Unit - VApplet: Applet Life Cycle, Applet Tag, Adding Applet to HTML File; Passing Parameters toApplets, Getting Input From User. AWT : AWT Classes, Working With Frame Windows,Working With Graphics, Working With Colour, Adding And Removing Controls,Responding To Controls, Labels, Buttons, Checkbox, Checkbox Group, Choice Control,Lists, Text Field, Text Area. Menus, Dialog Box, Handling Events.

Applet

Applet is a special type of program that is embedded in the webpage to generate the dynamic content. It runs inside the browser and works at client side.

* Applets are small Java applications that can be accessed on an Internet server, transported over Internet, and can be automatically installed and run as apart of a web document.
* After a user receives an applet, the applet can produce a graphical user interface. It has limited access to resources so that it can run complex computations without introducing the risk of viruses or breaching data integrity.

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

* An applet is a Java class that extends the java.applet.Applet class.
* A main() method is not invoked on an applet, and an applet class will not define main().
* Applets are designed to be embedded within an HTML page.
* When a user views an HTML page that contains an applet, the code for the applet is downloaded to the user's machine.
* A JVM is required to view an applet. The JVM can be either a plug-in of the Web browser or a separate runtime environment.

Advantage of Applet

There are many advantages of applet. They are as follows:

* It works at client side so less response time.
* Secured
* It can be executed by browsers running under many plateforms, including Linux, Windows, Mac Os etc.

Drawback of Applet

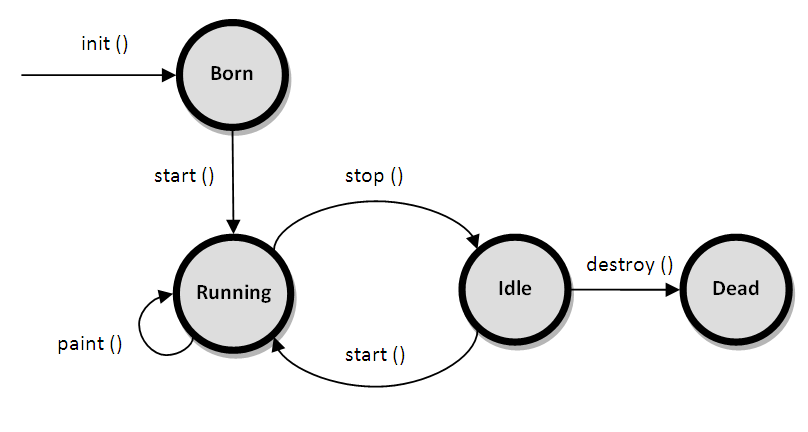
* Plugin is required at client browser to execute applet.

Lifecycle of Java Applet

1. Applet is initialized.
2. Applet is started.
3. Applet is painted.
4. Applet is stopped.
5. Applet is destroyed.

### Lifecycle methods for Applet:

The java.applet.Applet class 4 life cycle methods and java.awt.Component class provides 1 life cycle methods for an applet.



java.applet.Applet class

For creating any applet java.applet.Applet class must be inherited. It provides 4 life cycle methods of applet.

1. **public void init():** is used to initialized the Applet. It is invoked only once.
2. **public void start():** is invoked after the init() method or browser is maximized. It is used to start the Applet.
3. **public void stop():** is used to stop the Applet. It is invoked when Applet is stop or browser is minimized.
4. **public void destroy():** is used to destroy the Applet. It is invoked only once.

java.awt.Component class

The Component class provides 1 life cycle method of applet.

1. **public void paint(Graphics g):** is used to paint the Applet. It provides Graphics class object that can be used for drawing oval, rectangle, arc etc.

|  |  |
| --- | --- |
| **First.java** | **First.html** |
| 1. //First.java 2. **import** java.applet.Applet; 3. **import** java.awt.Graphics; 4. **public** **class** First **extends** Applet{ 6. **public** **void** paint(Graphics g){ 7. g.drawString("welcome",150,150); 8. } 10. } | 1. <html> 2. <body> 3. <applet code="First.class" width="300" height="300"> 4. </applet> 5. </body> 6. </html>   **c:\>javac First.java**  **c:\>appletviewer First.html** |
| class must be public because its object is created by Java Plugin software that resides on the browser | |

