Object - Oriented Programming 2

Week 6: Graphical User Interface -GUI (Introduction, Applet, Event Handling, AWT – Abstract Windowing Toolkit)

By Elubu Joseph - MSc.IS

Lecturer

Department of Information Technology

Kumi University

Email: josebulinda@gmail.com

jose@kumiuniversity.ac.ug

Agenda

1. Graphical User Interface -GUI Introduction,
2. Applet,
3. Event Handling,
4. AWT –Abstruct Windowing Toolkit,

Introduction to Graphical User Interface - GUI

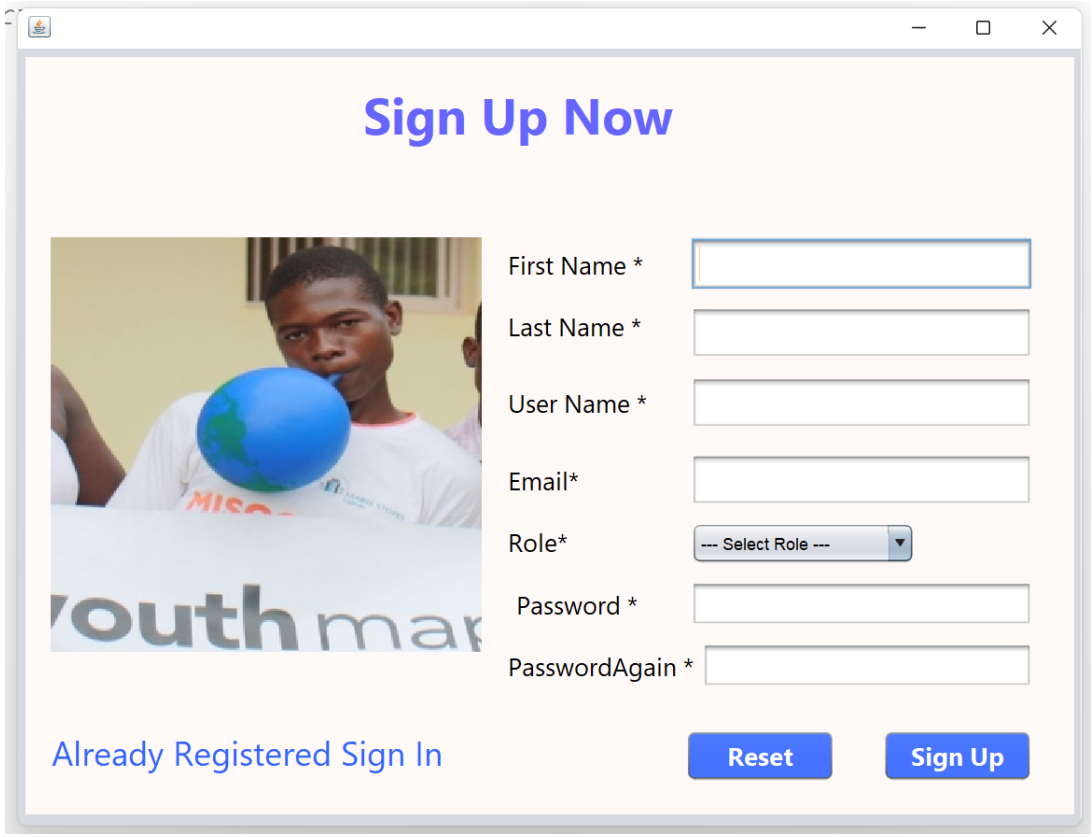
According to computerhope.com(2021), A **GUI** is a system of interactive visual components for computer software. A GUI displays objects that convey information, and represent actions that can be taken by the user. The objects change color, size, or visibility when the user interacts with them.

In the same vain, we can say GUI is a form of user interface that allows users to interact with electronic devices through graphical icons and audio indicator such as primary notation, instead of text-based UIs, typed command labels or text navigation.

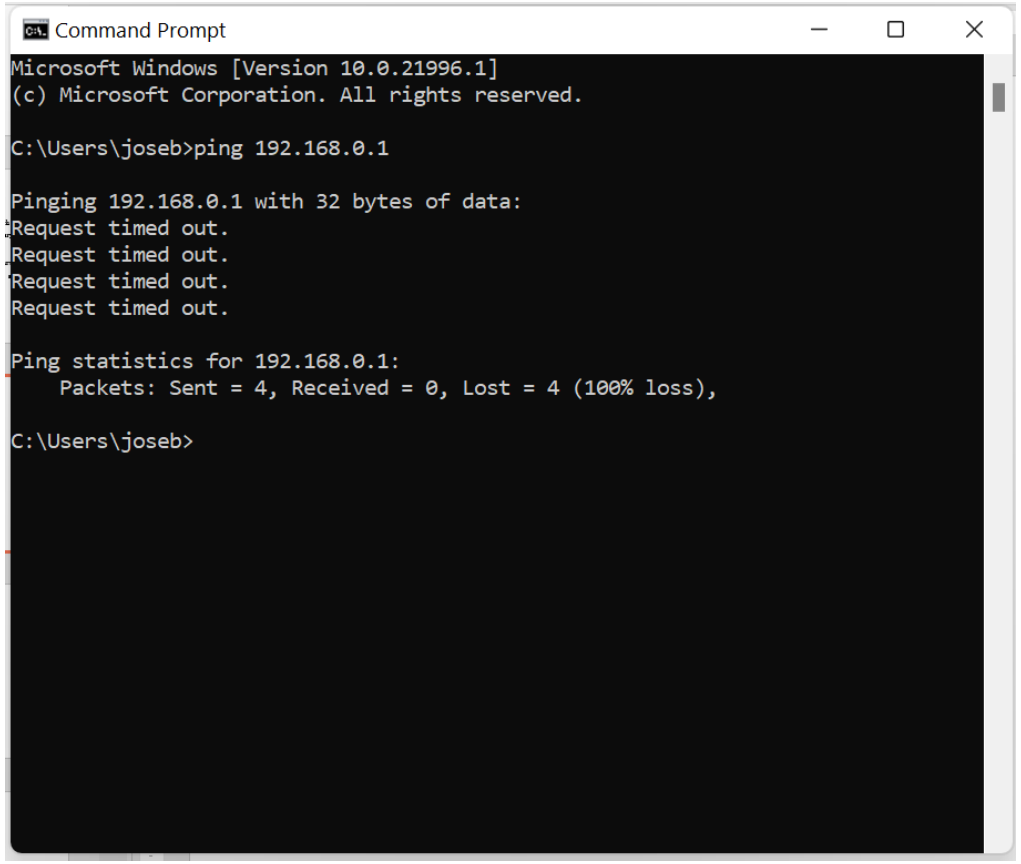
GUIs were introduced in reaction to the perceived steep learning curve of CLIs (command-line interfaces), which require commands to be typed on a computer keyboard, (Wikimedia Foundation. 2022).

GUI Vs CLI

GUI



CLI



Components of GUI

To make a GUI as user-friendly as possible, there are different elements and objects that the user use to interact with the software. Below is a list of each of these with a brief description.

1. **Button** - A graphical representation of a button that performs an action in a program when clicked.
2. **Dialog box** - A type of window that displays additional information, and asks a user for input.
3. **Icon** - Small graphical representation of a program, feature, or file.
4. **Menu** - List of commands or choices offered to the user through the menu bar.

Components of GUI

To make a GUI as user-friendly as possible, there are different elements and objects that the user use to interact with the software. Below is a list of each of these with a brief description.

5. **Menu bar** - Thin, horizontal bar containing the labels of menus.
6. **Ribbon** - Replacement for the file menu and toolbar that groups programs activities together.
7. **Tab** - Clickable area at the top of a window that shows another page or area.
8. **Toolbar** - Row of buttons, often near the top of an application window, that controls software functions.
9. **Window** - Rectangular section of the computer's display that shows the program currently being used.

(Computerhope.com)

Assignment

1. Find out the Advantages of GUI over CLI
2. What are examples of a GUI operating systems?
3. Find out examples of CLI.

Applets

Basics of Applets

An **applet** is a Java program that is embedded and runs in a web browser. An applet can be a fully functional Java application because it has the entire Java API at its disposal.

There are some important differences between an applet and a standalone Java application, including the following : -

1. An applet is a Java class that extends either java.applet.Applet class or javax.swing.JApplet class.
2. A main() method is not invoked on an applet, and an applet class will not define main().
3. Applets are designed to be embedded within an HTML page.

Applet Basics

4. When a user views an HTML page that contains an applet, the code for the applet is downloaded to the user's machine.
5. JVM is required to view an applet. The JVM can be either a plug-in of the Web browser or a separate runtime environment.
6. The JVM on the user's machine creates an instance of the applet class and invokes various methods during the applet's lifetime.
7. Applets have strict security rules that are enforced by the web browser. The security of an applet is often referred to as sandbox security, comparing the applet to a child playing in a sandbox with various rules that must be followed.
8. Other classes that the applet needs can be downloaded in a single Java Archive (JAR) file.

Life Cycle of an Applet

Five Methods in the Applet class that gives us the framework on which we build any serious applet : -

1. **init()** — This method is intended for whatever initialization is needed for your applet. It is called after the param tags inside the applet tag have been processed.
2. **start()** — This method is automatically called after the browser calls the init method. It is also called whenever the user returns to the page containing the applet after having gone off to other pages. also this method is started every time the browser is maximized.
3. **paint()** — Invoked immediately after the start() method, and also any time the applet needs to repaint itself in the browser. This method **take in the Graphics class object** as it parameter, which object is **used to access the Graphics class methods**. The paint() method is actually inherited from the java.awt.Components class.

