

## Java History

- Created by James Gosling et al. at Sun Microsystems in 1991 "The Green Team"



- Gosling created a processor-independent language for "StarSeven", a 2-way wireless remote-control device
  - Called the language 'Oak'



## Java History (cont'd)

- Could not find a market for the technology.
- During a Sun offsite with Bill Joy in 1994:
  - "why not use it on the Internet?"
  - Started development of "WebRunner", later to be renamed "HotJava" browser
    - a browser capable of downloading and running Java bytecode.
- Folks were impressed with interactive Web pages
  - e.g., spinning molecules, sorting demos
  - many downloads of HotJava
    - a big success
- Marc Andersen of Netscape agrees to support Java in Netscape browsers in 1995

2

## Applet Execution

- A JVM (Java Virtual Machine) running within the context of the client browser loads and runs the applet classes.
- Applet is not trusted.
  - Limited access to system resources (file system, network)

3

## How Java Applets Work

- `<embed>`, `<applet>`, or `<object>`
- Java "jar" files downloaded to client machine
  - contains
    - class files
    - other resources (images, sounds, property files, ...)
- Class indicated in the tag (must derive from `java.applet.Applet`) is invoked

4

## A Trivial Applet

- Old (and convenient way) of invoking applets (invokes built-in jre)
  - [TrivialApplet.html](#)
- Official (new way) of invoking applets (invokes SUN's jre1.5 plug-in)
  - [TrivialObject.html](#)

5

## TrivialApplet.html

```
<html>
<head>
<title>Applet Example</title>
</head>

<body>
<h1>Below the line is an applet</h1>
<hr/>
<applet code="Trivial.class" width="300" height="100"/>
</body>
</html>
```

6

## TrivialObject.html

```
<html>
<head>
<title>Applet Example IE Version</title>
</head>
<body>
<h1>Below the line is an applet</h1>
<hr/>
<object classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
codebase="http://java.sun.com/update/1.5.0/jinstall-1_5_0-windows-i586.cab"
height="100" width="300" >
<param name="code" value="Trivial.class" />
<param name="type" value="application/x-java-applet;version=1.5"/>
<strong>
This browser does not have a Java Plug-in. <br />
<a href="http://java.sun.com/products/plugin/downloads/index.html">
Get the latest Java Plug-in here.</a>
</strong>
</object>
</body>
</html>
```

7

## TrivialObject.html

```
<html>
<head>
<title>Applet Example Firefox Version</title>
</head>
<body>
<h1>Below the line is an applet</h1>
<hr/>
<object classid="java:Trivial.class"
type="application/x-java-applet"
height="100" width="300" />
</body>
</html>
```

8

## Trivial.java

```
import java.applet.Applet;
import java.awt.Graphics;
import java.awt.Font;

public class Trivial extends Applet {
    public void paint(Graphics g) {
        g.setFont( new Font("Default", Font.PLAIN, 28) );
        g.drawString("Hello, world!", 50, 25);
    }
}
```

9

## java.applet.Applet lifecycle methods

- void init()
  - invoked when applet is initially loaded
- void start()
  - invoked when started or restarted as a result of user flipping Web pages
- void stop()
  - invoked when user switches away from web page
- void destroy()
  - invoked (usually) on browser termination

10

## Applet contextual methods

- String getParameter(String)
- AudioClip getAudioClip(URL)
- Image getImage(URL)
- URL getCodeBase()
- URL getDocumentBase()
- void showStatus(string)

11

## Applet GUI

- Inherit from Applet and use the AWT GUI library
- Inherit from JApplet and use the JFC/Swing GUI
  - [GUI.html](#)

12

## GUI.html

```
<html>
<head>
<title>Applet Example</title>
</head>

<body>
<h1>Between the lines is an applet</h1>
<hr>
<applet code="GUI.class" width="192" height="251">
  <param name="image" value="grizzlypenny.jpg">
  <param name="sound" value="bearsound.au">
</applet>
<hr>
</body>
</html>
```

13

## Java/JavaScript Communications

- JavaScript can call methods defined in Java Applets
  - [JS.html](#)

14

## JS.html

```
<html>
<head>
<title>Test Applet</title>
</head>
<body>
<h1>This is a test of applets</h1>
<hr></hr>
<applet name="jsapplet" code="JS.class" height="300" width="300">
  <param name="text" value="Hello World!"></param>
  Text displayed by non-java enabled browsers
</applet>
<hr></hr>
<form>
  <input type="button"
  onclick="alert(document.jsapplet.getText())"
  value="Get Data From Applet">
</form>
</body>
</html>
```