## Commonly used methods of Graphics class:

1. **public abstract void drawString(String str, int x, int y):** is used to draw the specified string.
2. **public void drawRect(int x, int y, int width, int height):** draws a rectangle with the specified width and height.
3. **public abstract void fillRect(int x, int y, int width, int height):** is used to fill rectangle with the default color and specified width and height.
4. **public abstract void drawOval(int x, int y, int width, int height):** is used to draw oval with the specified width and height.
5. **public abstract void fillOval(int x, int y, int width, int height):** is used to fill oval with the default color and specified width and height.

|  |  |
| --- | --- |
| Demo.java | Demo.html |
| 1. **import** java.applet.Applet; 2. **import** java.awt.\*; 4. **public** **class** Demo **extends** Applet{ 6. **public** **void** paint(Graphics g){ 7. g.setColor(Color.red); 8. g.drawString("Welcome",50, 50); 9. g.drawLine(20,30,20,300); 10. g.drawRect(70,100,30,30); 11. g.fillRect(170,100,30,30); 12. g.drawOval(70,200,30,30);   }  } | 1. <html> 2. <body> 3. <applet code="Demo.class" width="300" height="300"> 4. </applet> 5. </body> 6. </html> |

# Parameter in Applet

We can get any information from the HTML file as a parameter. For this purpose, Applet class provides a method named getParameter(). Syntax:

|  |  |
| --- | --- |
| UseParam.java | UseParam.html |
| 1. **import** java.applet.Applet; 2. **import** java.awt.Graphics; 4. **public** **class** UseParam **extends** Applet{ 6. **public** **void** paint(Graphics g){ 7. String str=getParameter("msg"); 8. g.drawString(str,50, 50); 9. } 11. } | 1. <html> 2. <body> 3. <applet code="UseParam.class" width="300" height="300"> 4. <param name="msg" value="Welcome to applet"> 5. </applet> 6. </body> 7. </html> |

**Type of Applet**

|  |  |
| --- | --- |
|  |  |

**Local Applets:**A local applet is the one that is stored on our own computer system.  When the Web-page has to find a local applet, it doesn't need to retrieve information from the Internet.  A local applet is specified by a path name and a file name as shown below in which the codebase attribute specifies a path name, whereas the code attribute specifies the name of the byte-code file that contains the applet's code.

**Remote Applets:**A remote applet is the one that is located on a remote computer system . This computer system may be located in the building next door or it may be on the other side of the world. No matter where the remote applet is located, it's downloaded onto our computer via the Internet. The browser must be connected to the Internet at the time it needs to display the remote applet. To reference a remote applet in Web page, we must know the applet's URL (where it's located on the Web) and any attributes and parameters that we need to supply. A local applet is specified by a url and a file name.

### Getting Input from the User in Java Applet

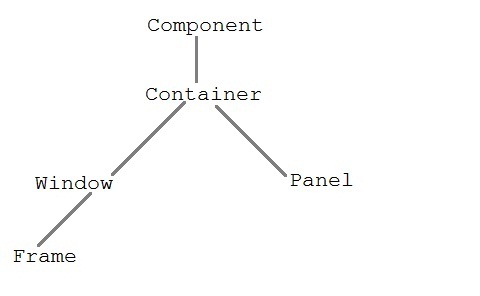
Applets work in graphical environment. Therefore applets treat input as text strings. We must first create an area of the screen in which user can type and edit input items. We can do this by using the TextField class of the applet package. The values of the fields can be given even editer after the creation of input fields. Next step is to retrieve the items from the fields for display of calculations.  
  
For any kinds of computation on the input field, we must convert it to the right form and the results again converted back to strings for display.

|  |  |
| --- | --- |
|  |  |
| import java.awt.\*;  import java.applet.\*;  public class UserInput extends Applet  {  TextField text1, text2;  public void init()  {  text1 = new TextField(8);  text2 = new TextField(8);  add(text1);  add(text2);  text1.setText(\"0\");  text2.setText(\"0\");  }  public void paint(Graphics g)  {  int x=0,y=0,z=0;  String s1,s2,s;  g.drawString(\"Input a number in each box \",10,50);  try  {  s1 = text1.getText();  x = Integer.parseInt(s1);  s2 = text2.getText();  y = Integer.parseInt(s2);  }  catch(Exception e) {}  z = x + y;  s = String.valueOf(z);  g.drawString(\"The Sum is : \",10,75);  g.drawString(s,100,75);  }  public boolean action(Event event, Object obj)  {  repaint();  return true;  }  } |  |

**AWT Classes :**

**Java AWT** (Abstract Window Toolkit) is *an API to develop GUI or window-based applications* in java. AWT contains large number of classes and methods that allows you to create and manage graphical user interface ( GUI ) applications, such as windows, buttons, scroll bars,etc.

Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavyweight i.e. its components are using the resources of OS.The java.awt [package](https://www.javatpoint.com/package) provides [classes](https://www.javatpoint.com/object-and-class-in-java) for AWT api such as [TextField](https://www.javatpoint.com/java-awt-textfield), [Label](https://www.javatpoint.com/java-awt-label), [TextArea](https://www.javatpoint.com/java-awt-textarea), RadioButton, [CheckBox](https://www.javatpoint.com/java-awt-checkbox), [Choice](https://www.javatpoint.com/java-awt-choice), [List](https://www.javatpoint.com/java-awt-list) etc.



**Component class**

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

**Container**

The Container is a component in AWT that can contain another components like [buttons](https://www.javatpoint.com/java-awt-button), textfields, labels etc. The classes that extends Container class are known as container such as Frame, Dialog and Panel.

**Window :** The window is the container that have no borders and menu bars. You must use frame, dialog or another window for creating a window.

**Panel :** The Panel is the container that doesn't contain title bar and menu bars. It can have other components like button, textfield etc.

**Frame :** The Frame is the container that contain title bar and can have menu bars. It can have other components like button, textfield etc.

**Points to Remember:**

1. While creating a frame (either by instantiating or extending Frame class), Following two attributes are must for visibility of the frame:
   * **setSize(int width, int height);**
   * **setVisible(true);**
2. When you create other components like Buttons, TextFields, etc. Then you need to add it to the frame by using the method - **add(Component's Object);**
3. You can add the following method also for resizing the frame - **setResizable(true);**

**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. |

**Event Handling**

|  |
| --- |
| Changing the state of an object is known as an event. For example, click on button, dragging mouse etc. The java.awt.event package provides many event classes and Listener interfaces for event handling. |

**Java Event classes and Listener interfaces**

|  |  |
| --- | --- |
| **Event Classes** | **Listener Interfaces** |
| ActionEvent | ActionListener |
| MouseEvent | MouseListener and MouseMotionListener |
| MouseWheelEvent | MouseWheelListener |
| KeyEvent | KeyListener |
| ItemEvent | ItemListener |

**AWT Button**

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.

|  |  |
| --- | --- |
|  |  |
| 1. **import** java.awt.\*; 2. **public** **class** ButtonExample { 3. **public** **static** **void** main(String[] args) { 4. Frame f=**new** Frame("Button Example"); 5. Button b=**new** Button("Click Here"); 6. b.setBounds(50,100,80,30); 7. f.add(b); 8. f.setSize(400,400); 9. f.setLayout(); 10. f.setVisible(**true**); 11. } 12. } | java awt button example 1 |

**Java AWT Button Example with ActionListener**

|  |  |
| --- | --- |
|  |  |
| 1. **import** java.awt.\*; 2. **import** java.awt.event.\*; 3. **public** **class** ButtonExample { 4. **public** **static** **void** main(String[] args) { 5. Frame f=**new** Frame("Button Example"); 6. **final** TextField tf=**new** TextField(); 7. tf.setBounds(50,50, 150,20); 8. Button b=**new** Button("Click Here"); 9. b.setBounds(50,100,60,30); 10. b.addActionListener(**new** ActionListener(){ 11. **public** **void** actionPerformed(ActionEvent e){ 12. tf.setText("Welcome to Java."); 13. } 14. }); 15. f.add(b);f.add(tf); 16. f.setSize(400,400); 17. f.setLayout(); 18. f.setVisible(**true**); 19. } 20. } | java awt button example 2 |

**AWT Label**

The [object](https://www.javatpoint.com/object-and-class-in-java) of 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 an application but a user cannot edit it directly.