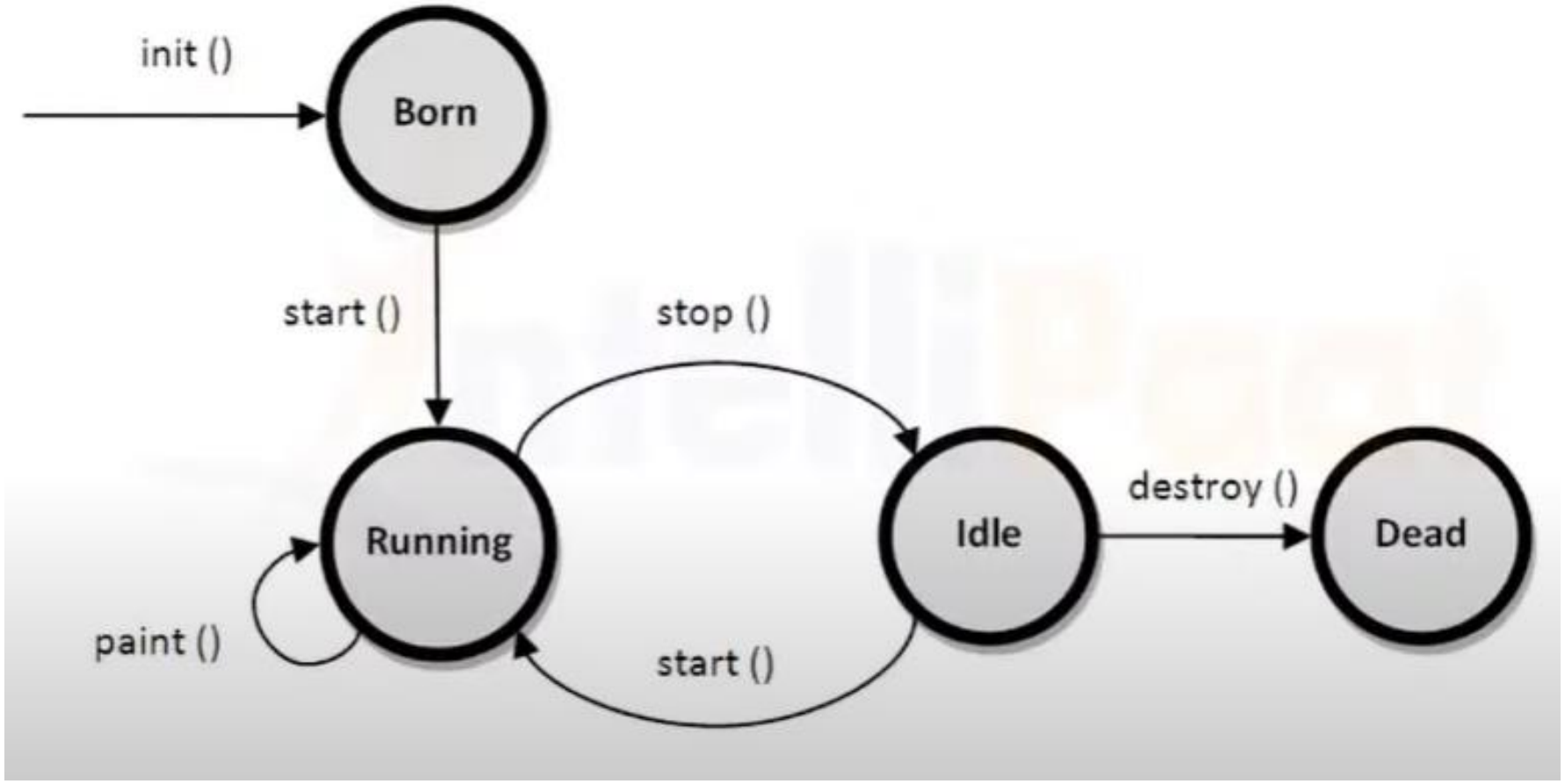
Life Cycle of an Applet+

Five Methods in the Applet class that gives us the framework on which you build any serious applet –

4. **stop()** – method is automatically called when the user moves off the page on which the applet sits or when the browser is minimized. It can, therefore, be called repeatedly in the same applet.

5. **destroy()** – method is only called when the browser shuts down normally. Because applets are meant to live on an HTML page, you **should not** normally leave resources behind after a user leaves the page that contains the applet.

Life Cycle of an Applet+



(intellipaaat. 2021)

Note

int(), start(), stop() and **destroy()** methods belongs to the java.awt.applet.**Applet Class**

While

paint() method belong to java.awt.**Component Class**.

Example of Applet program

Following is a simple applet named hwApplet.java

```
import java.applet.*;
import java.awt.*;

 public class hwApplet extends Applet{
     public void init(){
          //more code here
     }
     public void paint(Graphics g){
     //more code here
     }
}
```

Important Imports

The following import statements bring the classes into the scope of our applet class –

1. java.applet.Applet;
2. java.awt.Graphics;

Without those import statements, the Java compiler would not recognize the classes Applet and Graphics, which the applet class refers to.

The Applet Class

Every applet is an extension of the *java.applet.Applet class or javax.swing.JApplet class*. The base Applet class provides methods that a derived Applet class may call to obtain information and services from the browser context.

These include methods that do the following

1. Get applet parameters
2. Get the location of the HTML file that contains the applet
3. Get the location of the applet class directory

The Applet Class +

Every applet is an extension of the *java.applet.Applet class or javax.swing.JApplet class*. The base Applet class provides methods that a derived Applet class may call to obtain information and services from the browser context.

4. Print a status message in the browser
5. Fetch an image
6. Fetch an audio clip
7. Play an audio clip
8. Resize the applet

The Applet Class ++

Additionally, the Applet class provides an interface by which the viewer or browser obtains information about the applet and controls the applet's execution. The viewer may

1. Request information about the author, version, and copyright of the applet
2. Request a description of the parameters the applet recognizes
3. Initialize the applet

The Applet Class +++

4. Start the applet's execution
5. Stop the applet's execution
6. Destroy the applet

The Applet class provides default implementations of each of these methods. Those implementations may be overridden as necessary.

The "`hwApplet`" applet above is a half complete as it stands because it can actually run. Howerver some modifications are required to improve on its dispay by overriding some methods such as init(), paint() method etc.

Implementing an Applet

An applet may be invoked by embedding directives in an HTML file and viewing the file through an applet viewer of a Java-enabled browser. using the Applet viewer.

The <applet> tag is the basis for embedding an applet in an HTML file. Following is an example that invokes the " `hwApplet` " applet

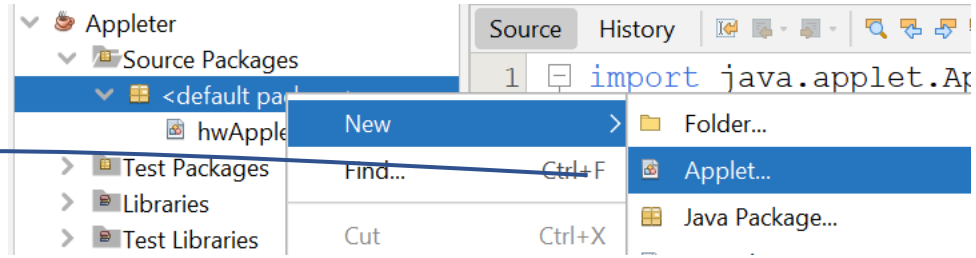
Implementation through IDE

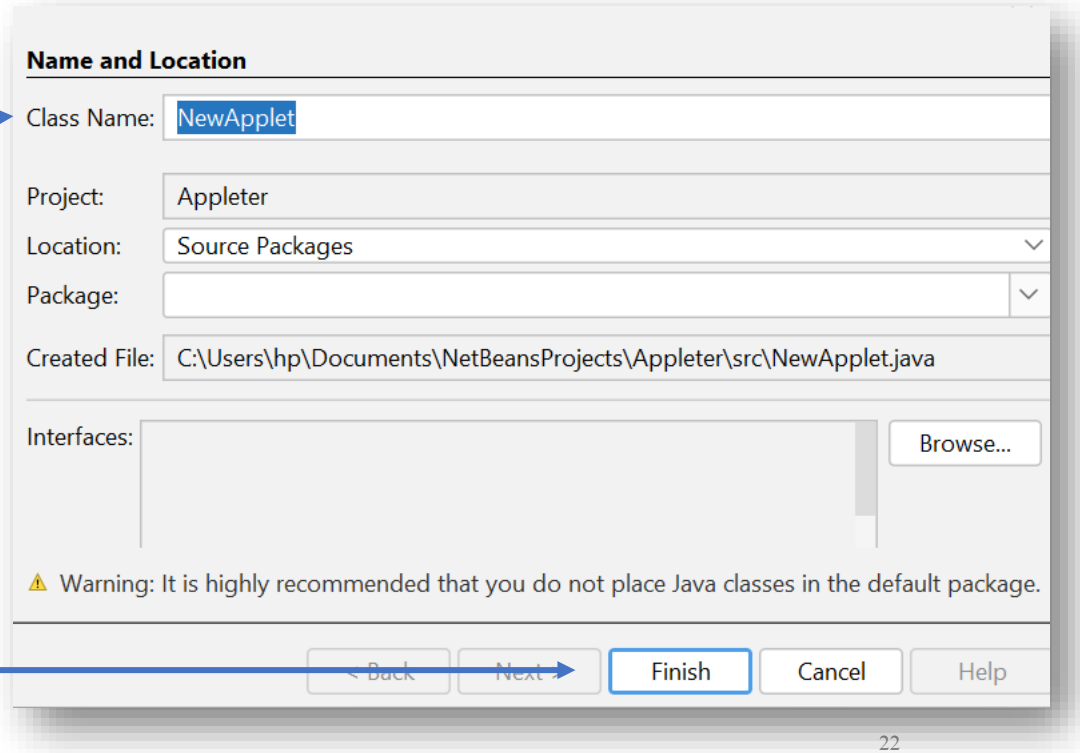
Before we run an Applet, lets first create one.

