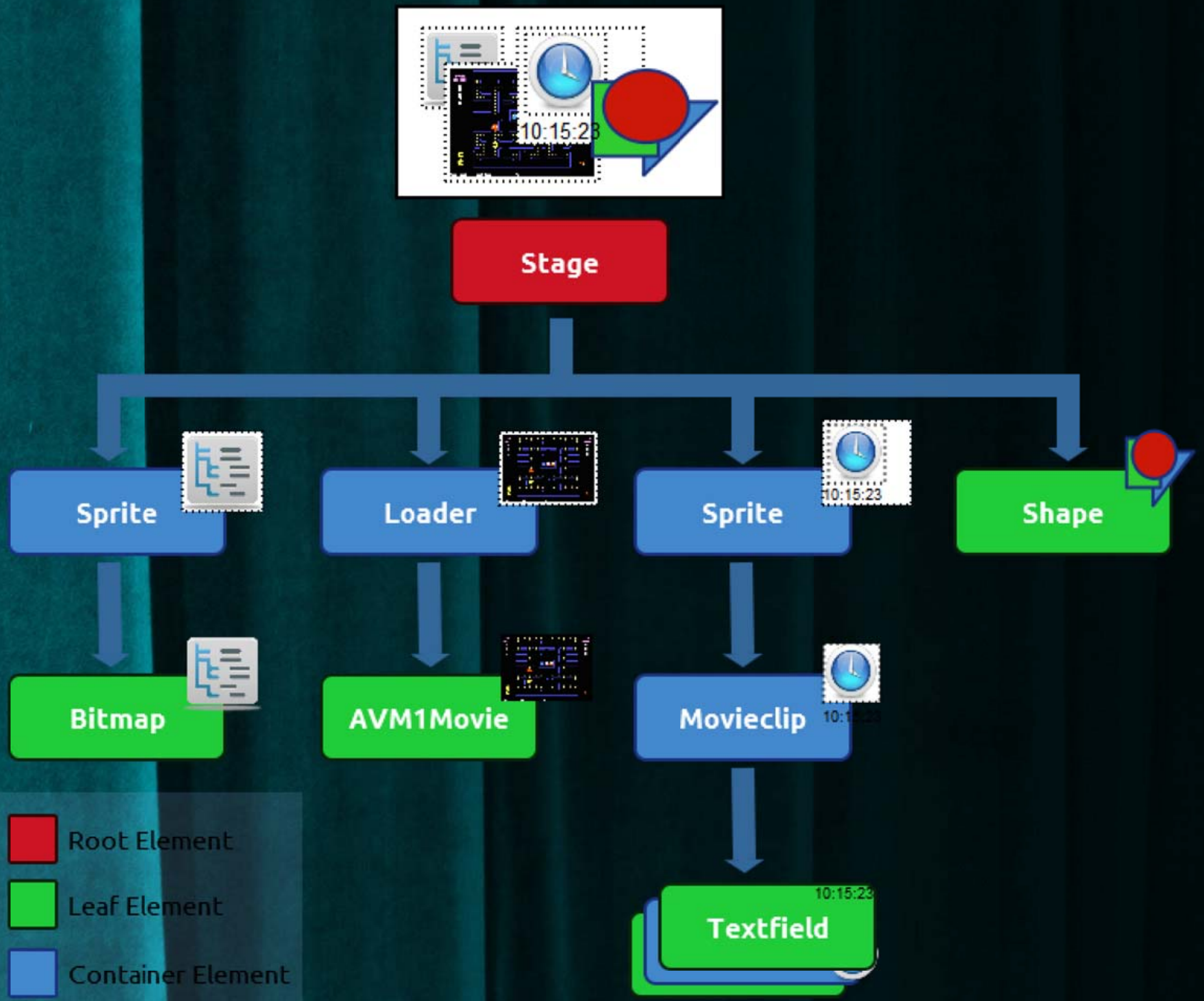
Actionscript 3.0 Basics

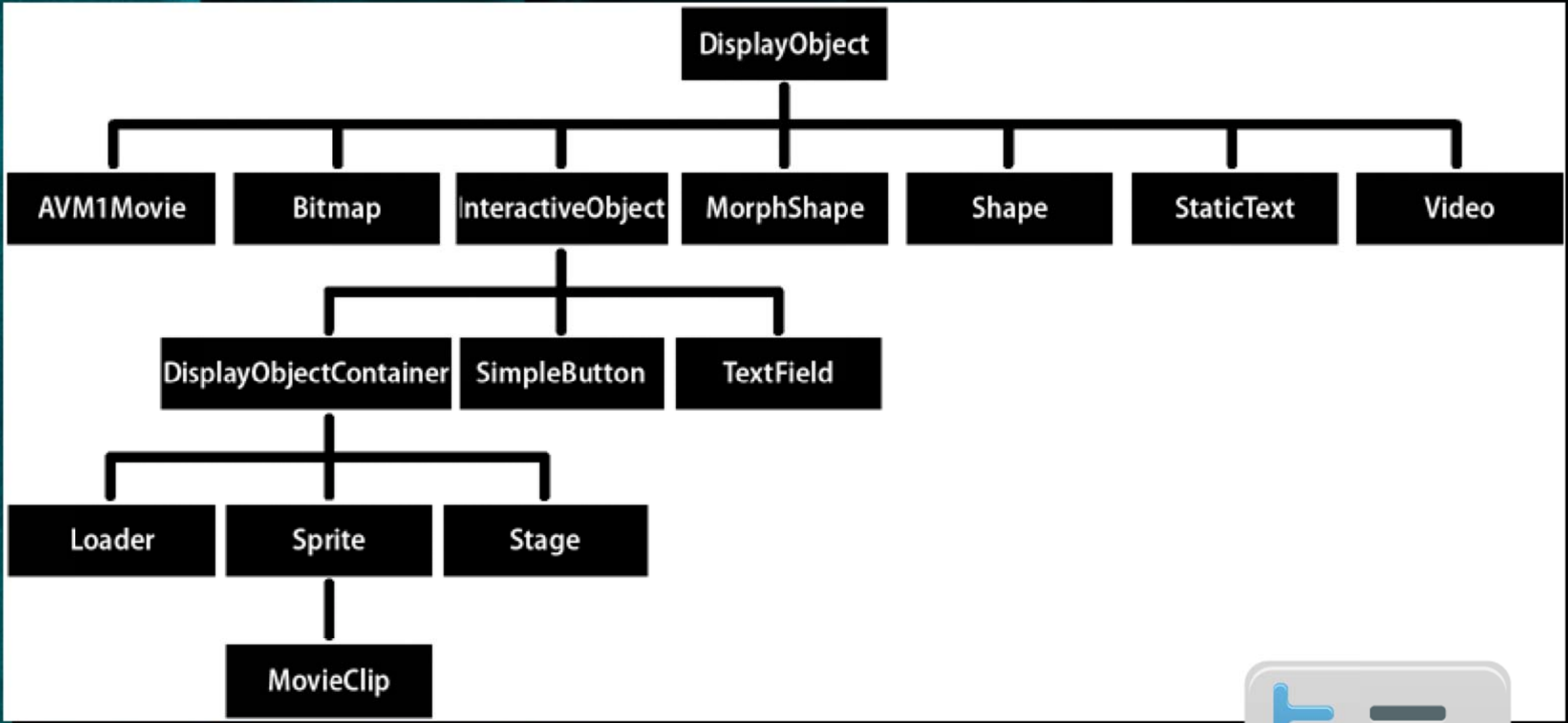
3 – Display List Api

Flash DOM



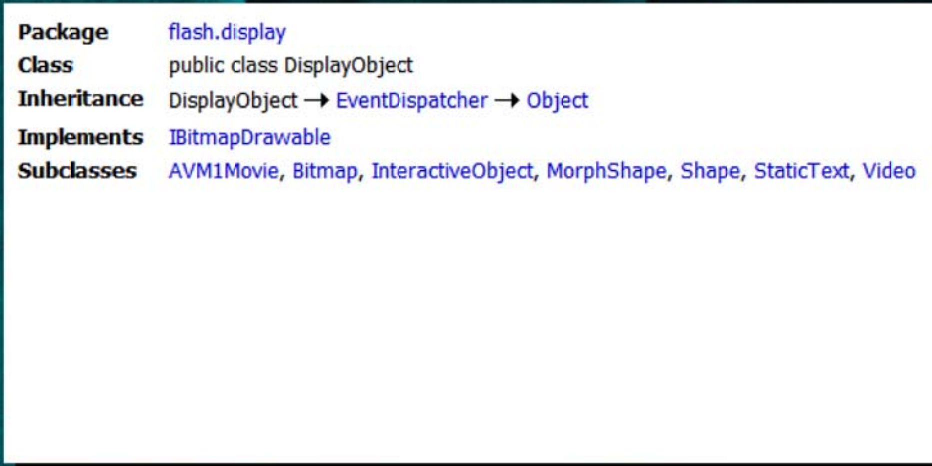
🟥 Root Element
🟩 Leaf Element
🟦 Container Element

Display Objects

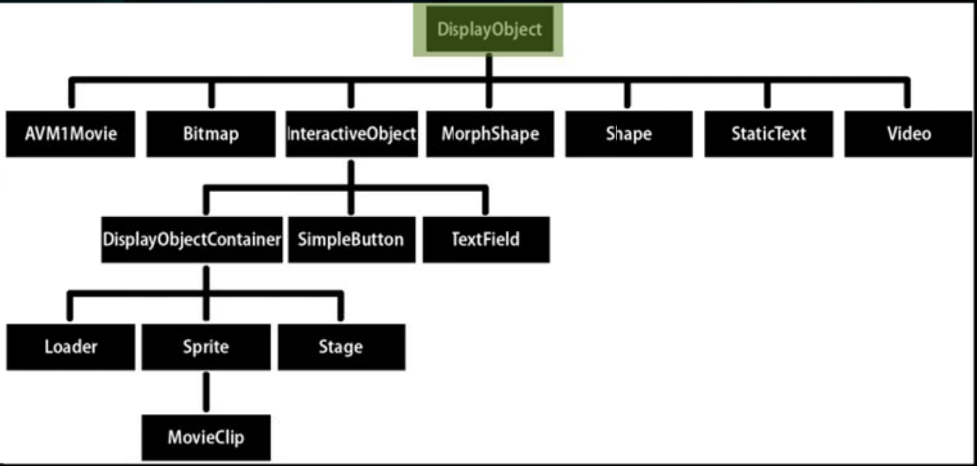


DisplayObject

- Is the base class for all objects that can be placed on the display list.
- The class itself does not include any APIs for rendering content onscreen.
- It is an abstract class, can't be instantiated and is intended to be sub-classed.
- Sub-classes of this class will implement rendering mechanisms.
- Provides the sizing, positioning, styling, etc. APIs that affect all display objects in flash.