|  |  |
| --- | --- |
|  |  |
| 1. **import** java.awt.\*; 2. **class** LabelExample{ 3. **public** **static** **void** main(String args[]){ 4. Frame f= **new** Frame("Label Example"); 5. Label l1; 6. l1=**new** Label("First Label."); 7. l1.setBounds(50,100, 100,30); 8. f.add(l1); 9. f.setSize(400,400); 10. f.setLayout(); 11. f.setVisible(**true**); 12. } 13. } | java awt label example 1 |

**AWT TextField**

The [object](https://www.javatpoint.com/object-and-class-in-java) of a TextField class is a text component that allows the editing of a single line text. It inherits TextComponent class.

|  |  |
| --- | --- |
|  |  |
| 1. **import** java.awt.\*; 2. **class** TextFieldExample{ 3. **public** **static** **void** main(String args[]){ 4. Frame f= **new** Frame("TextField Example"); 5. TextField t1; 6. t1=**new** TextField("Welcome."); 7. t1.setBounds(50,100, 200,30); 8. f.add(t1); 9. f.setSize(400,400); 10. f.setLayout(); 11. f.setVisible(**true**); 12. } 13. } | java awt textfield example 1 |

**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".

|  |  |
| --- | --- |
|  |  |
| 1. **import** java.awt.\*; 2. **public** **class** CheckboxExample 3. { 4. CheckboxExample(){ 5. Frame f= **new** Frame("Checkbox Example"); 6. Checkbox checkbox1 = **new** Checkbox("C++"); 7. checkbox1.setBounds(100,100, 50,50); 8. Checkbox checkbox2 = **new** Checkbox("Java", **true**); 9. checkbox2.setBounds(100,150, 50,50); 10. f.add(checkbox1); 11. f.add(checkbox2); 12. f.setSize(400,400); 13. f.setLayout(); 14. f.setVisible(**true**); 15. } 16. **public** **static** **void** main(String args[]) 17. { 18. **new** CheckboxExample(); 19. } 20. } | java awt checkbox example 1 |

**AWT List**

The object of List class represents a list of text items. By the help of list, user can choose either one item or multiple items. It inherits Component class.

|  |  |
| --- | --- |
|  |  |
| 1. **import** java.awt.\*; 2. **public** **class** ListExample 3. { 4. ListExample(){ 5. Frame f= **new** Frame(); 6. List l1=**new** List(5); 7. l1.setBounds(100,100, 75,75); 8. l1.add("Item 1"); 9. l1.add("Item 2"); 11. f.add(l1); 12. f.setSize(400,400); 13. f.setLayout(); 14. f.setVisible(**true**); 15. } 16. **public** **static** **void** main(String args[]) 17. { 18. **new** ListExample(); 19. } 20. } | java awt list example 1 |

**MenuItem and Menu**

The object of MenuItem class adds a simple labeled menu item on menu. The items used in a menu must belong to the MenuItem or any of its subclass.

The object of Menu class is a pull down menu component which is displayed on the menu bar. It inherits the MenuItem class.

|  |  |
| --- | --- |
|  |  |
| 1. **import** java.awt.\*; 2. **class** MenuExample 3. { 4. MenuExample(){ 5. Frame f= **new** Frame("Menu and MenuItem Example"); 6. MenuBar mb=**new** MenuBar(); 7. Menu menu=**new** Menu("Menu"); 8. Menu submenu=**new** Menu("Sub Menu"); 9. MenuItem i1=**new** MenuItem("Item 1"); 10. MenuItem i2=**new** MenuItem("Item 2"); 11. MenuItem i3=**new** MenuItem("Item 3"); 12. MenuItem i4=**new** MenuItem("Item 4"); 13. MenuItem i5=**new** MenuItem("Item 5"); 14. menu.add(i1); 15. menu.add(i2); 16. menu.add(i3); 17. submenu.add(i4); 18. submenu.add(i5); 19. menu.add(submenu); 20. mb.add(menu); 21. f.setMenuBar(mb); 22. f.setSize(400,400); 23. f.setLayout(); 24. f.setVisible(**true**); 25. } 26. **public** **static** **void** main(String args[]) 27. { 28. **new** MenuExample(); 29. } 30. } | java awt menuitem and menu example 1 |