Note. *You should be running on JDK 8 NOT later version, for your applets to run*

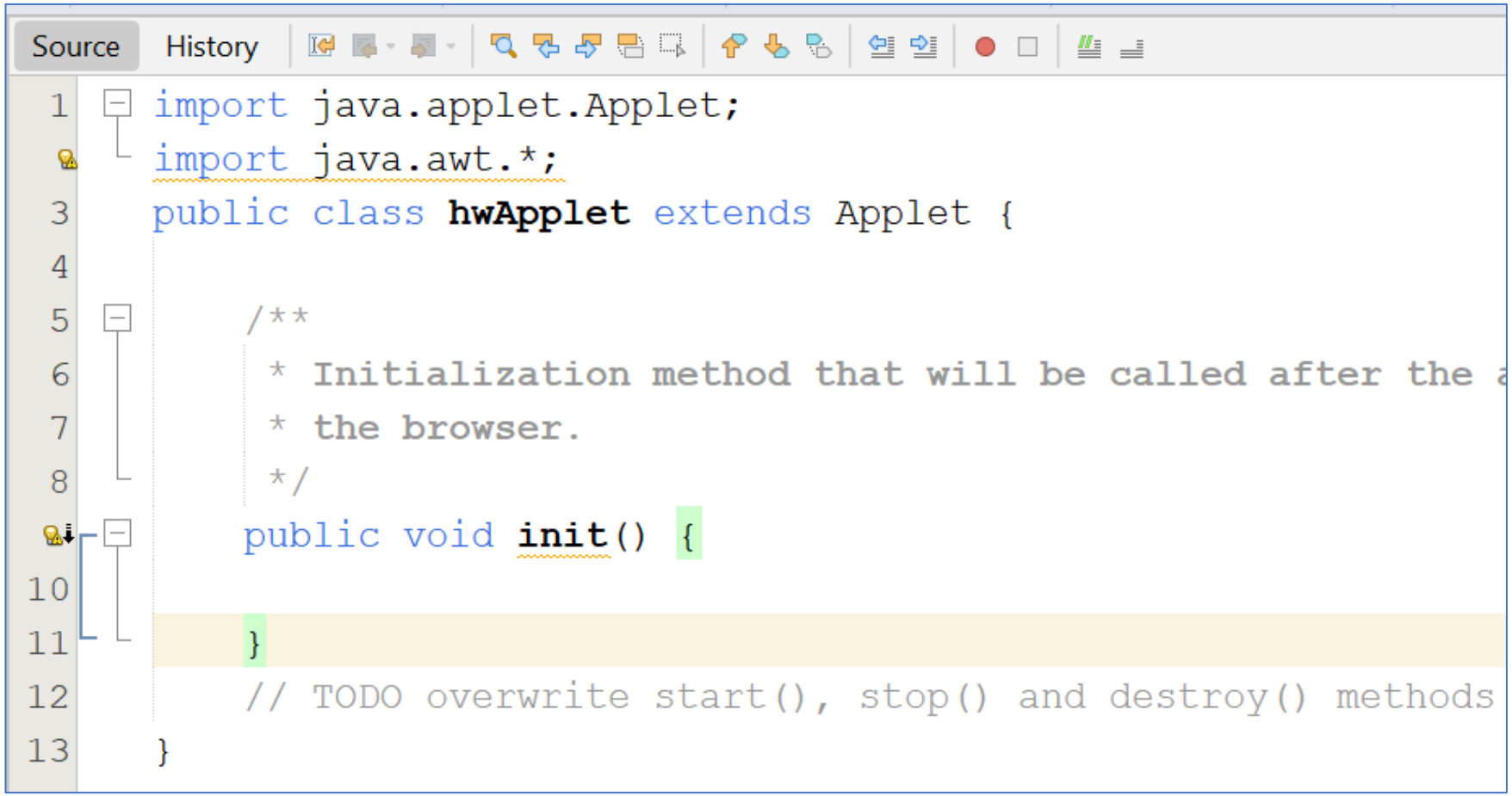
Steps of creating an Applet.

1. Create a project without a main() method
2. Open the source packages, then right click on the default package, **Point** at **New** then **Click Applet**
3. Enter Applet Name in the Class Name Text Box
4. Click Finish





Implementation through IDE-default code



This program should be able to run by now but will print nothing..

Make sure you are running on JDK 8 but not later version since Applet class is deprecated in Later JDK versions

Example of Applet program

Following is a simple applet named hwApplet.java —

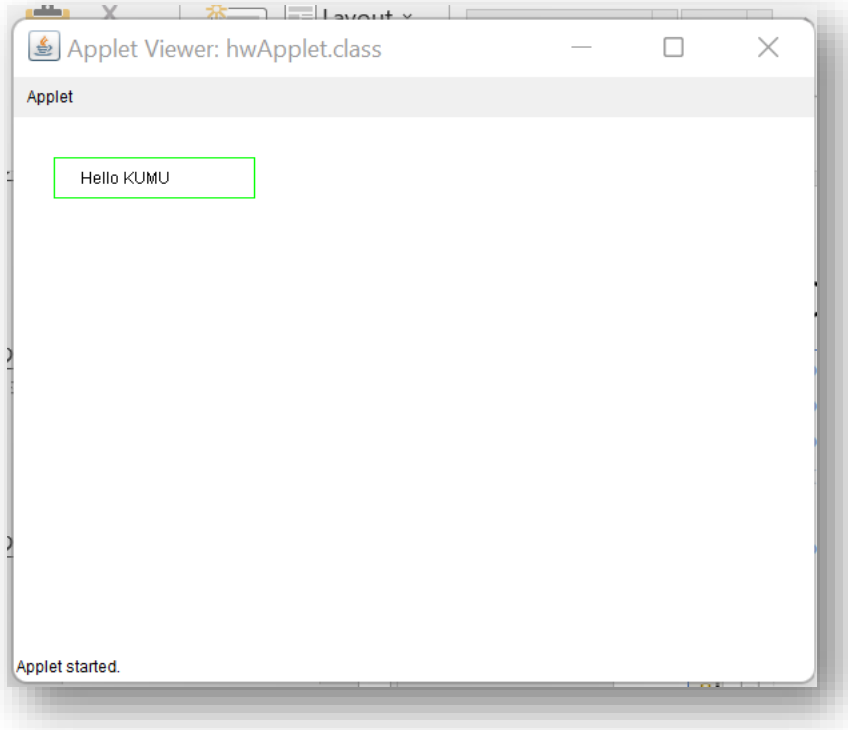
```
import java.applet.*;
import java.awt.*;

 public class hwApplet extends Applet{
     public void init(){
          setForeground(Color.BLACK)
     }
     public void paint(Graphics g){
     g.drawString ("Hello KUMU", 50,50 );
 }
}
```

Run the code normally

Code Running on Netbeans with some modifications

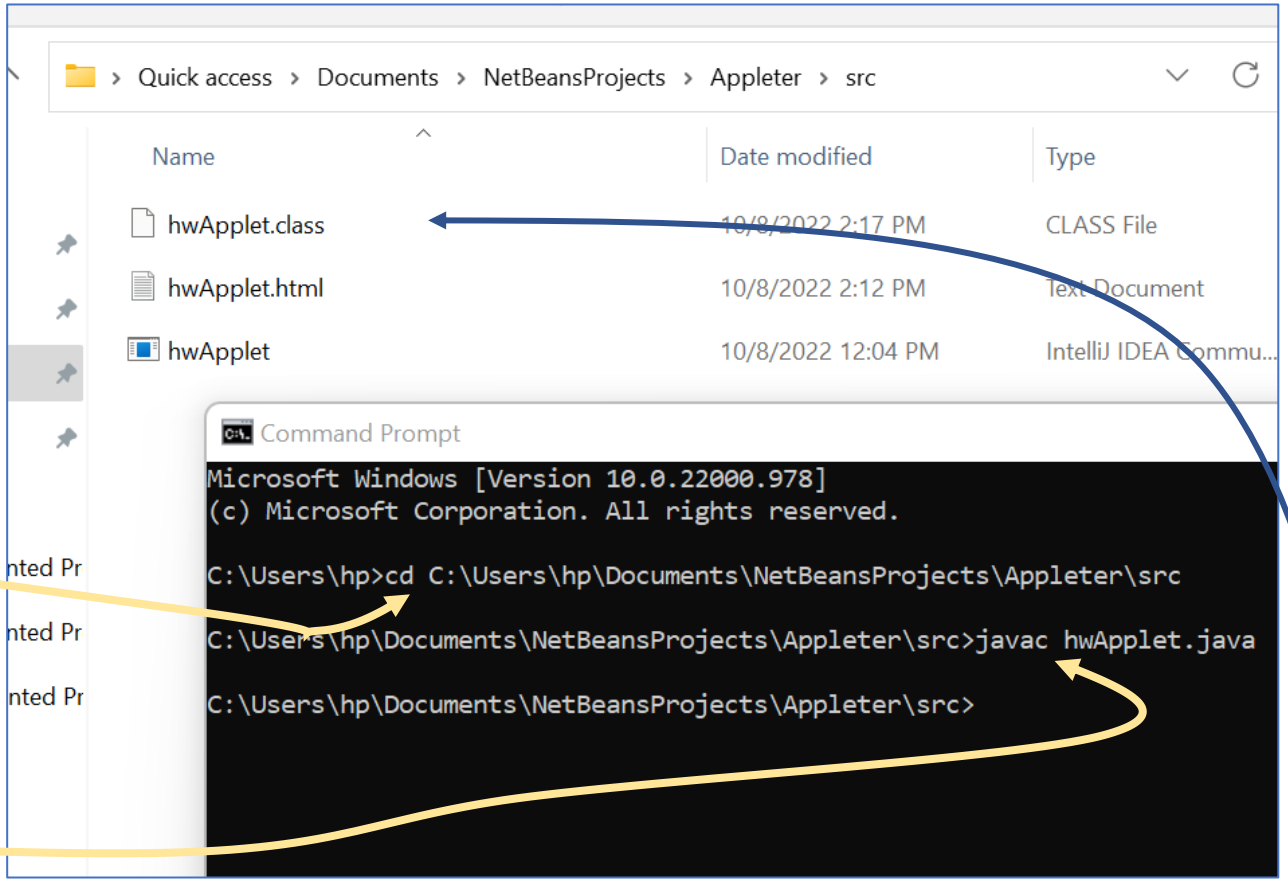
```
1  import java.applet.Applet;
2  import java.awt.Color;
3  import java.awt.Graphics;
4  /*<applet code =hwApplet width = 600 height = 400></applet>
5  */
6  public class hwApplet extends Applet {
       public void init() {
8          setForeground(Color.BLACK);
9      }
       public void paint(Graphics g){
11         g.drawString ("Hello KUMU", 50, 50);
12
13         g.setColor(Color.GREEN);
14         g.drawRect(30, 30, 150, 30);
15 }
16 }
```