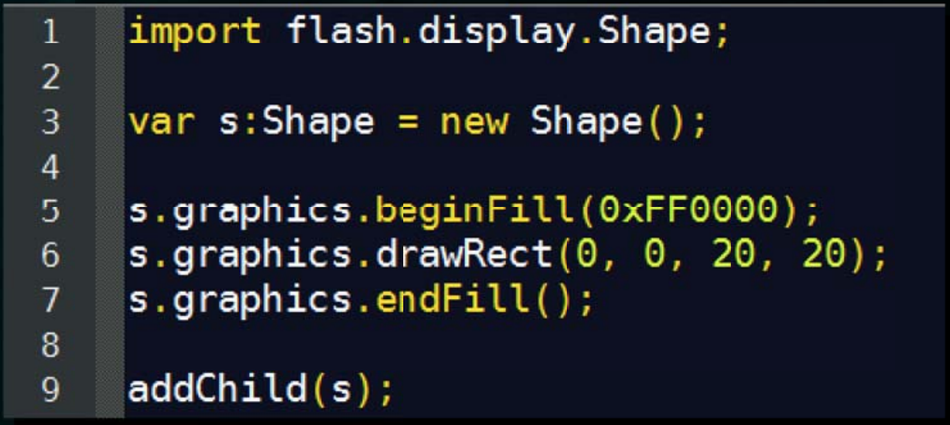


Package	flash.display
Class	public class DisplayObject
Inheritance	DisplayObject → EventDispatcher → Object
Implements	IBitmapDrawable
Subclasses	AVM1Movie, Bitmap, InteractiveObject, MorphShape, Shape, StaticText, Video

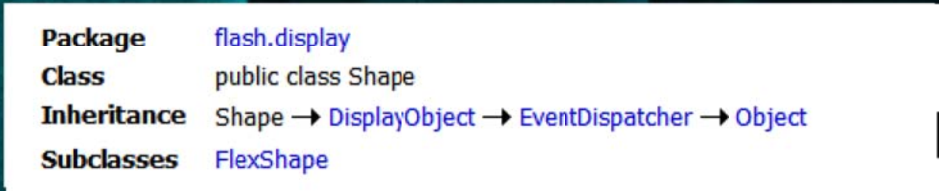


Shape

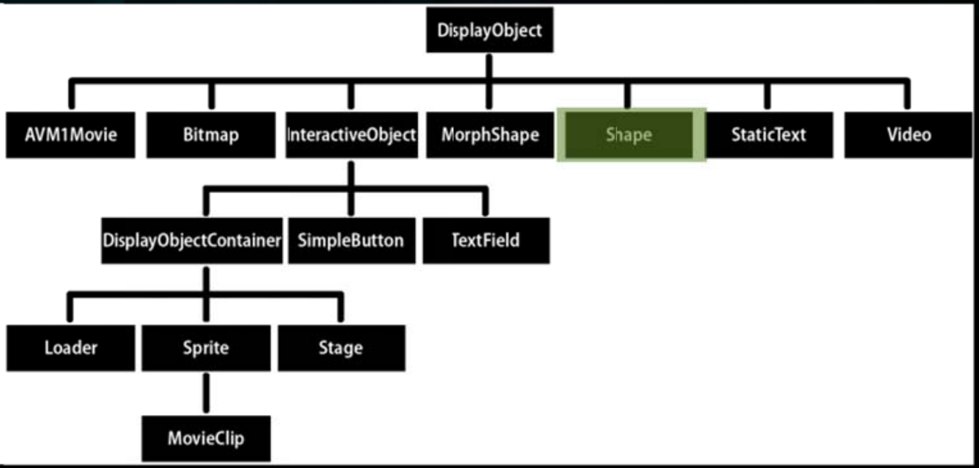
This class is used to create lightweight shapes using the ActionScript drawing api.
Shape objects consume less memory than Sprite objects.
A Sprite object supports user input events, while a Shape object does not.



```
import flash.display.Shape;

var s:Shape = new Shape();

s.graphics.beginFill(0xFF0000);
s.graphics.drawRect(0, 0, 20, 20);
s.graphics.endFill();

addChild(s);
```

