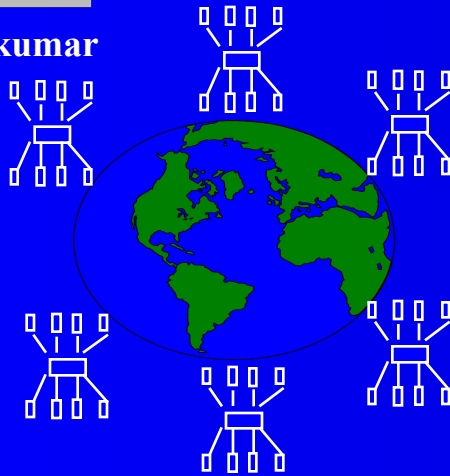




Internet and Java

Foundations, Programming and Practice

(c) Rajkumar



Rajkumar Buyya

School of Computer Science and Software Engineering

Monash University

Melbourne, Australia

Email: rajkumar@dgs.monash.edu.au

URL: <http://www.dgs.monash.edu.au/~rajkumar>



Agenda

(c) Rajkumar

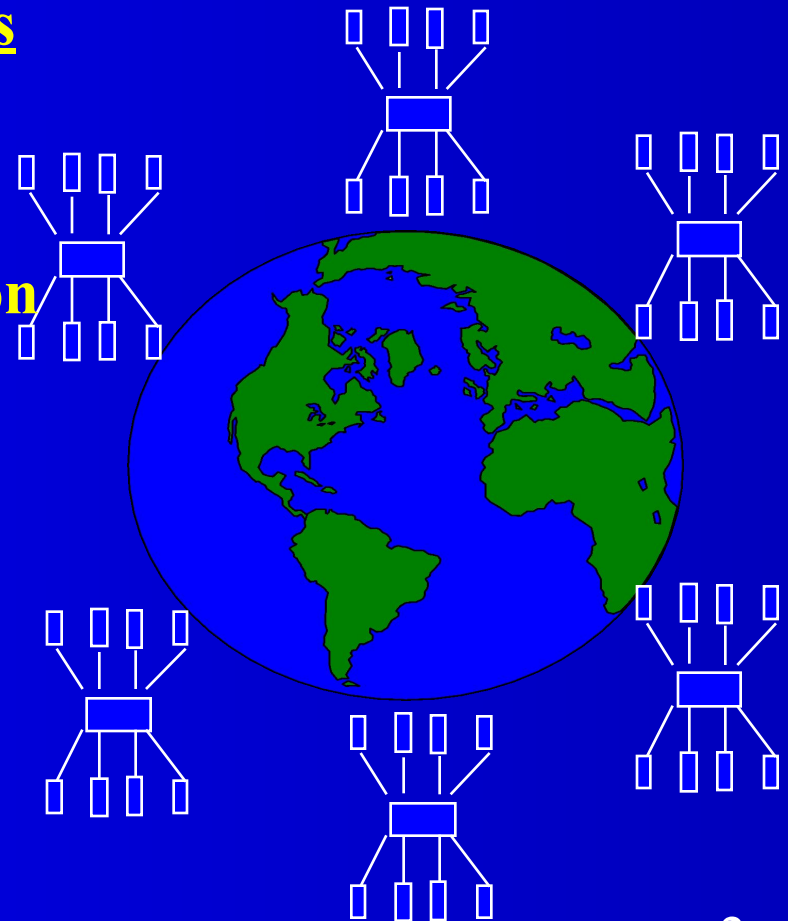
- **Internet and its Evolution**
- **Internet Tools**
- **Web and its Programming**
- **Java for Internet Programming**
- **Java Nuts and Bolts**
- **Java Platform**
- **Developing Applets and Applications**
- **Challenges and Future Directions**



What is the Internet ?