15

## JS.java

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

public class JS extends Applet {
  String text = "error";

  public void Init() {
    text = getParameter("text");
  }

  public void paint(Graphics g) {
    int style = Font.BOLD + Font.ITALIC;
    g.setFont(new Font("TimesRoman", style, 36));
    g.drawString(text, 50, 50);
  }

  public String getText() {
    return text;
  }
}
```

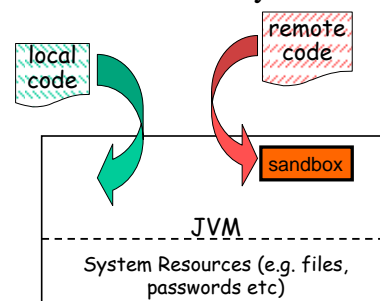
16

## Java Security

- The original security model provided by the Java platform is known as the *sandbox model*
  - Very restricted environment in which to run untrusted code obtained from the Internet
- Local code is trusted
  - Full access to vital system resources (such as the file system)
- Downloaded remote code (an applet) is not trusted
  - Access only the limited resources provided inside the sandbox

17

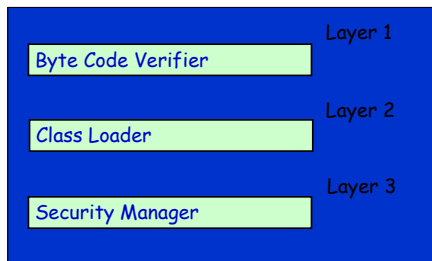
## JDK 1.0 Security Model



Rely on Type System for Security

18

## Java Security



19

## Security

- Byte Code Verifier
  - Static checks are made against the Java bytecode to ensure it attempts to do nothing improper
  - Like forge a function pointer
    - Treat an integer as a memory address
- Class Loader
  - Trusted part of the JVM
  - Defines a local name space that can be used to ensure that an untrusted applet cannot interfere with the running of other programs
- Security Manager
  - Trusted part of the JVM
  - As the applet runs, whenever it needs access to a system resource the SecurityManager inspects call stack to determine if call came from Applet or trusted local code.

20

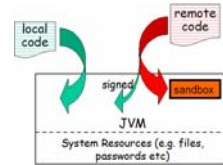
## Security Holes

- Applets are supposed to be able to talk only to the server that they originated from.
  - March 1996, Steve Gibbons and Dean et al., independently discovered holes in the implementation that allows applets to make connections to any host on the Internet
  - April 2000, IE 5.0 for Macintosh. Bug allows applets to open nw connection to any host on the Internet.
- On March 5, 1997, an internal security audit at JavaSoft revealed a bug in the Java bytecode verifier.
  - In theory, this bug could be exploited to bypass the Java Security Manager and execute forbidden operations
- Nov 2004, bug in Java-to-JavaScript data exchange allows JavaScript to load unsafe class. As a result, a remote attacker could execute hostile applets to access, download, upload or execute arbitrary files.

21

## JDK 1.1 Security Model

- Introduced the concept of a *signed applet*
  - Signed applets, together with their signatures, are delivered in the JAR (Java Archive) format
  - The signature tells who the applet comes from, and that the applet has not been tampered with.
  - Doesn't tell you anything about quality of the applet
- Signed applet is treated as trusted local code if the signature key is trusted
- Unsigned applets run in sandbox



22

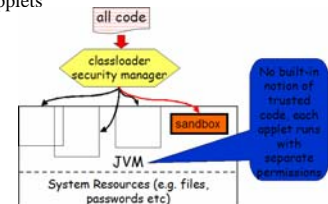
## Signed Applets

- The applet should generate a Java Security dialog, indicating that a java applet from "the sender's signature id" is requesting additional privileges, e.g. reading, modification, or deletion of any of your files. Granting the privilege is noted as high risk
- The Java Security dialog should indicate "Identity verified by *original issuer of certificate*" and display a button for you to examine the certificate

23

## JDK 1.2 Security Model

- Supports easily configurable security policy
- No longer a built-in concept that all local code is trusted
- Local code (e.g., non-system code, application packages installed on the local file system) is subject to the same security control as applets



24