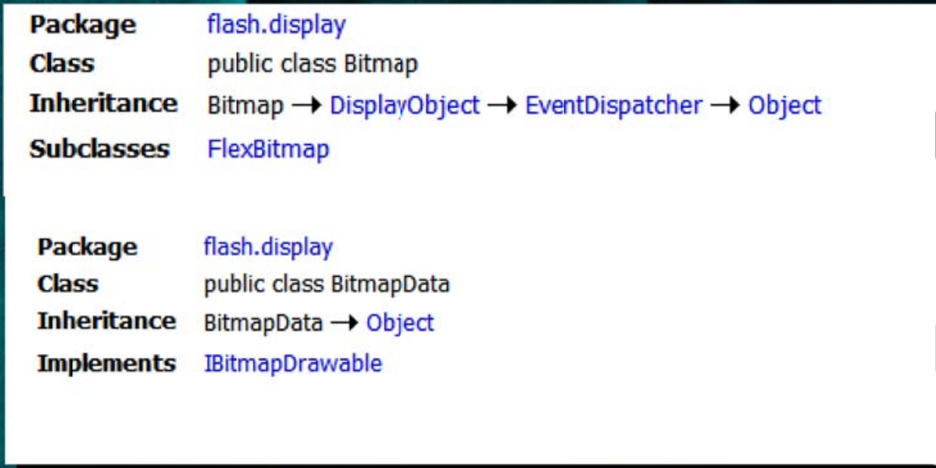


Package	flash.display
Class	public class Shape
Inheritance	Shape → DisplayObject → EventDispatcher → Object
Subclasses	FlexShape



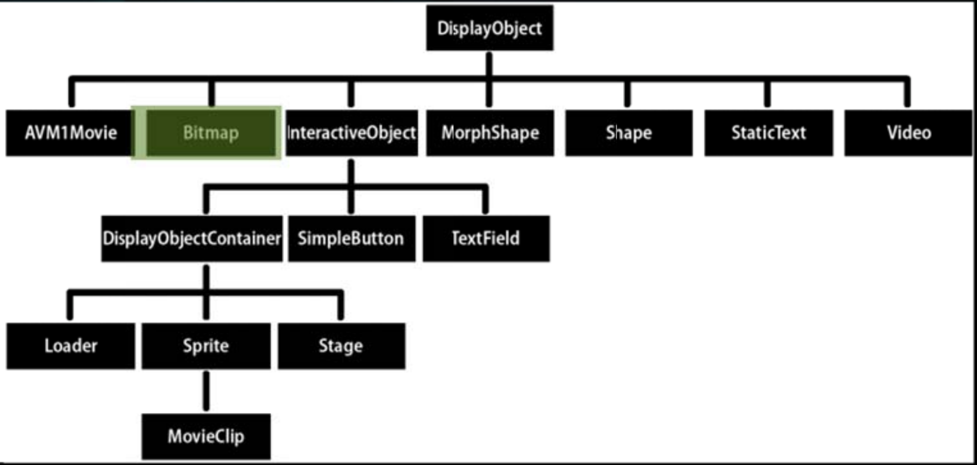
Bitmap (+ BitmapData)

The Bitmap class represents display objects that represent bitmap images.
- A Bitmap object can share its BitmapData reference among several Bitmap objects, you can create multiple Bitmap objects that reference the same BitmapData object avoiding the memory overhead of a BitmapData object for each display object instance.
- Like Shape, it won't dispatch user iteraction events.

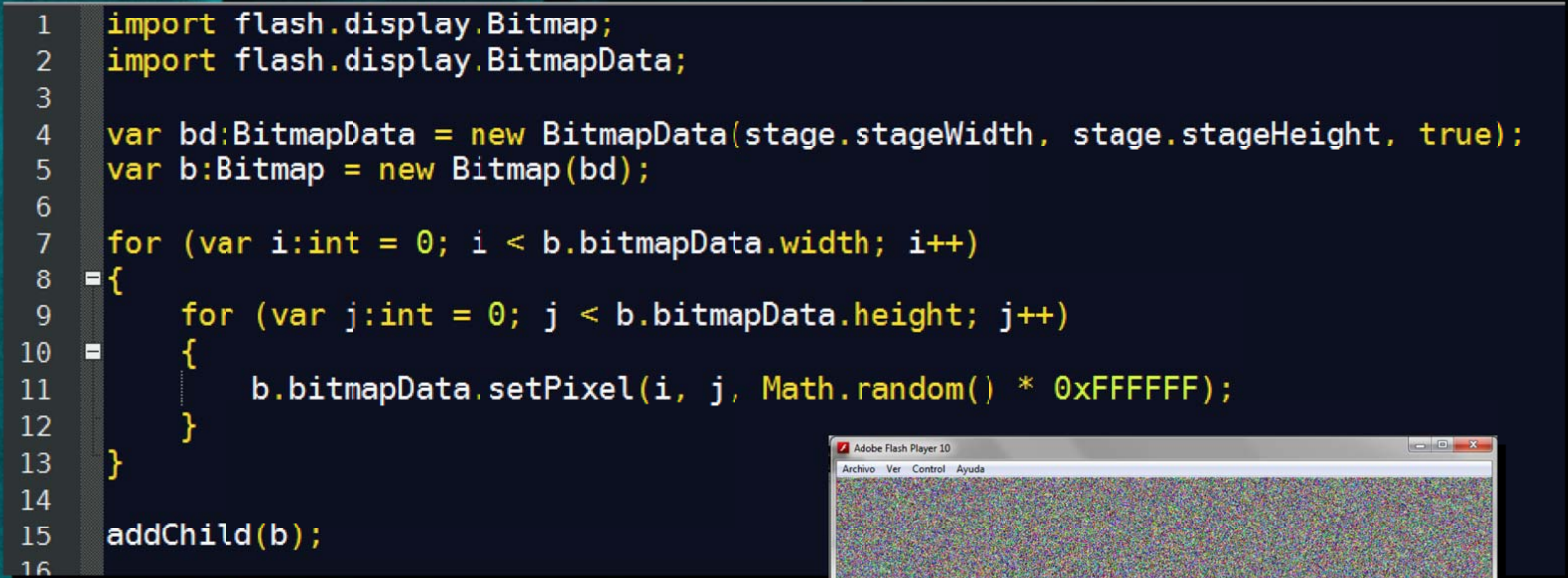


Package	flash.display
Class	public class Bitmap
Inheritance	Bitmap → DisplayObject → EventDispatcher → Object
Subclasses	FlexBitmap

