The Applet Class

Introduction

- An applet is a small program that is intended not to be run on its own, but rather to be embedded inside another application.
- The Applet class must be the superclass of any applet that is to be embedded in a Web page or viewed by the Java Applet Viewer.
- The Applet class provides a standard interface between applets and their environment.
- An applet is a program written in the Java programming language that can be included in an HTML page, much in the same way an image is included in a page.

- An Applet is a small Internet-based program written in Java, a programming language for the Web, which can be downloaded by any computer. The applet is also able to run in HTML. The applet is usually embedded in an HTML page on a Web site and can be executed from within a Browser.

- To create an applet, you must import the Applet class
  - This class is in the java.applet package
- The Applet class contains code that works with a browser to create a display window
- *Capitalization matters!*
  - applet and Applet are different names

- Here is the directive that you need:

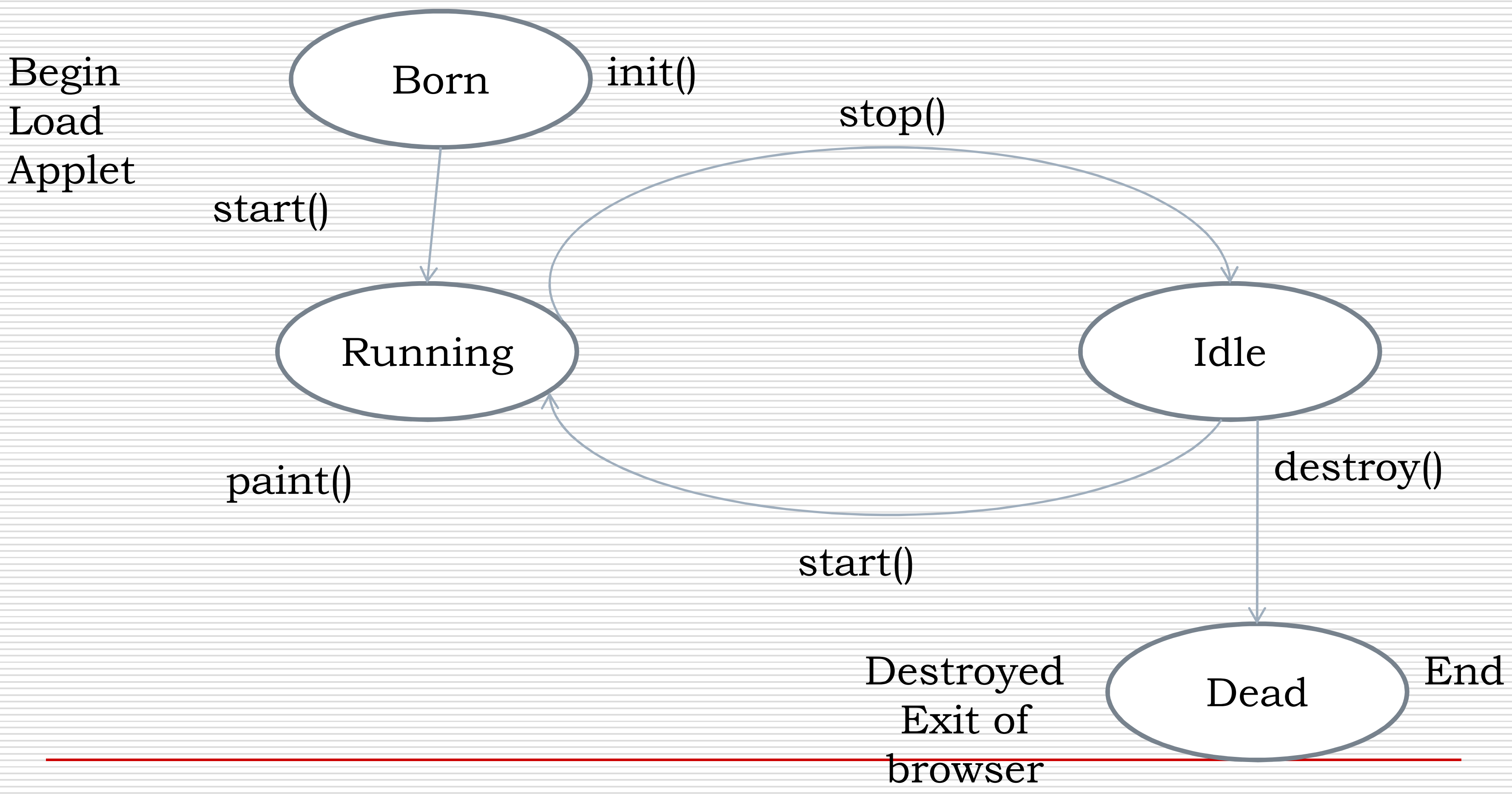
  import java.applet.Applet;
- import is a keyword
- java.applet is the name of the package
- A dot (.) separates the package from the class
- Applet is the name of the class
- There is a semicolon (;) at the end

The java.awt package

- "awt" stands for "Abstract Window Toolkit"
- The java.awt package includes classes for:
  - Drawing lines and shapes
  - Drawing letters
  - Setting colors
  - Choosing fonts
- If it's drawn on the screen, then java.awt is probably involved!