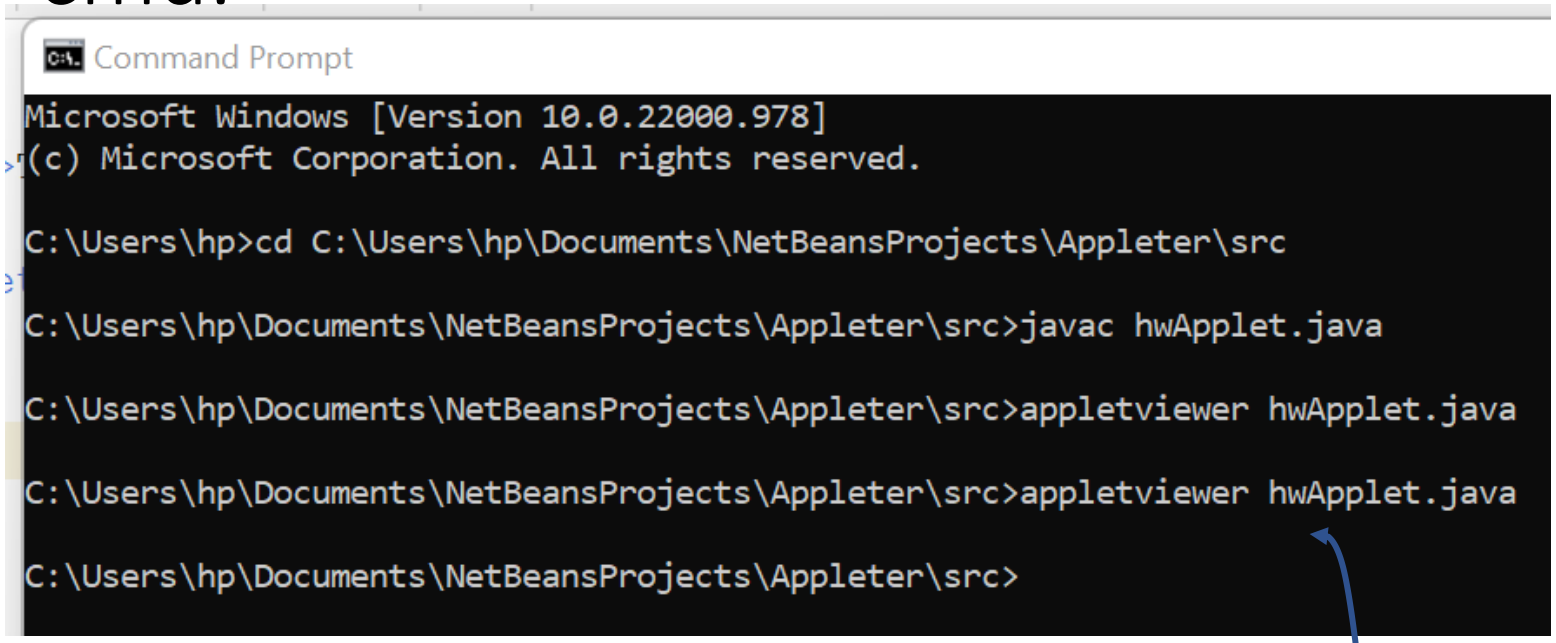
Implementing same Applet code in using cmd+ Compile java code with cmd.

Now we can compile and run the code

1. Copy the address of your hwApplet.java again
2. Open **cmd**
3. Type cd leave a space then paste the address you copied
4. Press Enter
5. Now type Javac leave space then type hwApplet.java
6. Another File Called hwApplet.class should be in the above address.

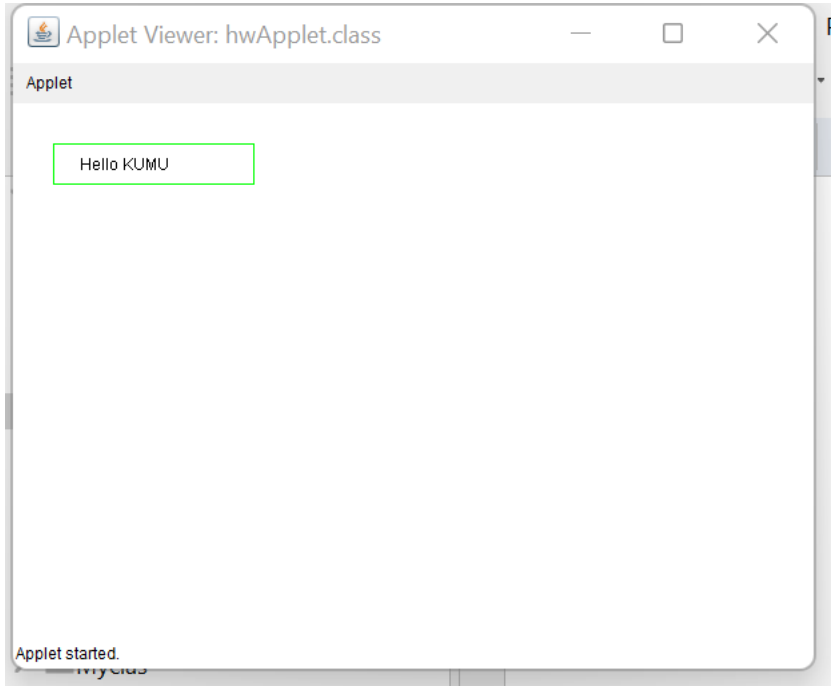


Implementing same Applet code in using cmd+ Compile java code with cmd.



Enter appletviewer hwApplet.java in cmd. To run the applet

Press Enter Key



Implementing Applet though html code.

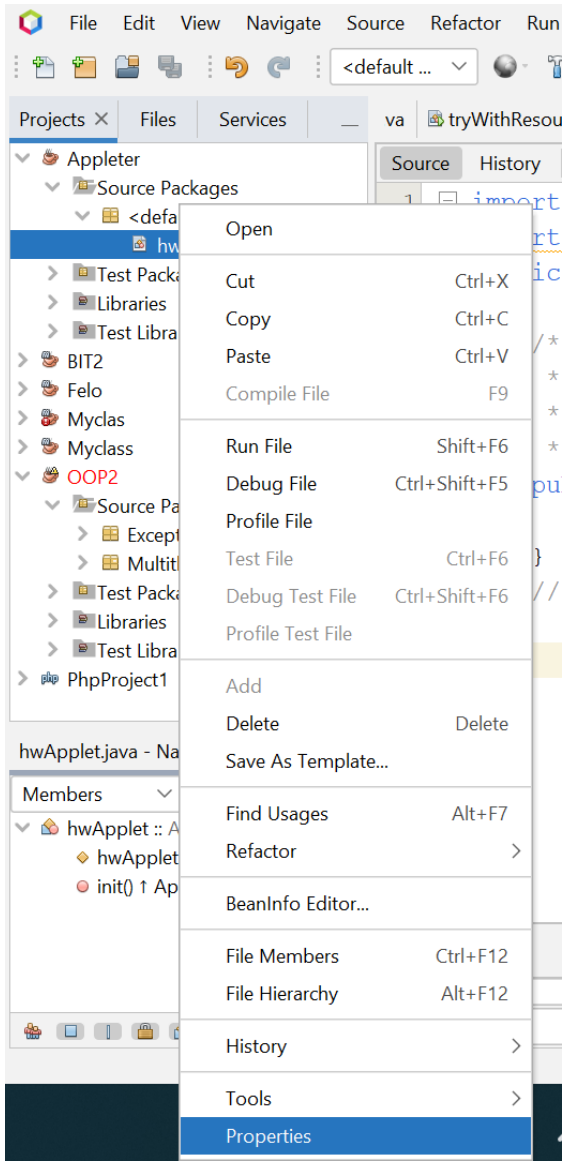
We said one of the ways to run an applet is embedding Applet code in HTML file. But before we can do that lets create an HTML file with some sample code.

We now need to create an HTML file in the same directory where our Applet file called hwApplet.java is.

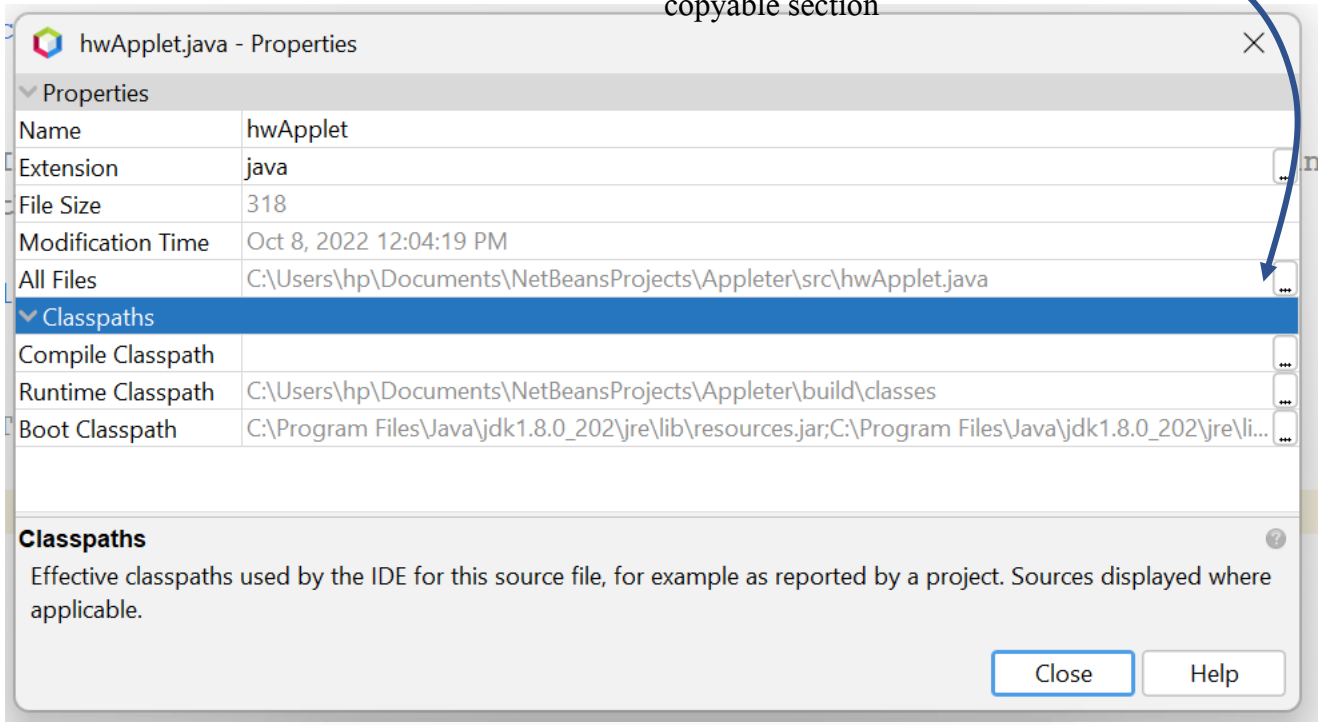
If you can't locate where the above file is stored on your computer, kindly right click on your derived Aplpet class and click properties.