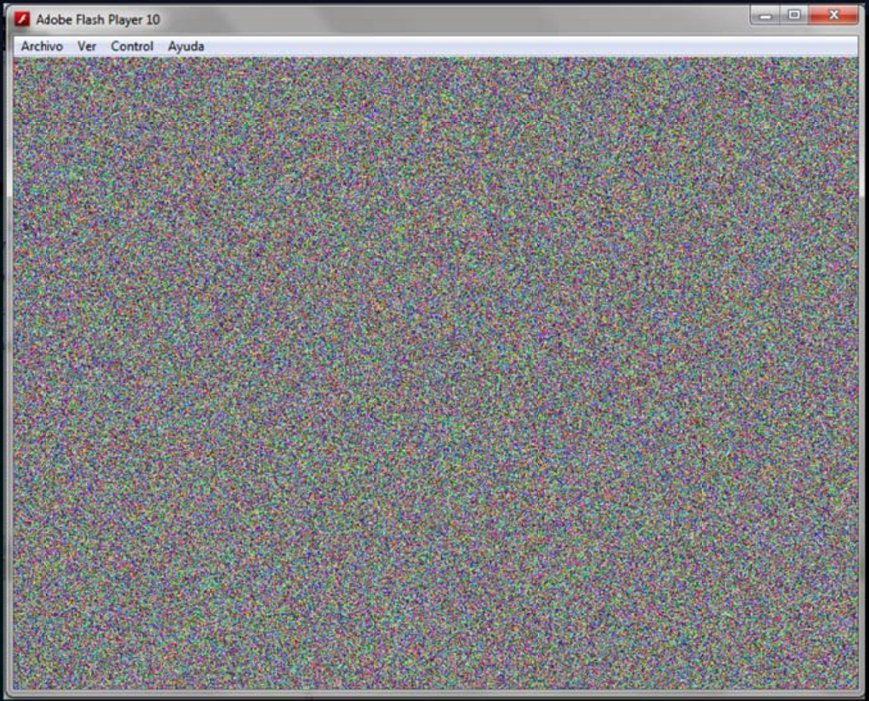
Package	flash.display
Class	public class BitmapData
Inheritance	BitmapData → Object
Implements	IBitmapDrawable



Bitmap (+ BitmapData)

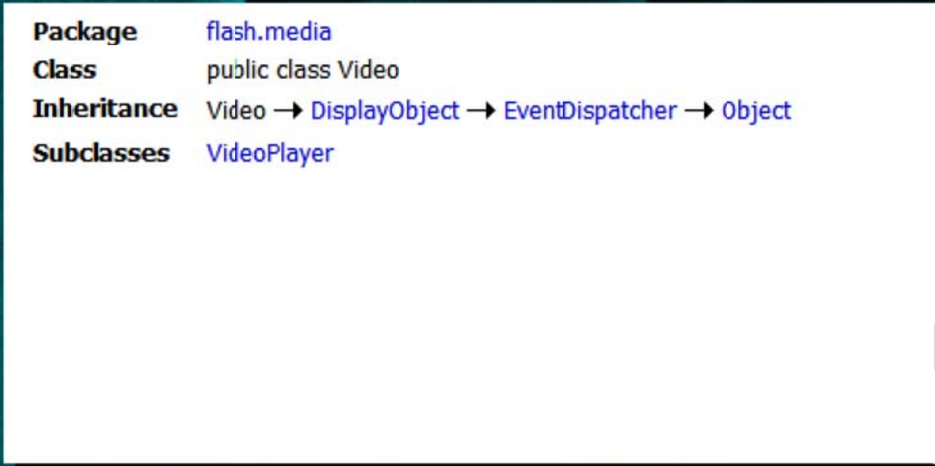


```
import flash.display.Bitmap;
import flash.display.BitmapData;

var bd:BitmapData = new BitmapData(stage.stageWidth, stage.stageHeight, true);
var b:Bitmap = new Bitmap(bd);

for (var i:int = 0; i < b.bitmapData.width; i++)
{
    for (var j:int = 0; j < b.bitmapData.height; j++)
    {
        b.bitmapData.setPixel(i, j, Math.random() * 0xFFFFFF);
    }
}

addChild(b);
```

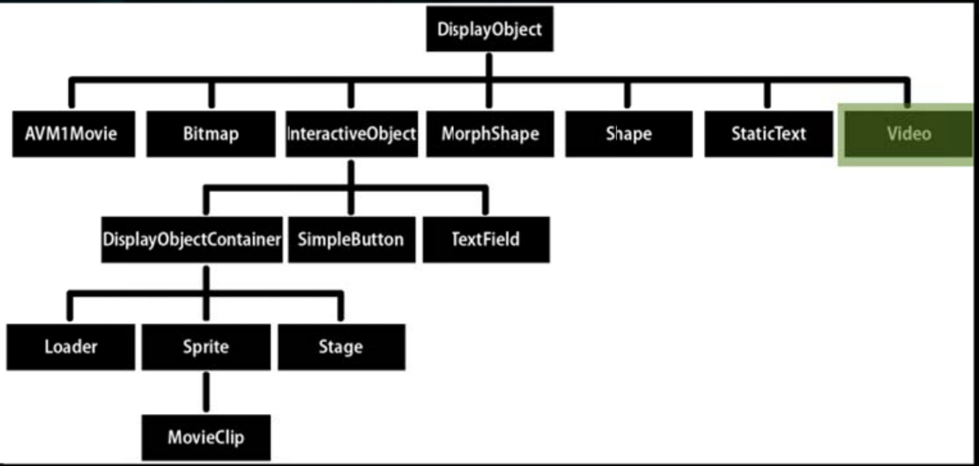


Video

The Video class displays live or recorded video in an application without embedding the video in your SWF file. This class creates a Video object that plays either of the following kinds of video: recorded video files stored on a server or locally, or live video captured by the user. A Video object is a display object on the application's display list and represents the visual space in which the video runs in a user interface.

