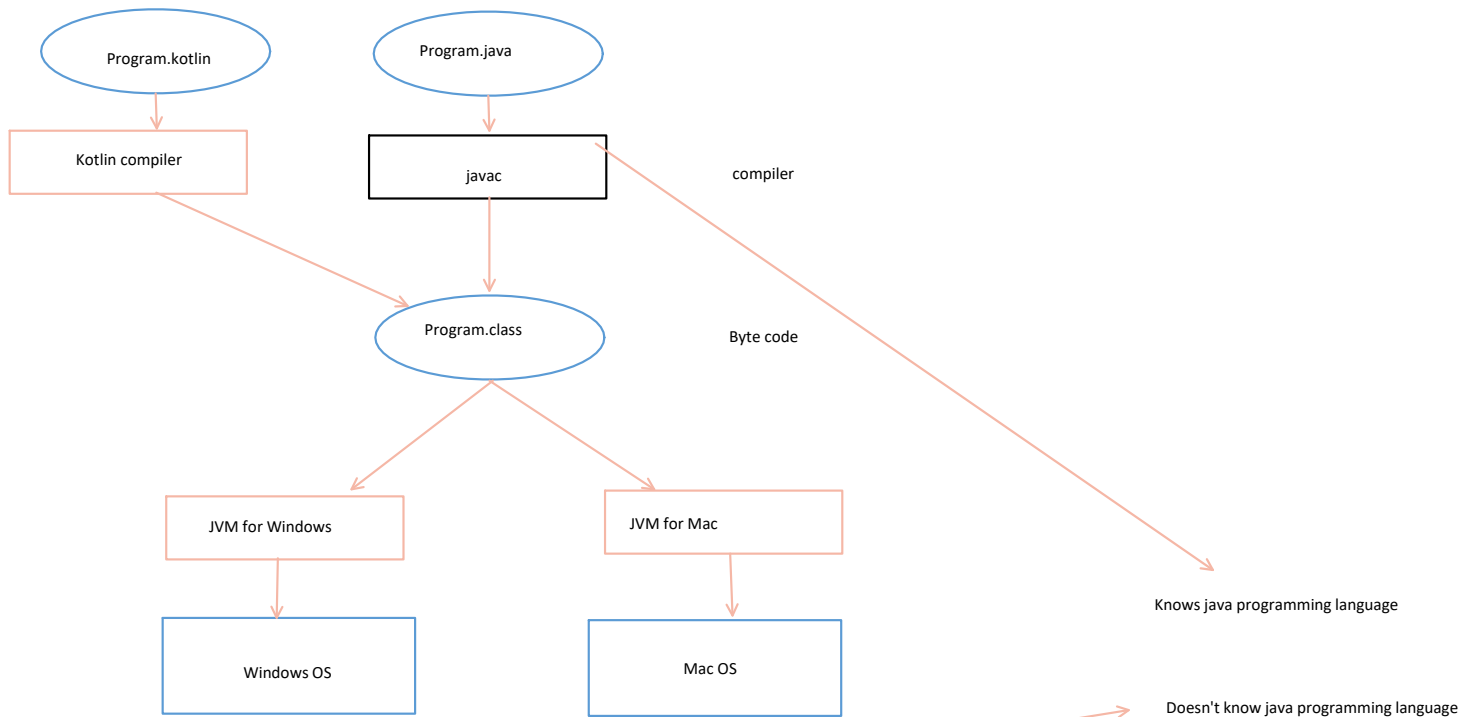


# Java Architecture

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 compiler. Knows java

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.

C:\> **javac** Program.java

C:\> **java** Program

Executes the byte code that is not a java program

## 2. It's an Execution Platform to Execute an Application

- a. Android Phones use Java Programming language to write the program
- b. **But** Android 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