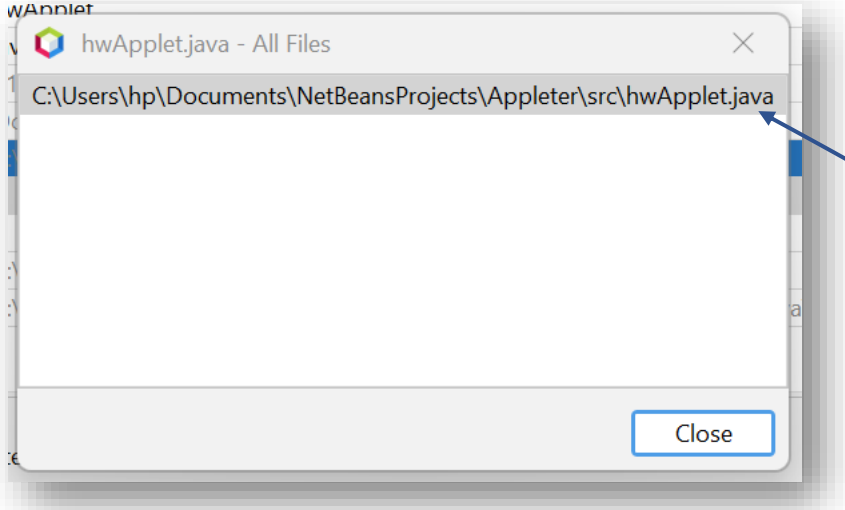
This will allow you see and copy the address of you derived applet class then pate it in your windows explorer address bar.

Know where you Java file is located

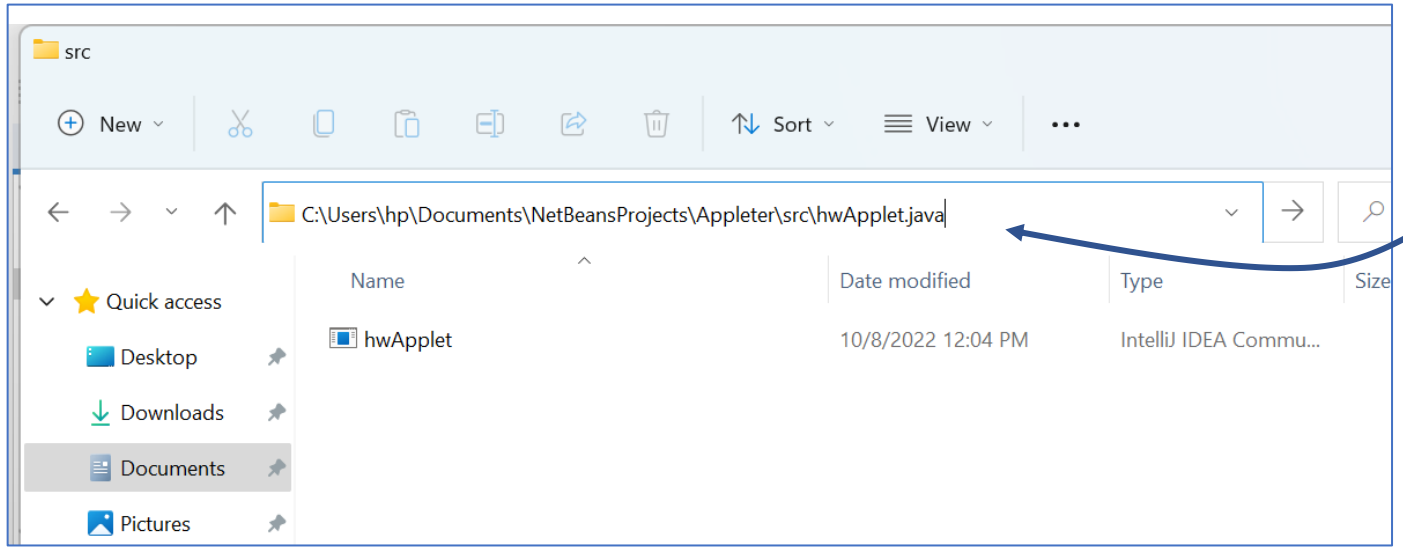


Click on this Icon to access copyable section



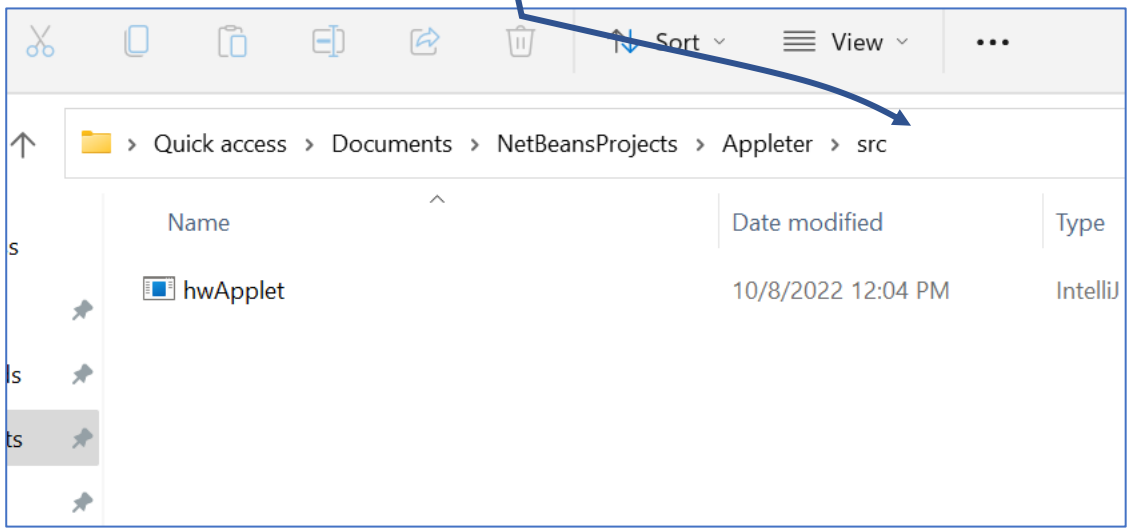


Highlight and copy this directory



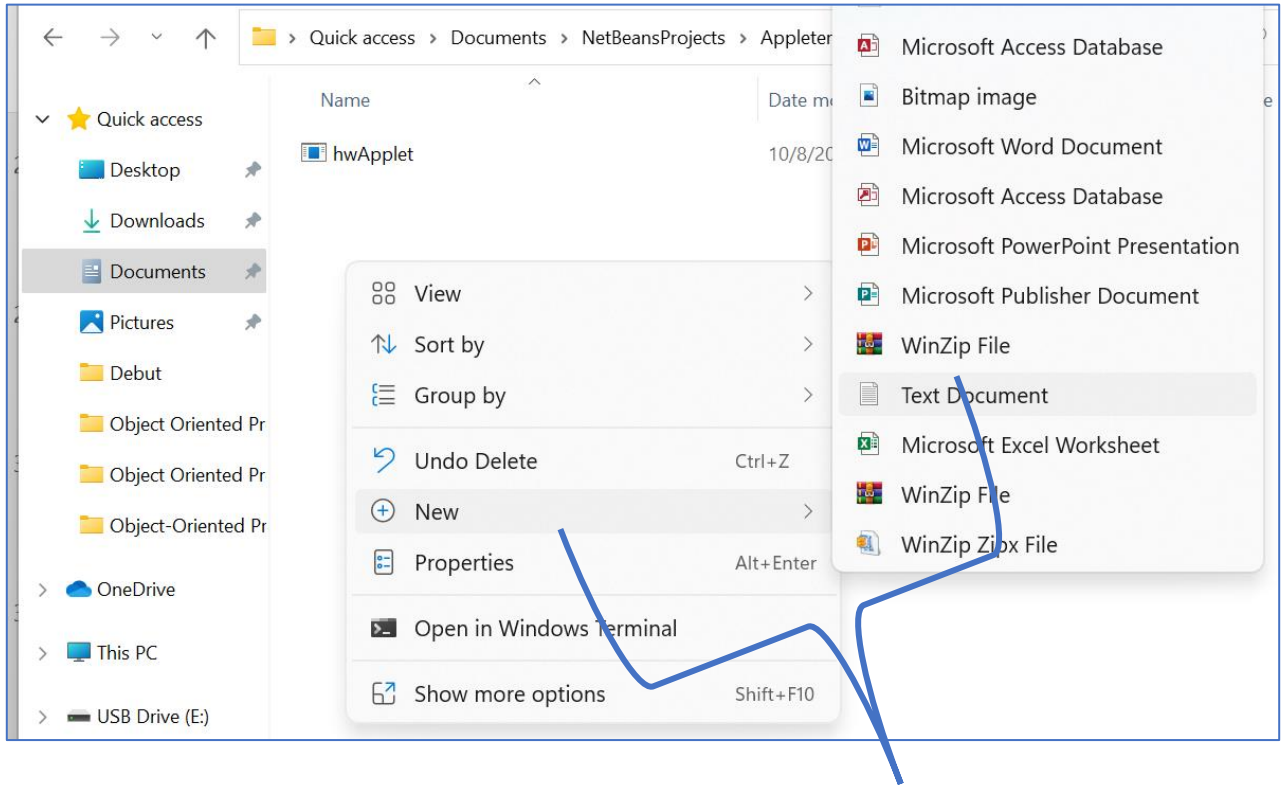
Paste your copied directory address in the Windows Explorer address bar

Remove the name of your derived applet e.g hwApplet.jaa from the pasted address then press Enter Key.

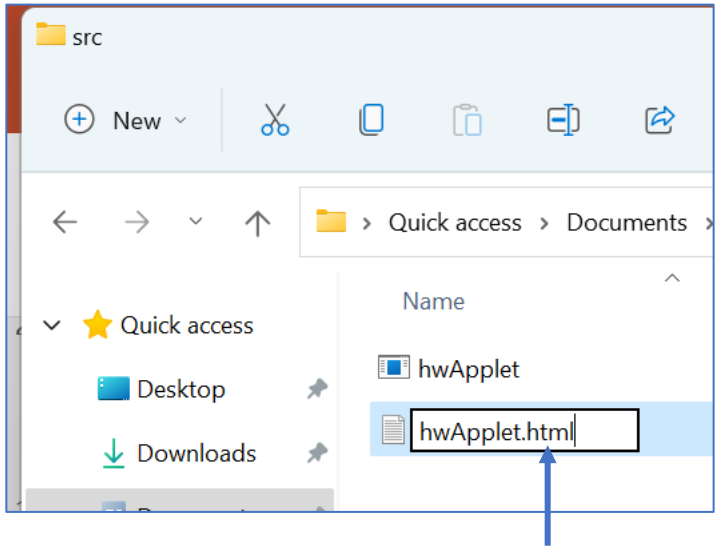


Now create a new hwApplet.html file

Now create a new hwApplet.html file



Right click on any empty space within the Src folder you are in, then point at New, then click Text Document



Name the file as hwApplet.html

Then double press Enter Key to open the file.