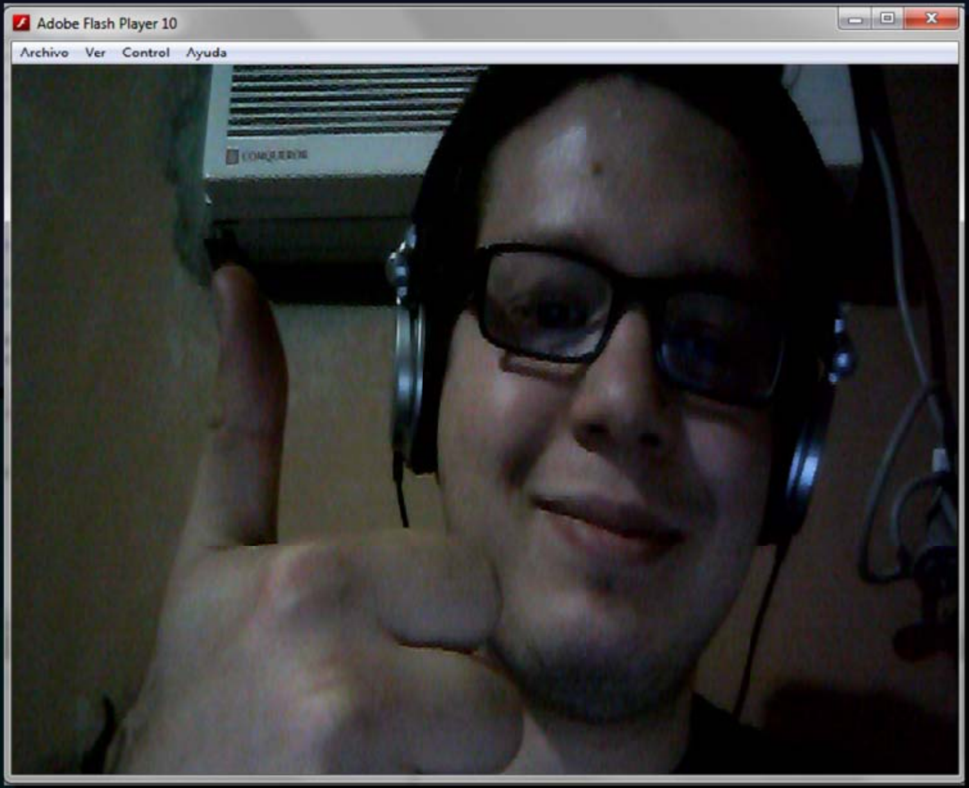


Package	flash.media
Class	public class Video
Inheritance	Video → DisplayObject → EventDispatcher → Object
Subclasses	VideoPlayer



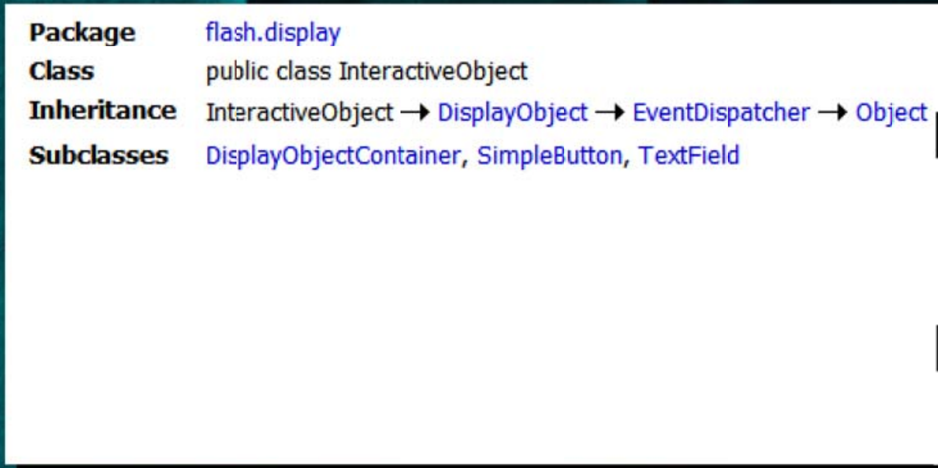
Video

```
import flash.media.Camera;
import flash.media.Video;

var v:Video = new Video(stage.stageWidth, stage.stageWidth);
var c:Camera = Camera.getCamera();
if (c) {
    c.setMode(stage.stageWidth, stage.stageHeight, 30);
    v.attachCamera(c);
}
else {
    trace("No camera!");
}

addChild(v);
```

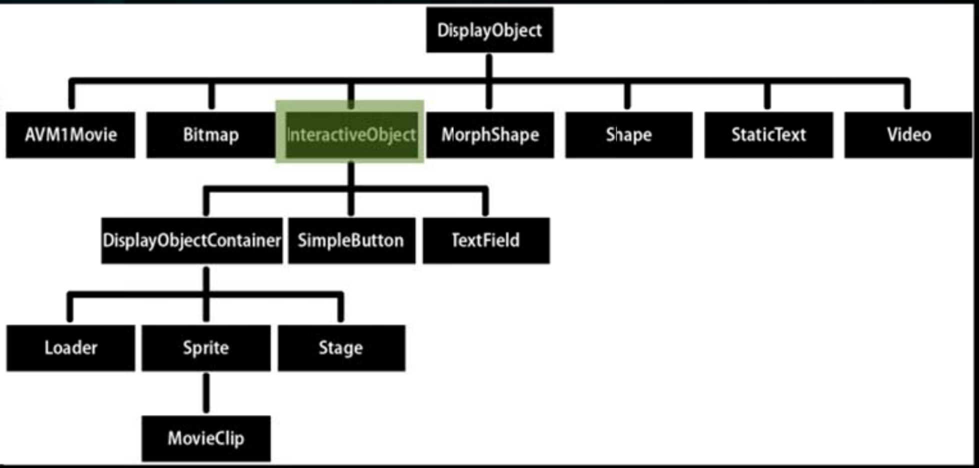


InteractiveObject

The InteractiveObject class is the abstract base class for all display objects with which the user can interact, using the mouse, keyboard, or other user input device.



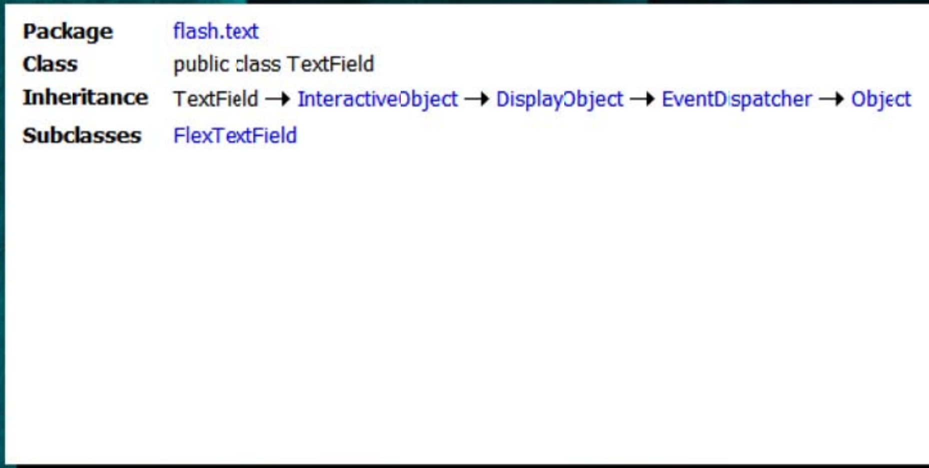
Package	flash.display
Class	public class InteractiveObject
Inheritance	InteractiveObject → DisplayObject → EventDispatcher → Object
Subclasses	DisplayObjectContainer, SimpleButton, TextField



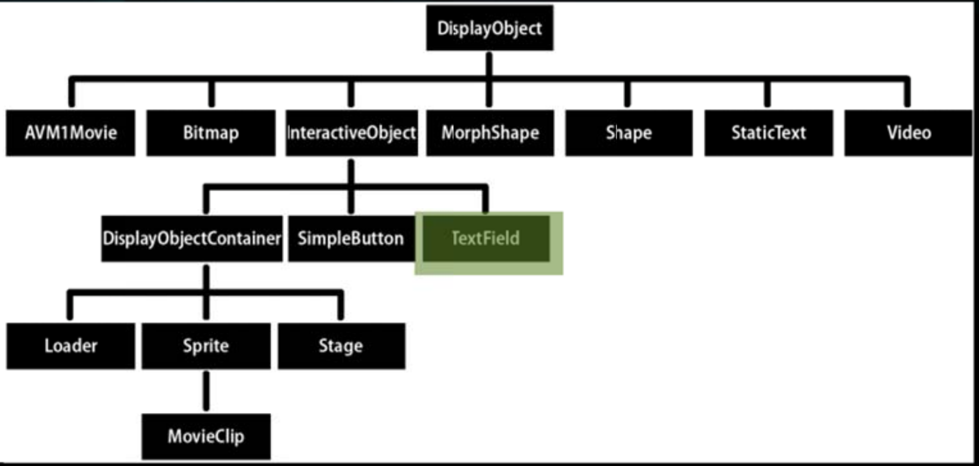
TextField

The TextField class is used to create display objects for text display and input.