## 3. JRE — Just another word for 'Java Platform'

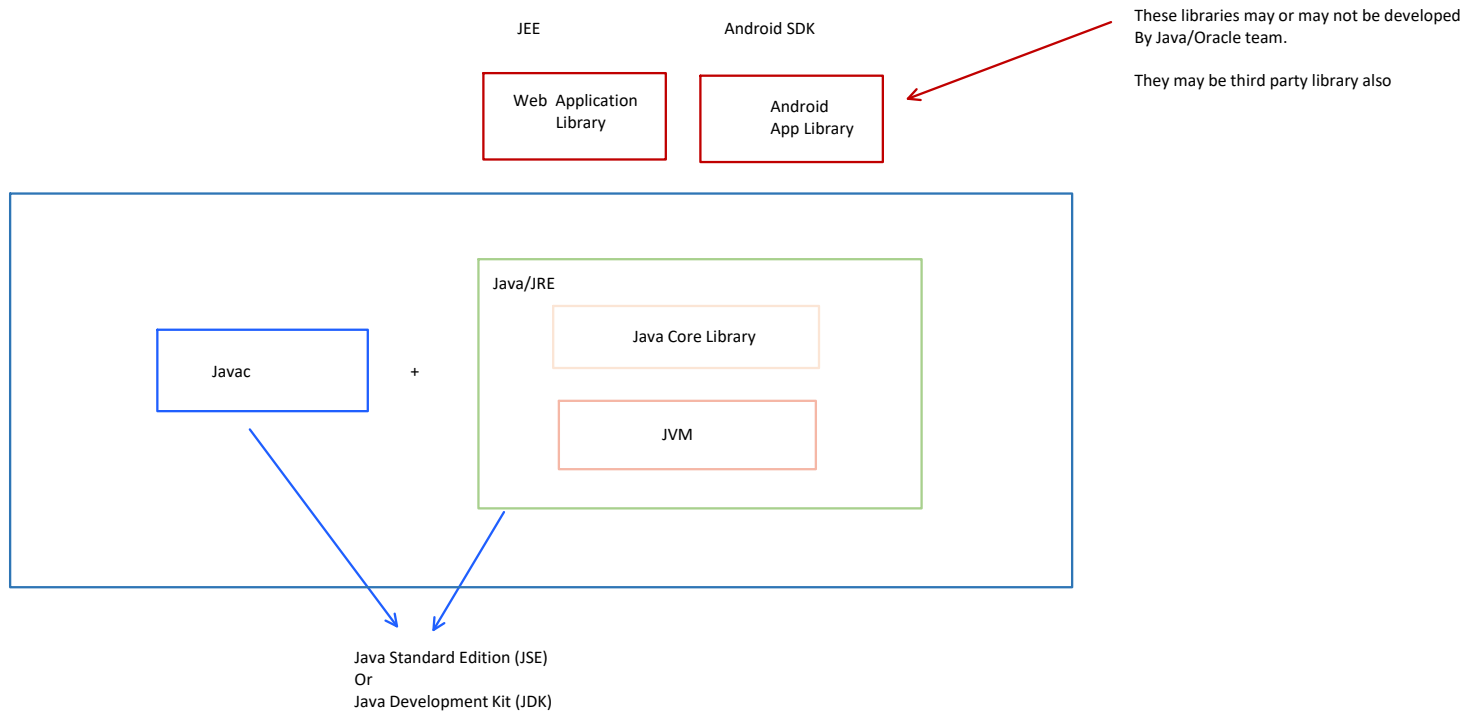
- a. When you want to run an application on an operating system you need Java installed -> you need JRE installed.

## 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

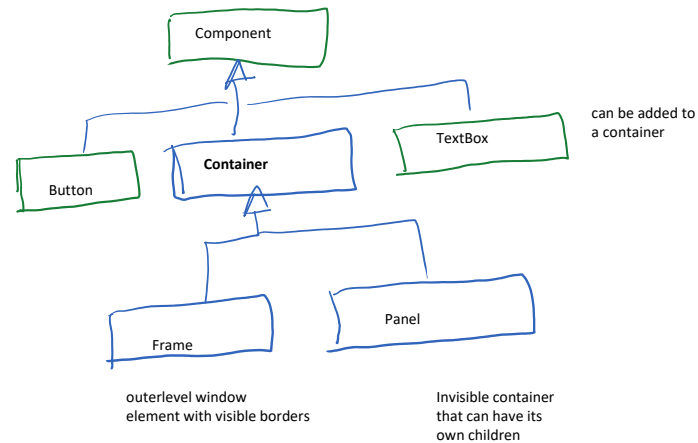
Monday, 25 May 2020 4:36 PM

- 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 **Container**
  - The Exact hierarchy
    - Frame extends **Container** extends Component
- A **Container** is a component that can have childrens
  - Example
    - Frame
    - Panel
    - List
  - A Container can include another container