Now Enter the code below and resave you file as *.html

Implementing Applet though html code.

```
<html>

      <title>The Hello, World Applet</title>

      <applet code = hwApplet.class width = 400 height = 400></applet>

</html>
```

Implementing Applet though html code.+ hwApplet.html opened

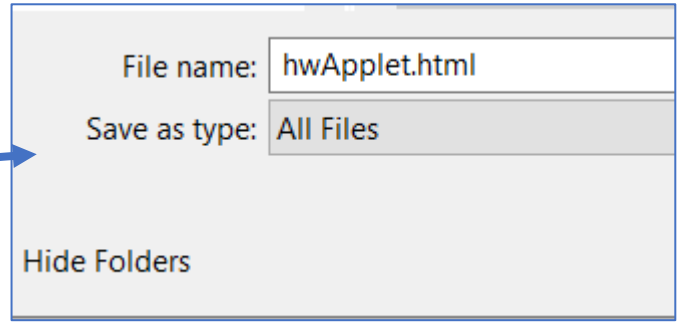
File Edit Search View Encoding Language Settings Tools Macro Run Plugins

Amoit.o | hwApplet.html

```
1  <html>
2
3      <title>The Hello, World Applet</title>
4
5      <applet code = hwApplet.class width = 600 height = 400>
6         </applet>
7
8
9  </html>
```

Re-Save As the above name.html making sure

Save as type is **All Files.**.



Java GUI Event Handling

Java GUI Event Handling

Any program that uses GUI (graphical user interface) such as Java application written for windows, is event driven. Event describes the change in state of any object.

For Example : Pressing a button, Entering a character in Textbox, Clicking or Dragging a mouse, etc.

Components of Event Handling

Event handling has three main components,

1. **Events :** An event is a change in state of an object.
2. **Events Source :** Event source is an object that generates an event.
3. **Listeners :** A listener is an object that listens to the event. A listener gets notified when an event occurs.

How Events are handled?

A source generates an Event and send it to one or more listeners registered with the source. Once event is received by the listener, they process the event and then return.

Events are supported by a number of Java packages, like

1. **java.util**,
2. **java.awt** and
3. **java.awt.event**.

Important Event Classes and Interface

Event Classes	Description	Listener Interface
ActionEvent	Generated when button is pressed, menu-item is selected, list-item is double clicked	ActionListener
MouseEvent	generated when mouse is dragged, moved,clicked,pressed or released and also when it enters or exit a component	MouseListener
KeyEvent	generated when input is received from keyboard	KeyListener
ItemEvent	generated when check-box or list item is clicked	ItemListener
TextEvent	generated when value of textarea or textfield is changed	TextListener
MouseWheelEvent	generated when mouse wheel is moved	MouseWheelListener

Important Event Classes and Interface

Event Classes	Description	Listener Interface
WindowEvent	generated when window is activated, deactivated, deiconified, iconified, opened or closed	WindowListener
ComponentEvent	generated when component is hidden, moved, resized or set visible	ComponentEventListener
ContainerEvent	generated when component is added or removed from container	ContainerListener
AdjustmentEvent	generated when scroll bar is manipulated	AdjustmentListener
FocusEvent	generated when component gains or loses keyboard focus	FocusListener

Steps to handle events

1. Implement appropriate interface in the class.
2. Register the component with the listener.

Example.

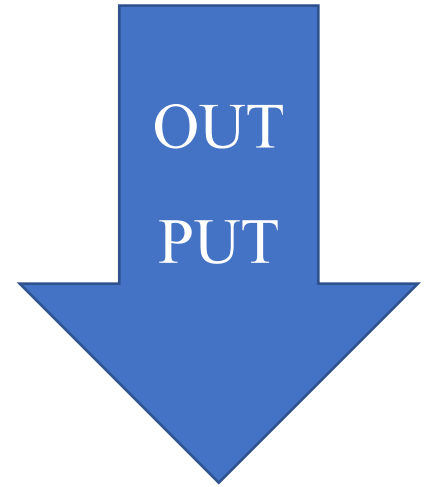
The following program shows us the mouse key states.

Example of Event Handling Program

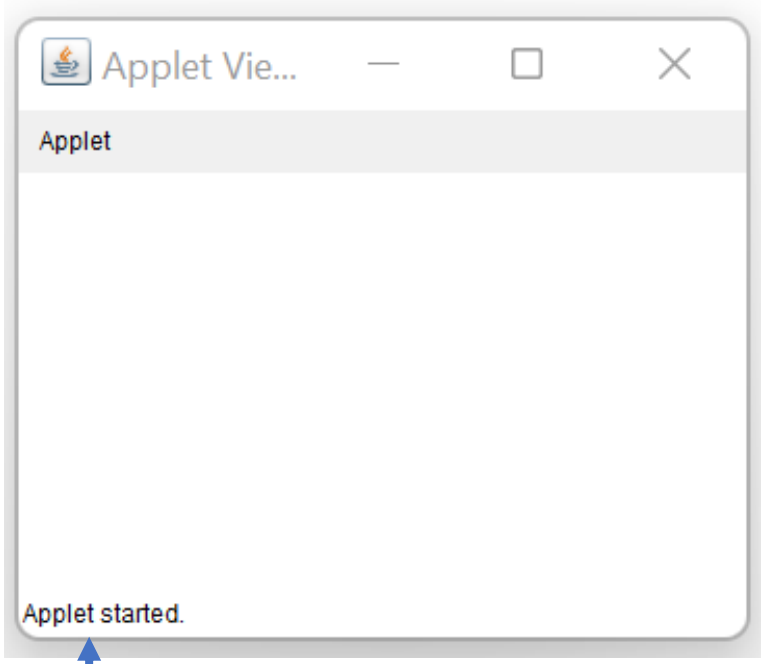
```
import java.awt.*; import java.awt.event import java.applet.*;
public class SApplet extends Applet implements KeyListener {
        String msg="";
        public void init() {
        addKeyListener(this);
        }
        public void keyPressed(KeyEvent k) {
        showStatus("KeyPressed");
        }
        public void keyReleased(KeyEvent k) {
        showStatus("KeyRealesed");
        }
        public void keyTyped(KeyEvent k) {
        msg = msg+k.getKeyChar(); repaint();
        }
        public void paint(Graphics g){
        setForeground(Color.BLACK);
        g.drawString(msg, 20, 40);
        }
}
```

Example of Event Handling Program

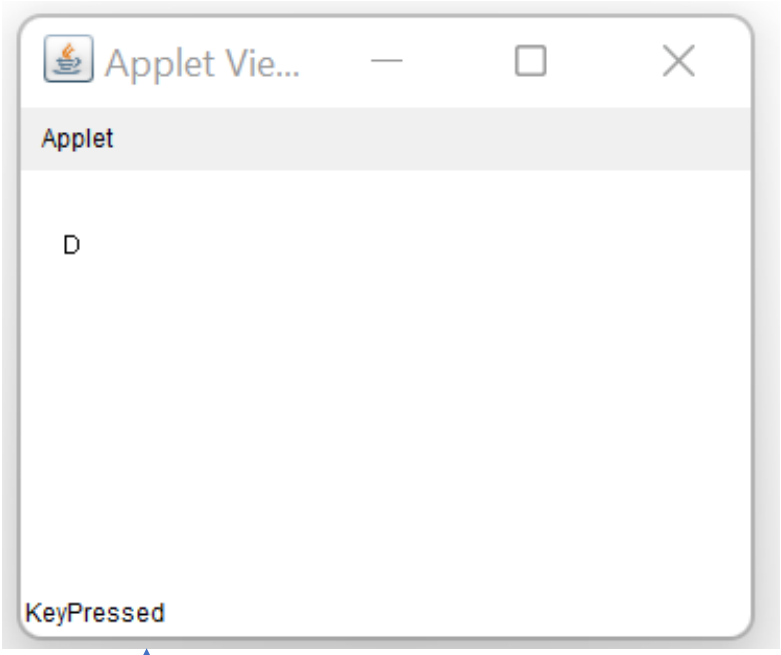
```
1  import java.applet.Applet;import java.awt.*; import java.awt.event.*;
2  public  class SApplet extends Applet implements KeyListener{
3  String msg="";
4          public void init() {
5          addKeyListener(this);
6          }
7          public void keyPressed(KeyEvent k) {
8          showStatus("KeyPressed");
9          }
10         public void keyReleased(KeyEvent k) {
11         showStatus("KeyRealesed");
12         }
13         public void keyTyped(KeyEvent k) {
14         msg = msg+k.getKeyChar();
15         repaint();
16         }
17         public void paint(Graphics g) {
18         setForeground(Color.BLACK);
19         g.drawString(msg, 20, 40);
20 }}
```



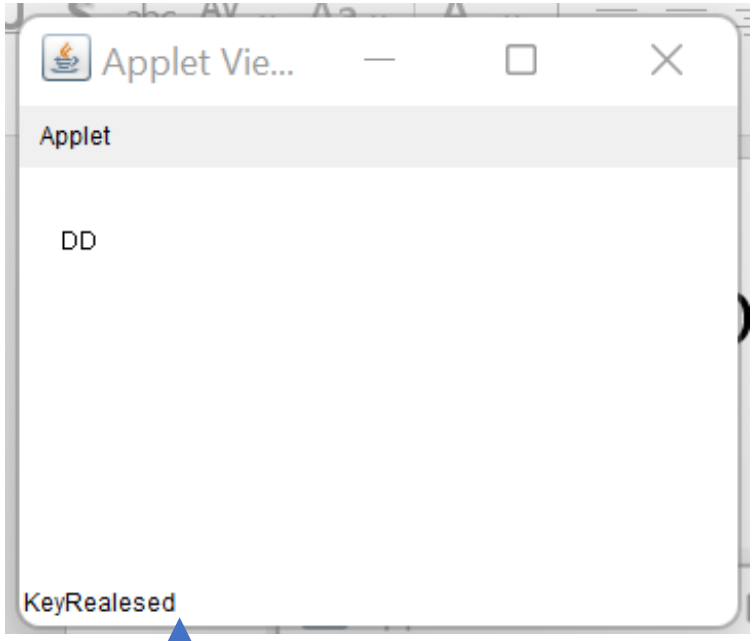
Example of Event Handling Program++ Output



State 1. Applet started



State 2. KeyPressed



State 3. KeyReleased

Java Abstract Windowing Toolkit-AWT

Java AWT is an API that contains large number of classes and methods to create and manage graphical user interface (GUI) applications.

