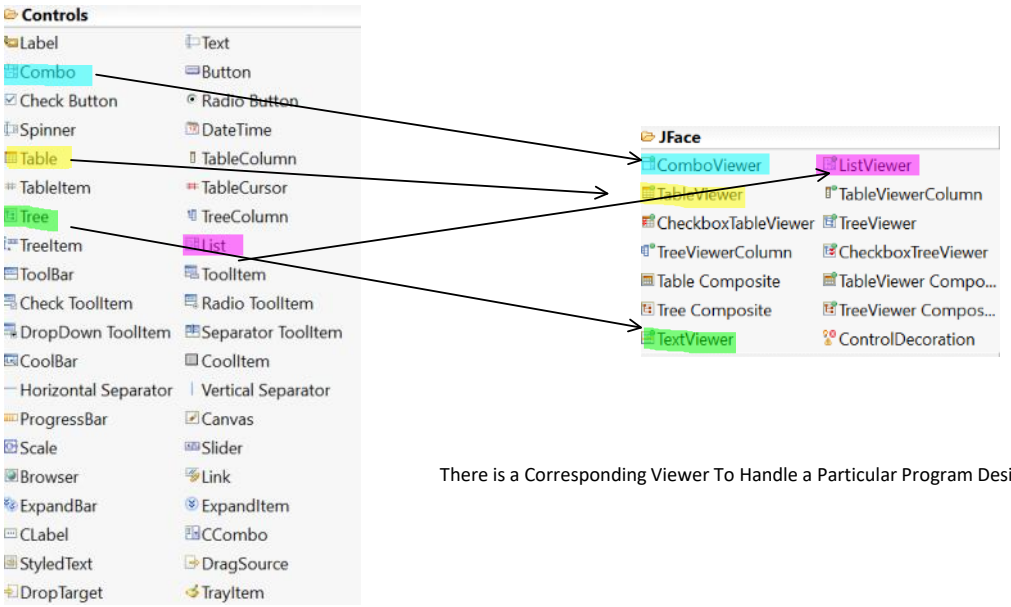
- A Decorator associates itself with an existing SWT Component.
- It can add new functionality associating it with existing controls
- Think conceptually JFace control will be a wrapper around SWT Control
- ***WindowsBuilder actually represents the idea in a inverse way***
    - ***It shows decorator inside control***

JFace Viewer

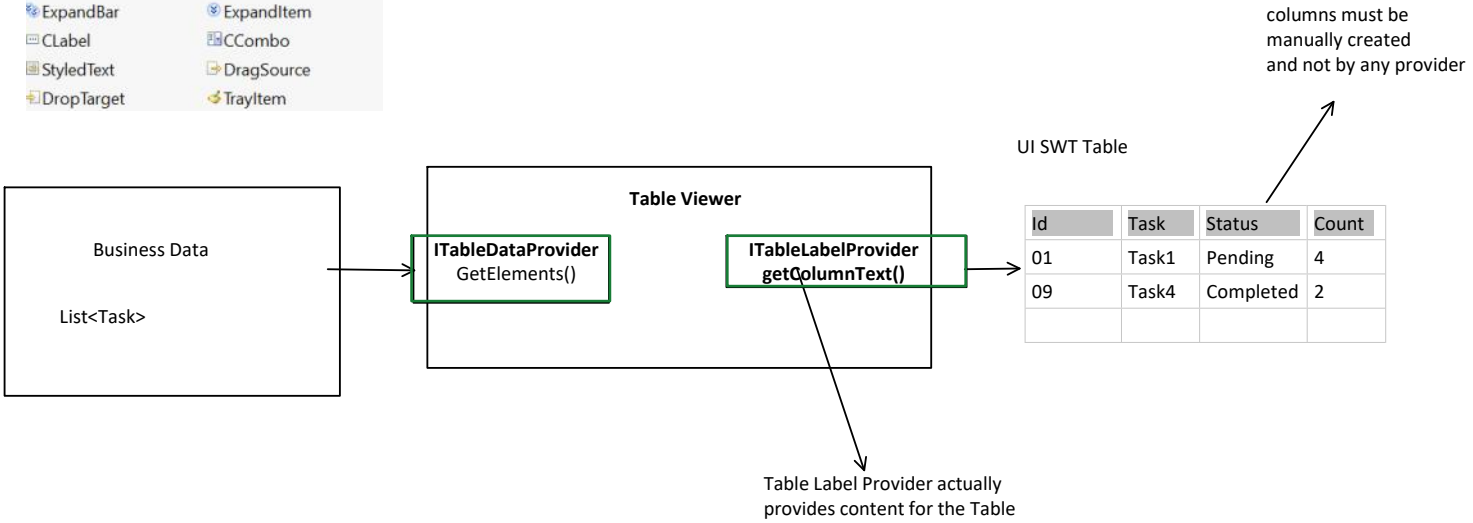
Saturday, May 30, 2020 11:59 AM

- A Viewer is like a Decorator
- It generally decorates a more complex data oriented control such as
  - Table
  - Combobox
  - ListBox
  - Tree
- It provides a structured way to access data from the Model (Business Objects) and automatically generates the UI element for the underlying controls
  - Values for List or Combo
  - Rows, Columns and cells for Table
  - Nodes for Tree
- For Each different Control Type there is a special Viewer Object

SWT Controls



There is a Corresponding Viewer To Handle a Particular Program Design!