### 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

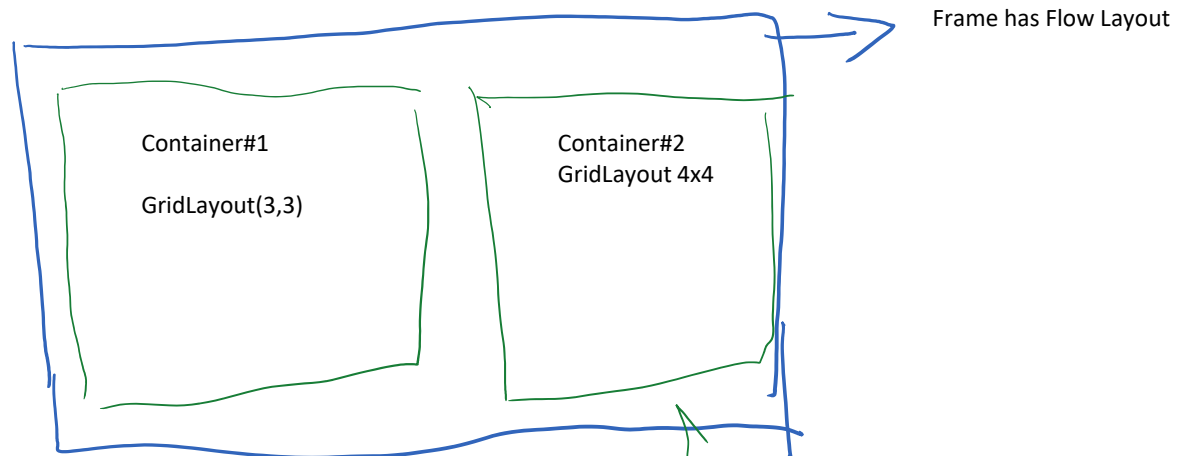
- a. Simplest of all the layouts
- b. Positions each component one after another left to right
- c. When the line is full horizontally, moves next component to next row
- d. The components keep changing their position as container is resized
- e. The size of the component is not changed.

## 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 row 1, 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 multiple containers (Panel) over the frame
  - a. 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 <b>EventDriven</b> . 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
<b>Example</b> <ul style="list-style-type: none"><li>• you want to create a simple calculator<ol style="list-style-type: none"><li>1. you ask user number1</li><li>2. you ask user number2</li><li>3. you ask user operator (+, -, *, /)</li><li>4. you print result</li><li>5. you ask user if they want a second calculation</li><li>6. end the program if they don't want else loop to step 1</li></ol></li><li>• Problem<ul style="list-style-type: none"><li>○ You can't change the order of input -- number1, number2, operator</li><li>○ if you want same number1 and number 2 for different operations, you need to enter again and again</li><li>○ If you want to quit after entering number1, you can't unless you enter the other details and reach the prompt on step 5.</li></ul></li></ul>	<b>Example</b> <ul style="list-style-type: none"><li>• you want to create a simple calculator<ol style="list-style-type: none"><li>1. Create the UI with two text box and 4 buttons</li><li>2. respond to button clicks</li><li>3. Main ends after initial setup</li></ol></li><li>• User may choose which number he would enter first and second</li><li>• They can apply multiple operations on the same number</li><li>• They may close application whenever they like</li><li>• There is no nagging prompt if we want to continue</li></ul>
*	

## Event Handling

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

## Import Event Handling 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
        - ◆ MouseEvent
        - ◆ MouseEvent
        - ◆ 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
  - Important events
    1. ActionEvent
    2. TextEvent
    3. MouseEvent

## Event types

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

### Low level event

- something happened with the hardware or device 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 Listener

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

## 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 implements **XListener**
  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 **addXListener()** 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

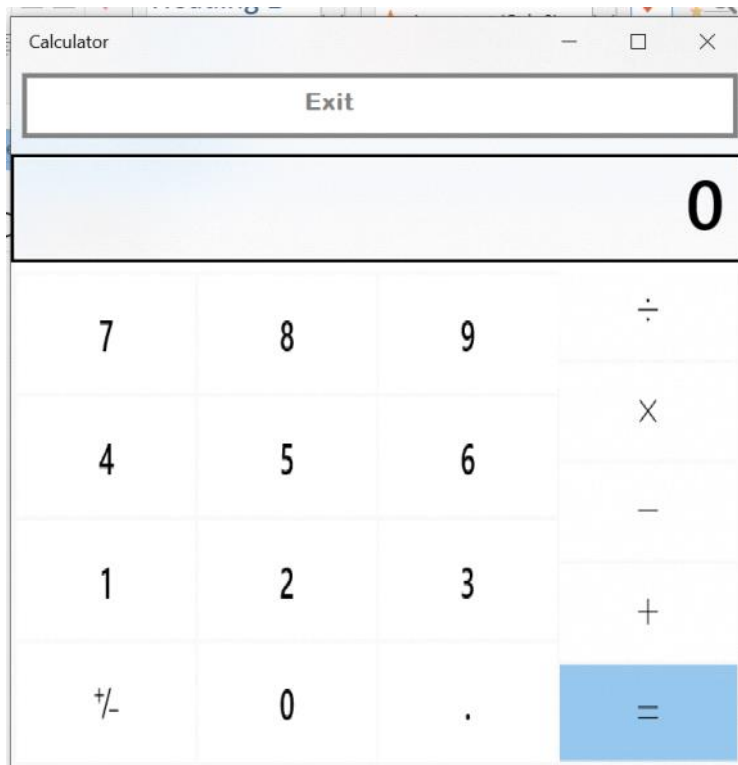
- when text values changes in a **TextField**
  - normal keyboard input causes TextEvent
  - enter key causes ActionEvent