The AWT was designed to provide a common set of tools for GUI design that could work on a variety of platforms.

Advantage of AWT. Preservation of individual platform look. The tools provided by the AWT are implemented using each platform's native GUI toolkit, hence preserving the look and feel of each platform.

But the **disadvantage** of such an approach is that GUI designed on one platform may look different when displayed on another platform that means AWT component are platform dependent.

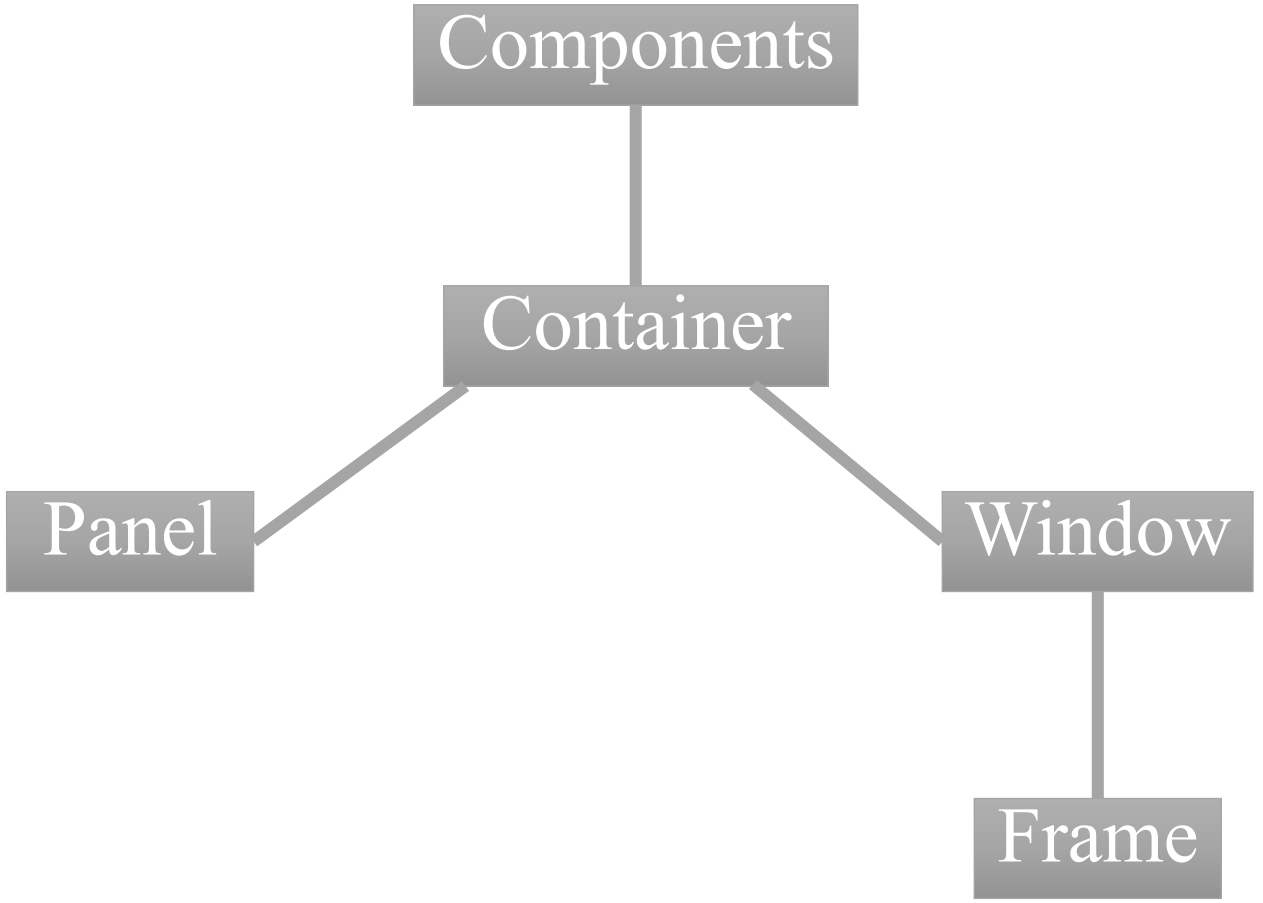
(studytonight.com, n.d.)

Java Abstract Windowing Toolkit-AWT+

AWT is the foundation upon which Swing is made i.e Swing is a improved GUI API that extends the AWT. But nowadays AWT is merely used because most GUI Java programs are implemented using Swing because of its rich implementation of GUI controls and light-weighted nature.

Java AWT Hierarchy

The hierarchy of Java AWT classes are given below, all the classes are available in **java.awt** package.



Java AWT Hierarchy+

Component class

is at the top of AWT hierarchy. It is an abstract class that encapsulates all the attributes of visual component. This class object is responsible for remembering the current foreground and background colors and the currently selected text font.

Container class

is a subclass of component class that contains other components such as; **button**, **Textfield**, **tables** etc.. This class keeps track of components that are added to another component.

Java AWT Hierarchy++

Panel class

is a concrete subclass of **Container** which is used for holding components. It does not contain title bar, menu bar or border..

Window class

is a subclass of Container that creates a top level window. It does not have borders and menu bar.

Java AWT Hierarchy++

Frame

is a subclass of **Window** class. It contains resizing canvas other components such as; Button, Title bar, Textfield, label etc.

In Java, most of the AWT applications are created using **Frame** window. Frame class has two different constructors, one without a parameter and the other that takes in a parameter

```
Frame() throws HeadlessException
Frame(String title) throws HeadlessException
```

Creating a Frame

There are two ways to create a Frame. i.e;-

1. By Instantiating Frame class
2. By extending Frame class

Lets create some Frame Windows.

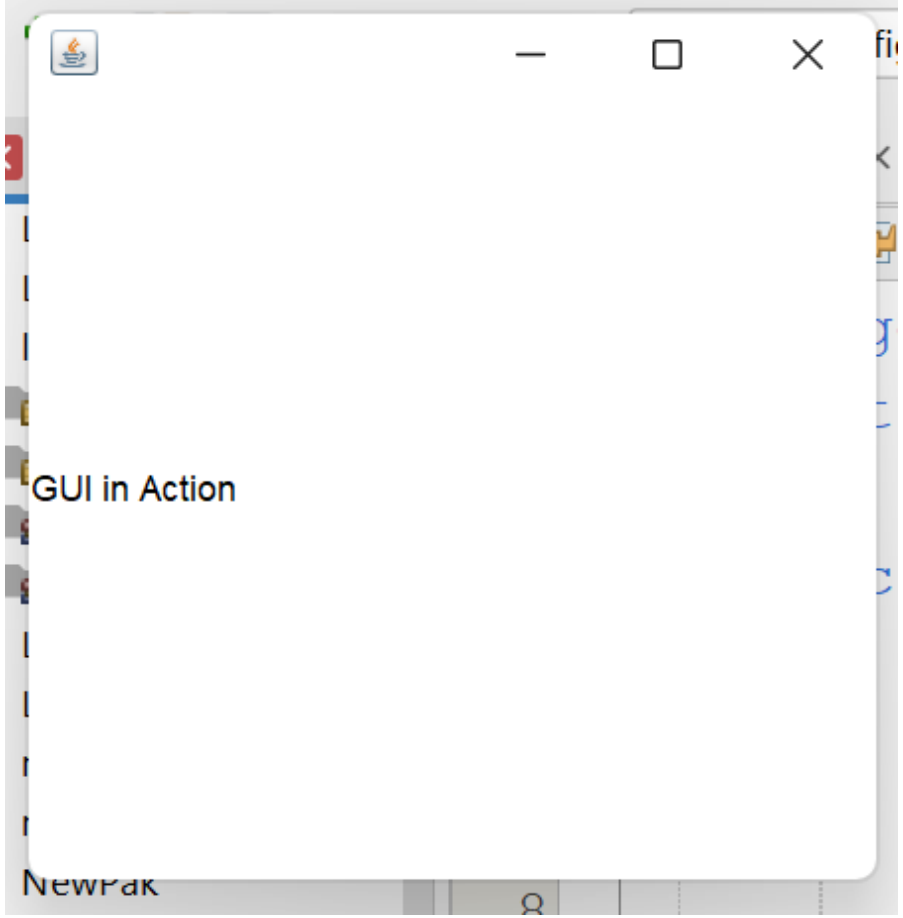
Creating Frame Window by Instantiating Frame class

```
import java.awt.*;
public class awtPro {
      awtPro(){
      Frame fm=new Frame(); //Creating a frame
      Label lb = new Label("GUI in Action");//Creating a label
      fm.add(lb); //adding label to the frame
      fm.setSize(300, 300); //setting frame size.
      fm.setVisible(true); //set frame visibilty true
      }
      public static  void main(String args[]) {
            awtPro ta = new awtPro();
      } }
```

Creating Frame Window by Instantiating Frame class

```
1   package GUI;
2   import java.awt.Frame;import java.awt.Label;import java.lang.*;
3
4   public class awtPro extends Frame{
5           awtPro(){
6           Frame fm=new Frame(); //Creating a frame
7           Label lb = new Label("GUI in Action"); //Creating a label
8           fm.add(lb); //adding label to the frame
9           fm.setSize(300, 300); //setting frame size.
10          fm.setVisible(true); //set frame visibilty true
11          }
12          public static void main(String args[]) {
13          awtPro ta = new awtPro();
14          } }
```