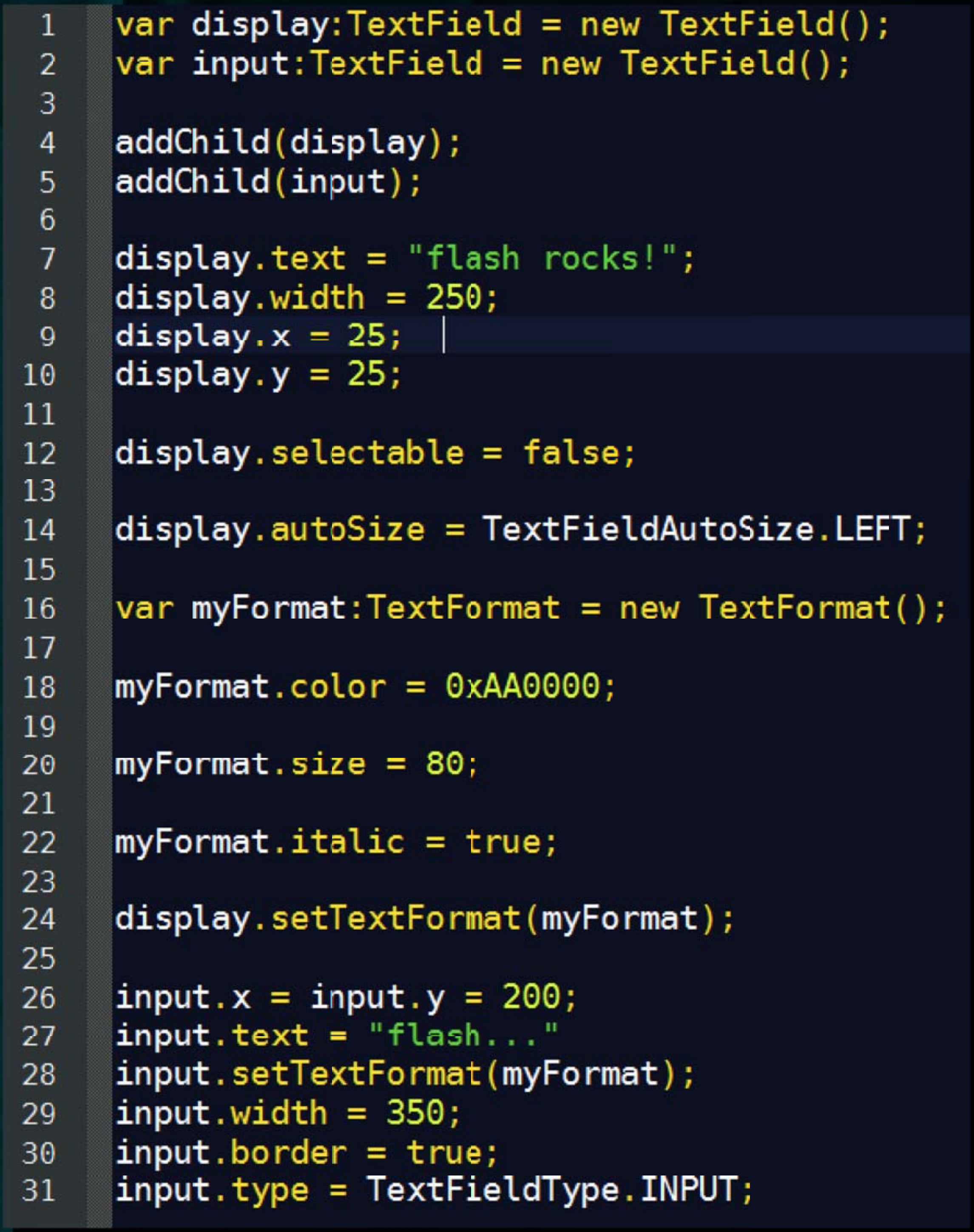
- The methods of the TextField class let you set, select, and manipulate text in a dynamic or input text field that you create during authoring or at runtime.
- ActionScript provides several ways to format your text at runtime. The TextFormat class lets you set character and paragraph formatting for TextField objects. You can apply Cascading Style Sheets (CSS) styles and use some html formatting tags.



Package	flash.text
Class	public class TextField
Inheritance	TextField → InteractiveObject → DisplayObject → EventDispatcher → Object
Subclasses	FlexTextField



TextField

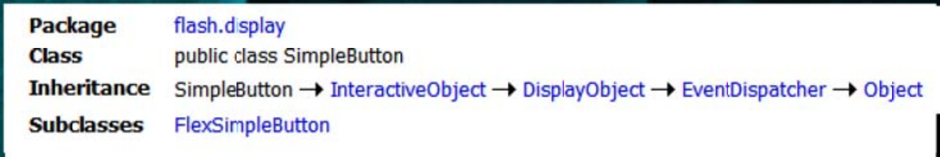


```
var display:TextField = new TextField();
var input:TextField = new TextField();

addChild(display);
addChild(input);

display.text = "flash rocks!";
display.width = 250;
display.x = 25;
display.y = 25;

display.selectable = false;

display.autoSize = TextFieldAutoSize.LEFT;

var myFormat:TextFormat = new TextFormat();

myFormat.color = 0xAA0000;

myFormat.size = 80;

myFormat.italic = true;

display.setTextFormat(myFormat);

input.x = input.y = 200;
input.text = "flash..."
input.setTextFormat(myFormat);
input.width = 350;
input.border = true;
input.type = TextFieldType.INPUT;
```

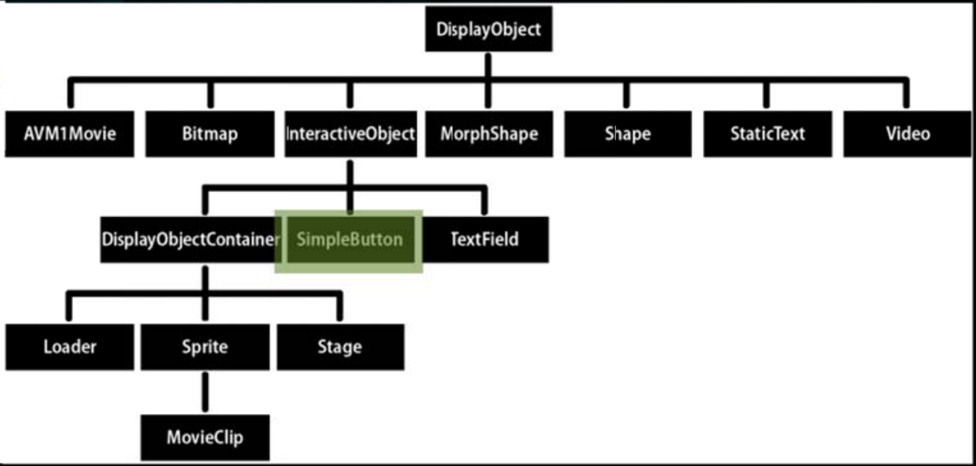
SimpleButton

The SimpleButton class lets you control all instances of button symbols in a SWF file.