### 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

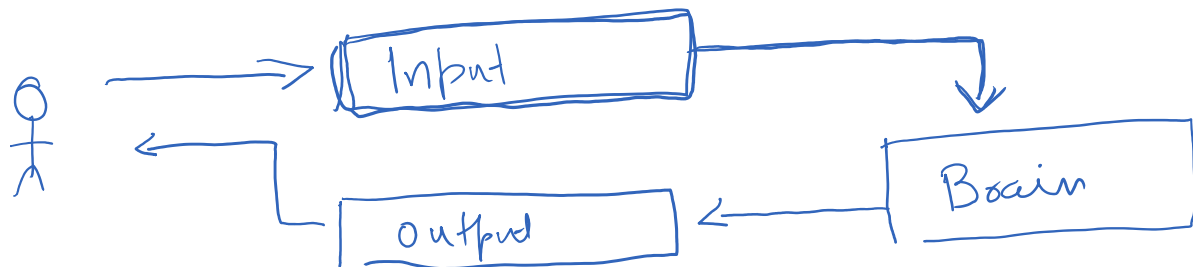
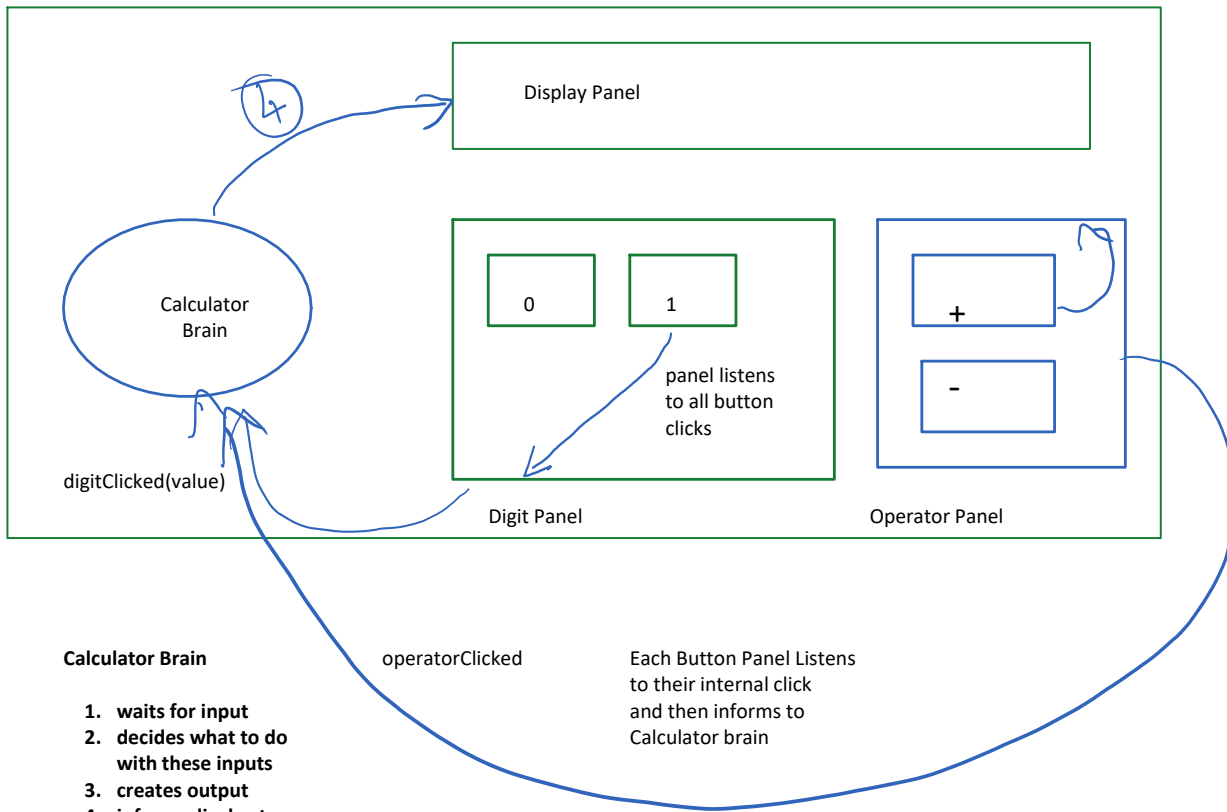
A Panel added to the **East** of main Frame  
This Pannel has 5 Buttons  
What layout should use?

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



## Assignment — New keys

Wednesday, May 27, 2020 10:20 AM