Creating Frame Window by Instantiating Frame class+Output



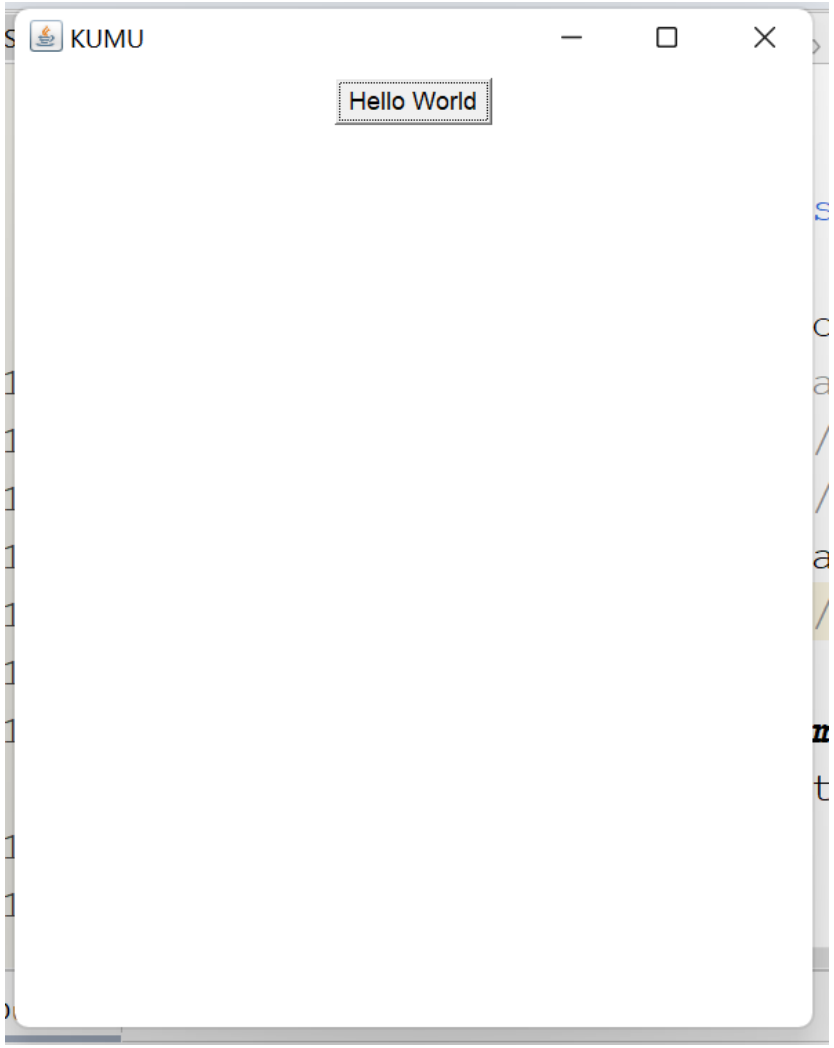
Creating Frame window by extending Frame class

```
import java.awt.*;
import java.awt.event.*;
public class awtPro1 extends Frame {
      public awtPro1(){
      Button btn=new Button("Sign in");
      add(btn); //adding a new Button.
      setSize(400, 500); //setting size.
      setTitle("KUMU"); //setting title.
      setLayout(new FlowLayout());
      setVisible(true); //set frame visibilty true.
      }
      public static void main (String[] args) {
       awtPro1 ta = new awtPro1(); //creating a frame.
      }
}
```

Creating Frame window by extending Frame class+Output

```
1   package GUI;
2   import java.awt.Button;import java.awt.FlowLayout;import java.awt.Frame;
3
4   public class awtPro1 extends Frame {
5           public awtPro1(){
6           Button btn=new Button("Hello World");
7           add(btn); //adding a new Button.
8           setSize(400, 500); //setting size.
9           setTitle("KUMU"); //setting title.
10          setLayout(new FlowLayout());
11          setVisible(true); //set frame visibilty true.
12          }
13          public static void main (String[] args) {
            awtPro1 ta = new awtPro1(); //creating a frame.
15  } }
```

Creating Frame window by extending Frame class+



Points to Remember

While creating a frame (either by instantiating or extending Frame class), Following two attributes are a must for visibility of the frame:

1. **setSize(int width, int height);**
2. **setVisible(true);**

When you create other component like Buttons, TextFields, etc. Then you need to add it to the frame by using the method - **add(Component's Object);**

You can add the following method also for resizing the frame - **setResizable(true);**

AWT Button

In Java, AWT contains a Button Class. It is used for creating a labelled button which can perform an action.

AWT Button Classs Declaration:

public class Button extends Component implements Accessible

Example: AWT Button Program

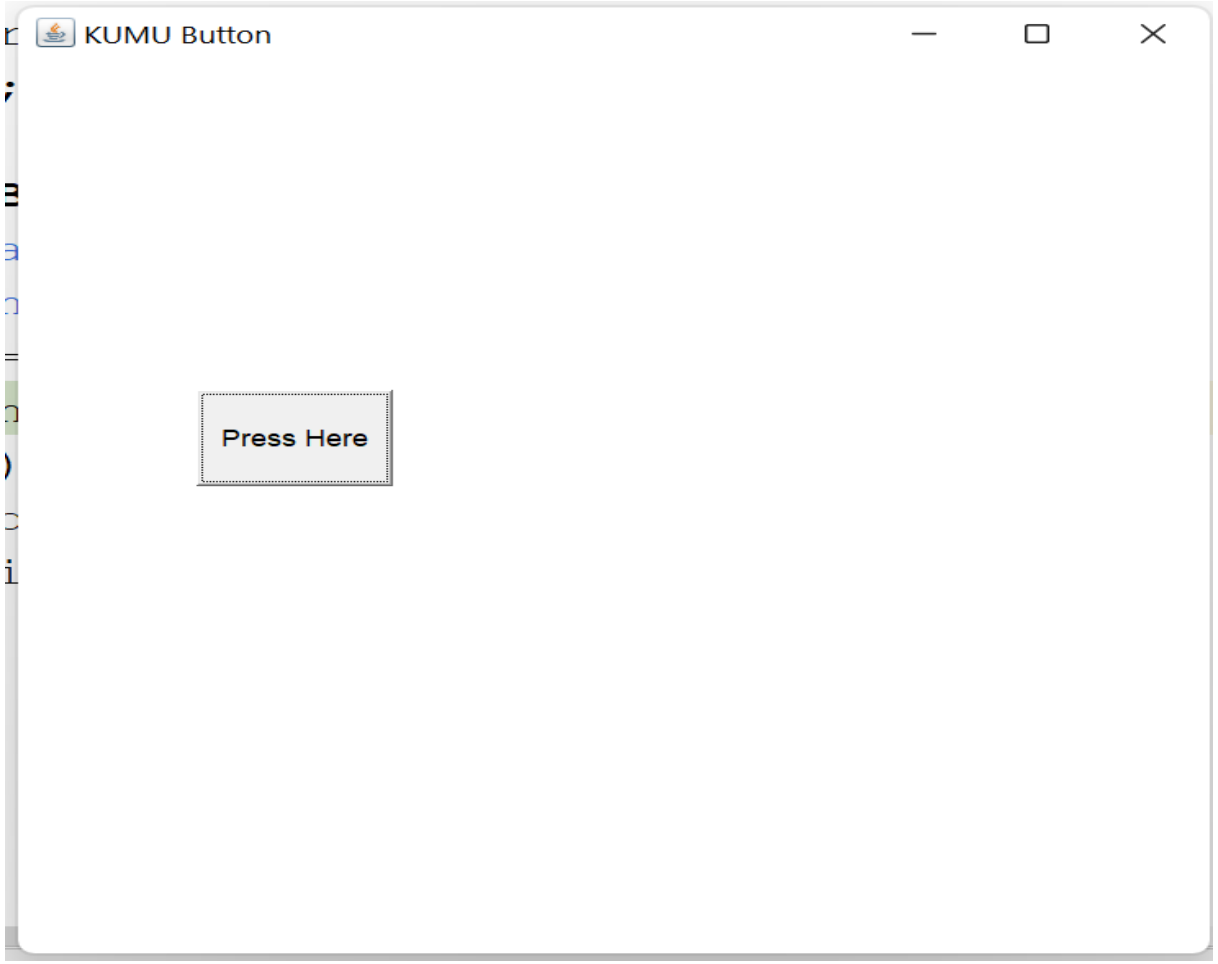
Lets take an example to create a button and it to the frame by providing coordinates.

```
import java.awt.*;
      public class ButtonPro {
      public static void main(String[] args) {
      Frame f1=new Frame("KUMU Button ");
      Button b1=new Button("Press Here");
      b1.setBounds(80,200,80,50);
      f1.add(b1);
      f1.setSize(500,500);
      f1.setLayout(null);
      f1.setVisible(true);
} }
```

Example: AWT Button Program+

```
1   package GUI;
2   import java.awt.Frame;
3   import java.awt.*;
4
5       public class ButtonPro{
6           public static void main(String[] args) {
7           Frame f1=new Frame("KUMU Button ");
8           Button b1=new Button("Press Here");
9           b1.setBounds(80,200,80,50);
10          f1.add(b1); f1.setSize(500,500);
11          f1.setLayout(null);
12          f1.setVisible(true);
13  } }
```

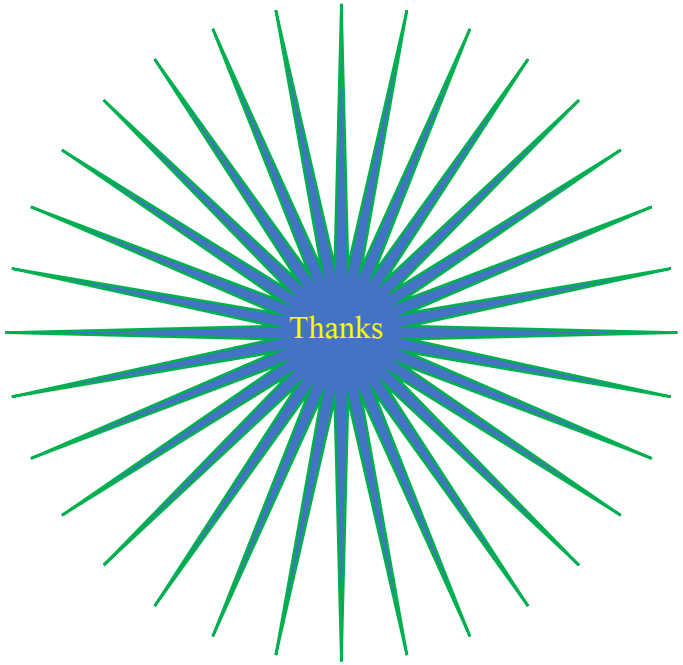
Output: Example: AWT Button Program ++ Output



Summary

1. Graphical User Interface -GUI Introduction,
2. Applet(Difinition, Created and implemented),
3. Event Handling-Look at various classes that support event handling.
4. AWT –Abstruct Windowing Toolkit – Difinintion, saw the AWT components and their classes.

Thank you for Listening



References

Wikimedia Foundation. (2022, September 21). *Graphical user interface*. Wikipedia. Retrieved September 29, 2022, from https://en.wikipedia.org/wiki/Graphical_user_interface

Computer Hope. (2021, April 12). *What is a GUI (graphical user interface)?* What is a GUI (Graphical User Interface)? Retrieved October 7, 2022, from https://www.computerhope.com/jargon/g/gui.htm#elements

intellipaaat. (2021, November 8). *Java applet | Applet in Java | Java Applets for Beginners |java applet tutorial | intellipaat*. YouTube. Retrieved October 7, 2022, from https://www.youtube.com/watch?v=3pr5OOfv_Dc