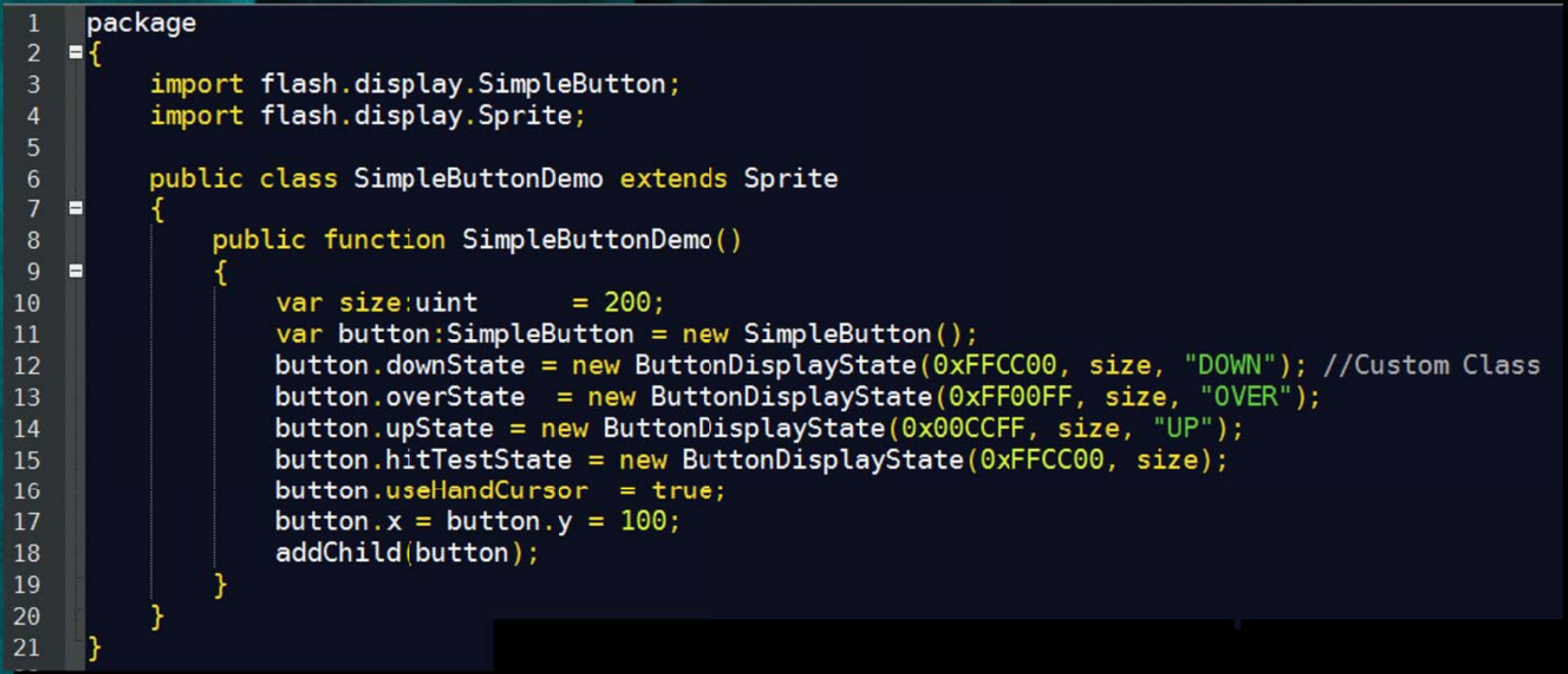
- Button symbols are ui elements the user can iteract with.
- This class makes easy to create buttons in flash.



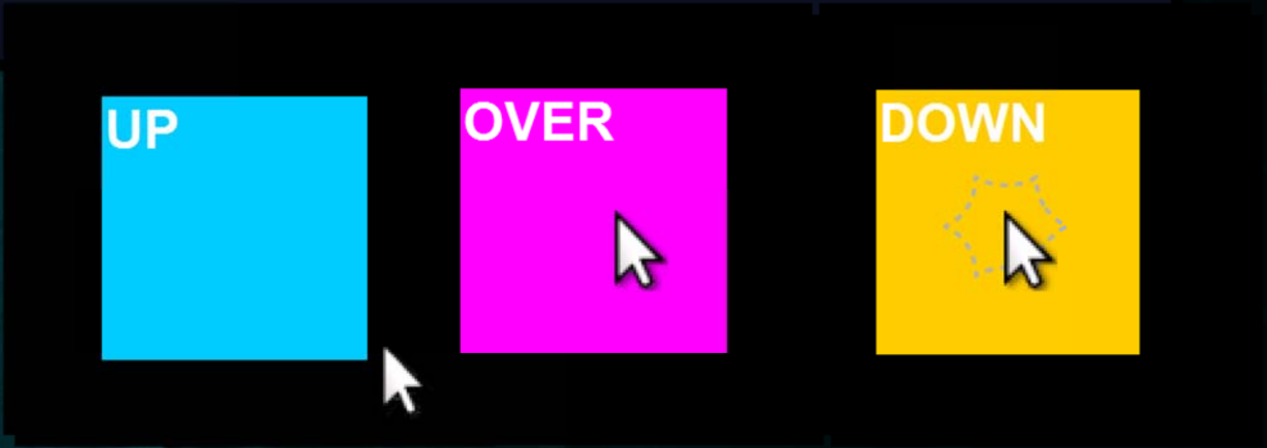
Package	flash.display
Class	public class SimpleButton
Inheritance	SimpleButton → InteractiveObject → DisplayObject → EventDispatcher → Object
Subclasses	FlexSimpleButton



SimpleButton



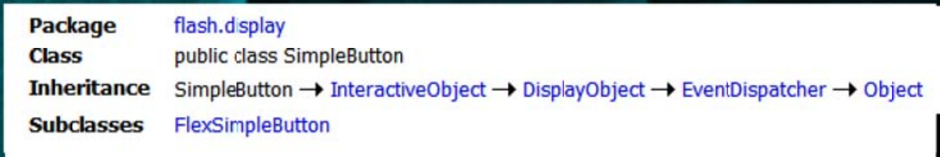
```
package
{
    import flash.display.SimpleButton;
    import flash.display.Sprite;

    public class SimpleButtonDemo extends Sprite
    {
        public function SimpleButtonDemo()
        {
            var size:uint      = 200;
            var button:SimpleButton = new SimpleButton();
            button.downState = new ButtonDisplayState(0xFFCC00, size, "DOWN"); //Custom Class
            button.overState  = new ButtonDisplayState(0xFF00FF, size, "OVER");
            button.upState = new ButtonDisplayState(0x00CCFF, size, "UP");
            button.hitTestState = new ButtonDisplayState(0xFFCC00, size);
            button.useHandCursor  = true;
            button.x = button.y = 100;
            addChild(button);
        }
    }
}
```