Framework vs Library

Friday, May 29, 2020 4:53 PM

What is the difference between Framework and Library

Library

- It is a set of reusable component (methods, objects, classes etc)
- Think of library as a set of tools in your toolbox (e.g. hammer, screwdriver, cutters, drill machine)
    - Each tool has a specific purpose (generally very small) purpose.
    - You can use one or more of the tools to create any good design or mechine
- These tools tell you what is their purpse They don't tell you how to use them

Framework

- Is often a library+
- Framework is a larger combination of tools to deliver a specific purpose
- Framework not only provides you a set of tools, it also provides you best practices guidelines as to how to use them
- Eg. A Car Assmebly system
    - It defines where you need to place all the components of a car
    - Assembly system, when provided correct parts in correct order can automatically assemble the car, test int make the production fast
    - It may provide conveyer belts where each belt will be for specific parts
        - You need to know the important fragments and where to place them
    - Once you know framework will take care of your design
- You don't use framework the way you like, but you use it the way Framework likes

Example from our project

- We have developed a few library components
    - Grid (Simple helper to generate Grid data)
    - MessageDialog (can show various message boxes)
- We have created a **Application** class is **Framework**
    - It defines
        - There will be
            - One Main Shell
            - The Shell can be accessed everywhere using **Application.instance.getShell()**
            - You can configure services which can be accessed everywhere
    - If you use Application class
        - You must use it the way it expects you to do

        - You must define your services with Application
        - You must define main shell with the Application
        - You must implmenet UIBuilder to define the main Shell UI design
    - Benefit of Use
        - Easy to access the service
        - A more systemetic design
        - Auto implementation of message loop
        - Easy to get messagebox, colors, fonts etc

Framework Component

- Generally a Framework will have
    - Core Startup logic (you may not write your main())
    - A set of interfaces to define your **plugins** to the component
    - A structured way for components to interact with each other
    - Few teriminology that are Framework specific and can be used to break your large design in smaller parts
    - A huge set of reusable libarary
        - This is the bribe for using the framework

Eclipse RCP

Friday, May 29, 2020 5:11 PM

Eclipse Rich Client Platform

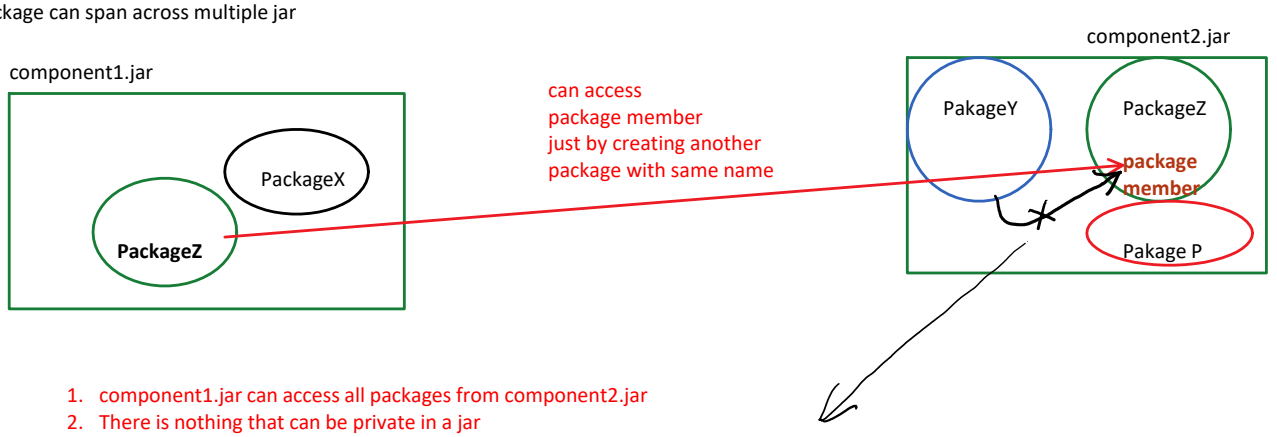
- This is a **Framework for Creating Rich Client Applications**
- Helps us develop Desktop Applications (Browser Application) using Component Model and Best Practices from OSGi Spcification
- It uses SWT to build its basic UI
- But it provides you a design model so that you can create your application following an architecture similar to that used by Eclipse itself
- Think About Eclipse (IDE) Components
  - Perspective
    - A collections of related windows to make your work easy
  - Window
    - One window with its specific data
  - Extensible Architecture
    - You can add new features to Eclipse Application
    - You can add dynamically new menu items
    - You can influence the exiting editor by adding colors, autocomplete etc