(c) Rajkumar

- It is a global network of computers that communicate with each other using a variety of protocols and overcoming various communication barriers.
- It is like International Telephone System





Internet Technology Evolution

(c) Rajkumar

- Internet is much bigger than what we think
- More than 25 years old
- More than doubling every year
- Technology effect
 - suddenly every body sees the need for a technology
 - like the radio or the TV
- 10 terabytes flows everyday



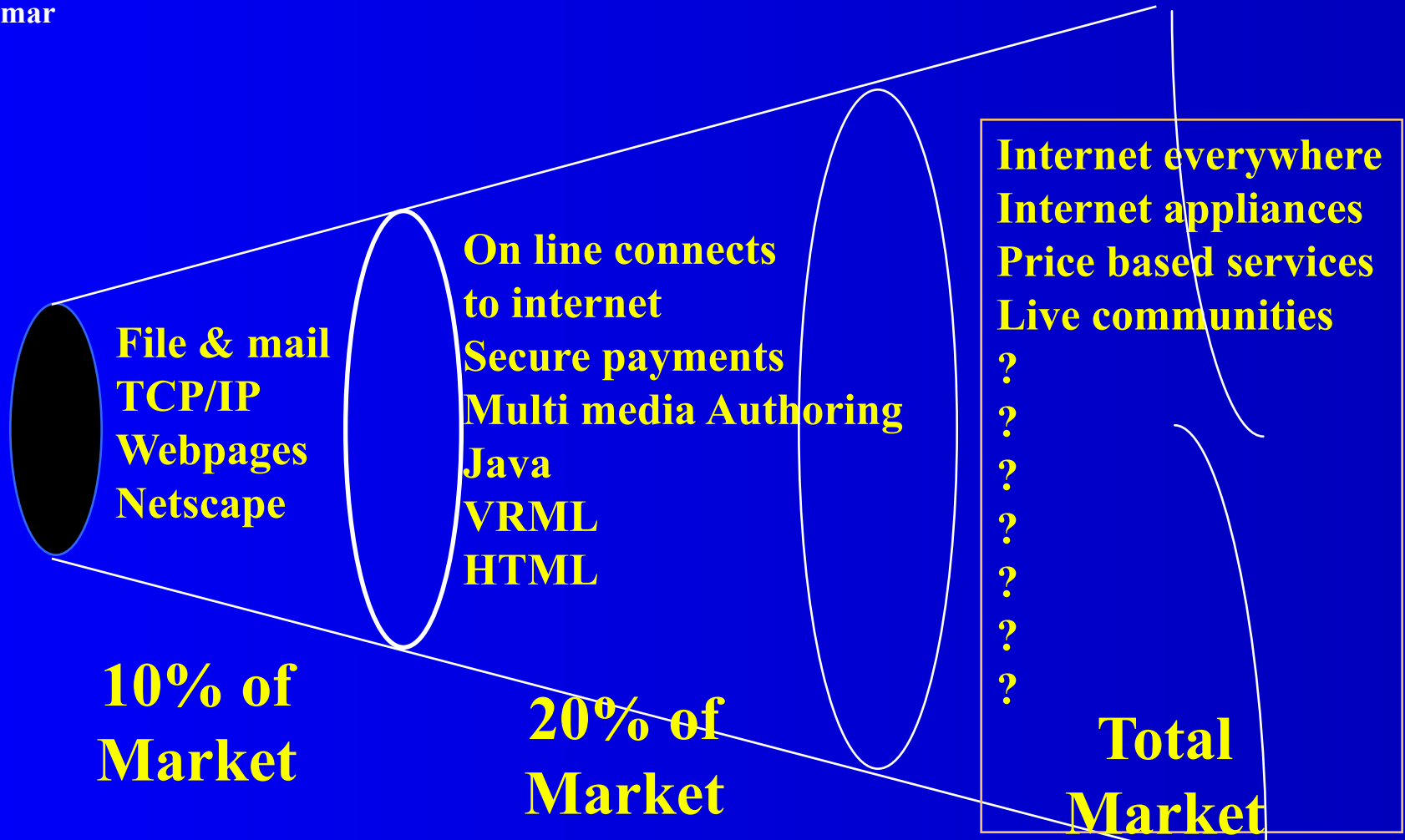
(c) Rajkumar

- Use of internet advertisement/elections/newspapers
- information is public
- Ubiquitous technology
- Network is the computer
- Intranets - internal TCP/IP nets
- PC accounts for 55% of total IT
- Applications tied to platform - API lock-in