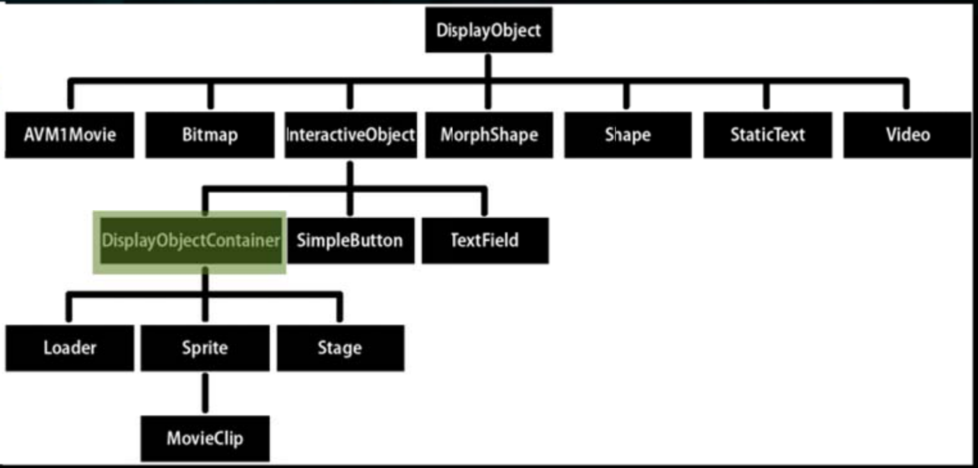
DisplayObjectContainer

The DisplayObjectContainer class is the base class for all objects that can serve as display object containers on the display list.

- Use the DisplayObjectContainer class to arrange the display objects in the display list.
- Each DisplayObjectContainer object has its own child list for organizing the z-order of the objects.

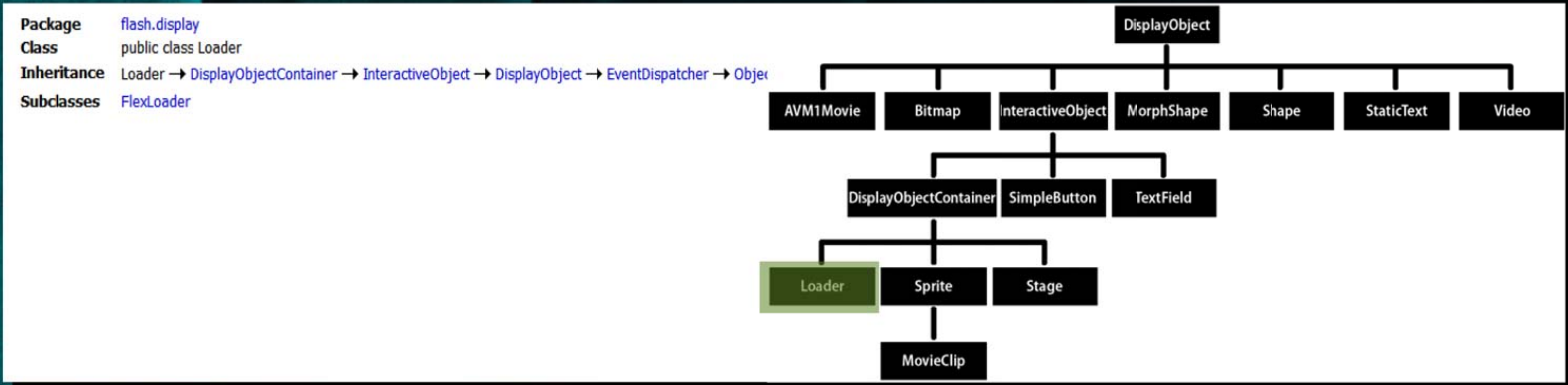


Package	flash.display
Class	public class SimpleButton
Inheritance	SimpleButton → InteractiveObject → DisplayObject → EventDispatcher → Object
Subclasses	FlexSimpleButton



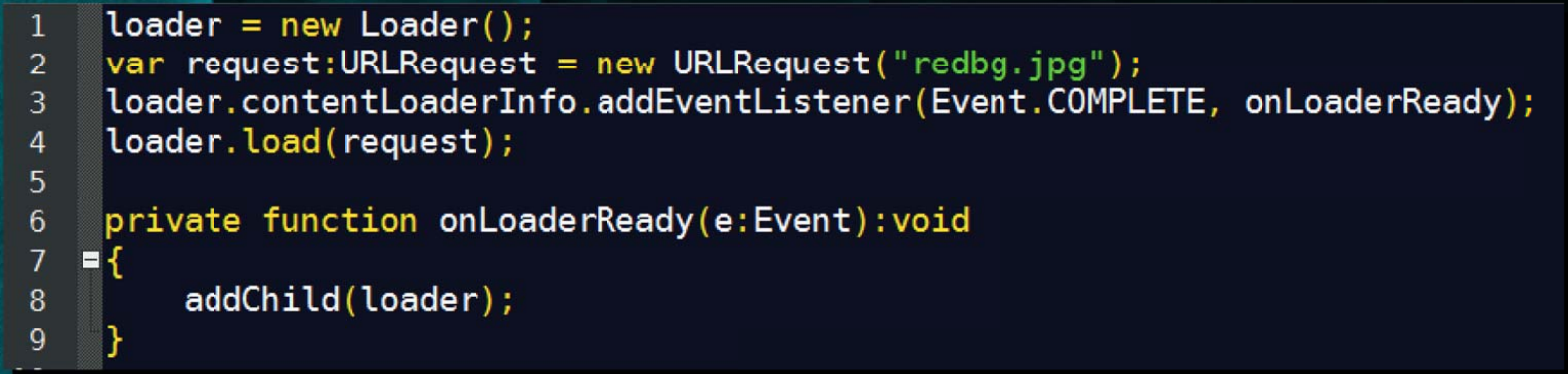
Loader

The Loader class is used to load SWF files or image (JPG, PNG, or GIF) files. Use the load() method to initiate loading. The loaded display object is added as a child of the Loader object.



Package	flash.display
Class	public class Loader
Inheritance	Loader → DisplayObjectContainer → InteractiveObject → DisplayObject → EventDispatcher → Objec
Subclasses	FlexLoader

Loader



```
loader = new Loader();
var request:URLRequest = new URLRequest("redbg.jpg");
loader.contentLoaderInfo.addEventListener(Event.COMPLETE, onLoaderReady);
loader.load(request);

private function onLoaderReady(e:Event):void
{
    addChild(loader);
}
```

Thanks!

github.com/matix/as3basics

matias.figueroa@globant.com
@matixfigueroa