Javas Non Modular Design (Prior to Java 9)

Saturday, May 30, 2020 3:32 PM

Java Non Modular Design Issue

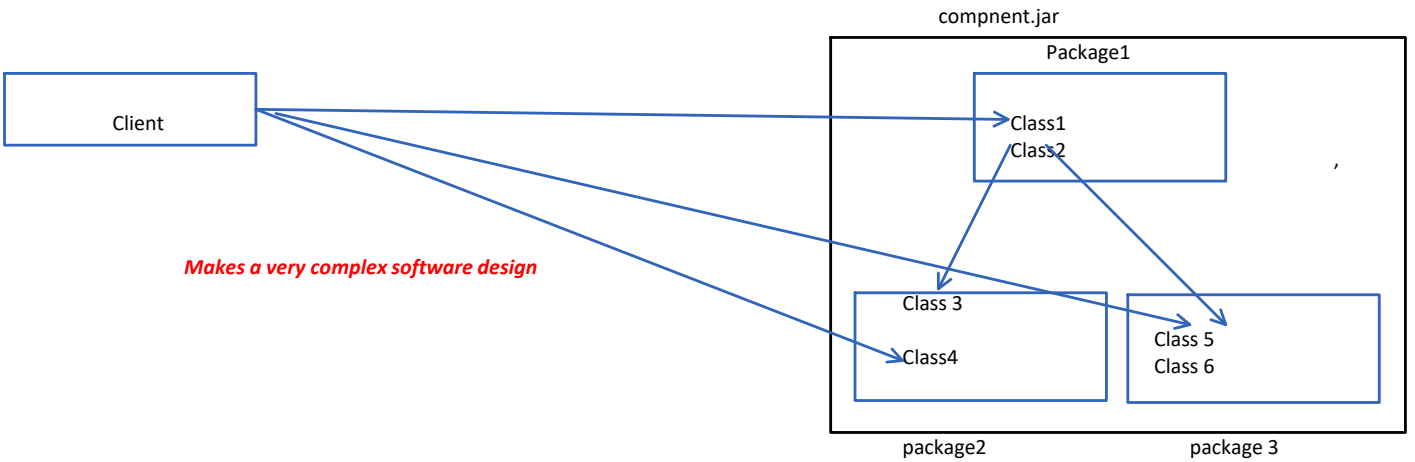
- Java package are conceptual entity
- Java Jar is a physical entity
- No vendor boundy
  - Nothing defines if something belongs to someone
- Class can be public or package
- Package classes can't be accessed outside jar
- But a package can span across multiple jar



1. component1.jar can access all packages from component2.jar
2. There is nothing that can be private in a jar
3. A package may have some package level class, not accessible from other package
4. **But can be accessed by a package with samename even in a different jar**
5. **There is no package boundry**

Components may have complex Dependency

- A client may need to use 10 classes from Component2
- When those classes change you change
- Component2 can't stop its public classes from getting accessed by outsiders



OSGi

Saturday, May 30, 2020 3:32 PM

Open Service Gateway Initiative

- Component Object Model for Java
- It is a specification which is implemented by several independnet vendors
- The initiative is not make sure that you have Module Design
- The Standard eveolved because Java community felt that the Java design is not sufficiently Modular
- The specification Predates Java 9 Module System
- Java 9 Module system solves a lot of the issues related OSGi
- OSGi allow you to
- OSGi takes a modular approach to Programming

OSGi Module Programming Approach

- OSGi defines a **Module** known as **Bundle** or **Plugin**
- A Bundle is specificall a Jar file with a specifal manifest
- The manifest defines which packages of the bundle should be accessible outside and which are not
  - **Bundles allows you to create private packages (and inturn classes)**
  - These classes/packages can't be accessed by anyone outside the Bundle
- Bundle can help you decide which elements of your design will be accessible from outsider
- Bundles are served using OSGi servers
  - Servers are application which will publish and allow the access to the components in structure and well defined way
  - These bundles can be dynamically configured on the server
  - They can start and stop providing the services
- Client and Service (component) talk to each other with the help of these bundles
- Different vendors (companies) have created their own OSGi implementation
  - Example
    - Apache --> Filix
    - Eclipse --> Equinox