Applet Architecture

- applets are event driven.
- the user initiates interaction with an applet


Begin
Load
Applet
Born
init()
stop()
start()
Running
Idle
paint()
destroy()
start()
Destroyed
Exit of
browser
Dead
End

Applet Skeleton

- It is important to understand the order in which the various methods shown in the skeleton are called.
- When an applet begins, the AWT calls the following methods.
  - 1. **init( )**
  - 2. **start( )**
  - 3. **paint( )**
- When an applet is terminated, the following sequence of method calls takes place:
  - 1. **stop( )**
  - 2. **destroy( )**

init()

- The **init( )** method is the first method to be called. This is where you should initialize variables. This method is called only once during the run time of your applet.

start()

- The **start( )** method is called after **init( )**. It is also called to restart an applet after it has been stopped. Whereas **init( )** is called once—the first time an applet is loaded—**start( )** is called each time an applet's HTML document is displayed onscreen. So, if a user leaves a web page and comes back, the applet resumes execution at **start( )**.

paint()

- The **paint( )** method is called each time your applet's output must be redrawn. This situation can occur for several reasons. For example, the window in which the applet is running may be overwritten by another window and then uncovered. Or the applet window may be minimized and then restored.
- **paint( )** is also called when the applet begins execution. Whatever the cause, whenever the applet must redraw its output, **paint( )** is called. The **paint( )** method has one parameter of type **Graphics**.

Con’t

- This parameter will contain the graphics context, which describes the graphics environment in which the applet is running. This context is used whenever output to the applet is required.

stop()

- The **stop( )** method is called when a web browser leaves the HTML document containing the applet—when it goes to another page, for example.
- When **stop( )** is called, the applet is probably running. You should use **stop( )** to suspend threads that don't need to run when the applet is not visible. You can restart them when **start( )** is called if the user returns to the page.

destroy()

- The **destroy( )** method is called when the environment determines that your applet needs to be removed completely from memory.
- At this point, you should free up any resources the applet may be using (e.g. to kill any threads created in the init ()). The **stop( )** method is always called before **destroy()**.

update()

- This method is defined by the AWT and is called when your applet has requested that a portion of its window be redrawn.
- The problem is that the default version of update() first fills an applet with the default background colour and then calls paint(), this gives rise to a flash of default color (usually gray) each time update is called.

The HTML APPLET Tag

- An applet viewer will execute each APPLET tag that it finds in a separate window, while web browsers like Netscape Navigator, Internet Explorer, and HotJava will allow many applets on a single page.

```
< APPLET
[CODEBASE = codebaseURL]
CODE = appletFile
[ALT = alternateText]
[NAME = appletInstanceName]
WIDTH = pixels HEIGHT = pixels
[ALIGN = alignment]
[VSPACE = pixels] [HSPACE = pixels]
>
[< PARAM NAME = AttributeName VALUE =
     AttributeValue>]
[< PARAM NAME = AttributeName2 VALUE =
     AttributeValue>]
. . .
[HTML Displayed in the absence of Java]
</APPLET>
```

CODEBASE

- CODEBASE is an optional attribute that specifies the base URL of the applet code, which is the directory that will be searched for the applet’s executable class file (specified by the CODE tag). The HTML document’s URL directory is used as the CODEBASE if this attribute is not specified. The CODEBASE does not have to be on the host from which the HTML document was read.

CODE

- CODE is a required attribute that gives the name of the file containing your applet’s compiled **.class file. This file is relative to the code base URL of the applet,** which is the directory that the HTML file was in or the directory indicated by CODEBASE if set.

ALT

- The ALT tag is an optional attribute used to specify a short text message that should be displayed if the browser understands the APPLET tag but can’t currently run Java applets. This is distinct from the alternate HTML you provide for browsers that don’t support applets.

NAME

- NAME is an optional attribute used to specify a name for the applet instance. Applets must be named in order for other applets on the same page to find them by name and communicate with them. To obtain an applet by name, use **getApplet( ),** which is defined by the **AppletContext interface.**

WIDTH AND HEIGHT

- WIDTH and HEIGHT are required attributes that give the size (in pixels) of the applet display area.

ALIGN

- ALIGN is an optional attribute that specifies the alignment of the applet. This attribute is treated the same as the HTML IMG tag with these possible values: LEFT, RIGHT, TOP, BOTTOM, MIDDLE, BASELINE, TEXTTOP, ABSMIDDLE, and ABSBOTTOM.

VSPACE AND HSPACE

- These attributes are optional. *VSPACE specifies the space*, in pixels, above and below the applet. HSPACE specifies the space, in pixels, on each side of the applet. They're treated the same as the IMG tag's VSPACE and HSPACE attributes

PARAM NAME AND VALUE

- The PARAM tag allows you to specify appletspecific arguments in an HTML page. Applets access their attributes with the **getParameter( ) method.**

- **getDocumentBase( )**
  - Gets the base URL.
- **getCodeBase( )**
  - Gets the URL of the document in which the applet is embedded.

Applet Life Cycle

- When Applet is loaded, it goes to series of changes in states. The Applet state include:
  - Born or Initialization State
  - Running State
  - Idle State
  - Dead or Destroy State

Handling Events

- Applets inherit a group of event-handling methods from the *Container* class.
- The *Container* class defines several methods, such as *processKeyEvent* and *processMouseEvent*, for handling particular types of events, and then one catch-all method called *processEvent.*
- To react to an event, an applet must override the appropriate event-specific method

Methods for UI Components

- Add
  - Adds the specified Component
- Remove
  - Removes the specified Component
- setLayout
  - Sets the layout manager

MouseListener

- Mouse events notify when the user uses the mouse to interact with a component.
- Mouse events occur when the cursor enters or exits a component's onscreen area and when the user presses or releases one of the mouse buttons.

The MouseListener Interface

Method	Purpose
mouseClicked(MouseEvent)	Called just after the user clicks
mouseEntered(MouseEvent)	Called just after the cursor enters the bounds
mouseExited(MouseEvent)	Called just after the cursor exits the bounds
mousePressed(MouseEvent)	Called just after the user presses a mouse button
mouseReleased(MouseEvent)	Called just after the user releases a mouse button