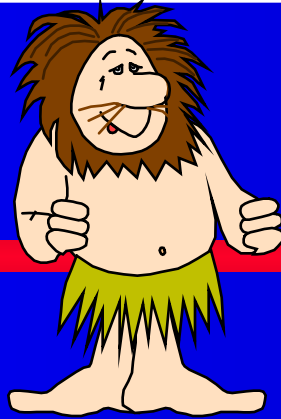
Internet Evolution

(c) Rajkumar





(c) Rajkumar



Early Internet

- **Early Internet supported only email .**
- **File Transfer Protocol development - ftp sites.**
- **Network News was added to the Internet.**
- **Archie - A program to canvass anonymous ftp sites and create a database of what is available**
- **Gopher- A menu-driven interface used to search for information.**
- **Archie and Gopher could answer questions only like ‘what FTP server contains info about “xxxx” ‘**



World Wide Web

(c) Rajkumar

- **World Wide Web conceptualized by Tim Berners-Lee at CERN in Switzerland**
- **Concept of Hypertext led to the development of the Hypertext Markup Language (HTML)**
- **Tim Berners-Lee proposed the ‘Browser’ program**
- **Scientists at CERN designed a TCP/IP based protocol to share Hypertext information called HTTP.**
- **WWW officially is described as a” Wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.**



HTML

(c) Rajkumar

- **Hypertext -A little Hype and a Little Text.**
- **Hypertext point to information which can be local or remotely located.**
- **HTML -Derivative of the SGML(Standard Generalized Markup Language).**
- **HTML -information , commands for the Browser for formatting documents.**
- **HTML -The de-facto language for publishing on the Internet.**
- **Hypermedia- Hyper-links to Multimedia.**



Internet Tools

(c) Rajkumar

- **Browsers-** A tool used to view documents on the WWW
- **Web servers -** Machines which run the HTTP-server Software that respond to HTTP requests which it receives
- **Authoring Tools -** Editors specially made for editing HTML documents
- **Filters -**Tools to convert legacy documents to HTML format
- **Scripting -**Languages used for scripting
- **WAIS-** Wide Area Information Servers (WAIS) for indexing and doing full text searches



How does the Web work ?

(c) Rajkumar

- **Web -Designed around Client/Server Architecture**
- **Web Clients (Web Browsers) -send requests for documents to any Web Server**
- **Web Server -Program that responds to HTTP requests**
- **Hyperlink**
- **Web client connects to the specified Web Server**
- **The server responds by sending the information asked for**
- **The Browser formats the received HTML data and displays it**



How does the Web Work

Send the "INFORMATION ABOUT C-DAC ACTS"

HTTP

The client sends an HTTP message to a computer running a Web Server program and asks for a document

The information about C-DAC ACTS

The web server sends the hypermedia HTML documents to the client. You end up seeing the document on your screen





HTML document

(c) Rajkumar

```
<HTML>
```

```
<TITLE>Centre for Development of Advanced  
Computing
```

```
</TITLE>
```

```
<BODY BGCOLOR="#E7CCCC" TEXT="#000000"  
LINK="#0000FF">
```

```
...
```

```
...
```

```
<A HREF="mailto:webmaster@cdacb.ernet.in">  
webmaster</A>
```

```
</BODY>
```

```
</HTML>
```



(c) Rajkumar

Netscape: Centre for Development of Advanced Computing

File Edit View Go Bookmarks Options Directory Window Help

Back Forward Home Reload Images Open Print Find Stop

Location:

What's New What's Cool Handbook Net Search Net Directory Software

Welcome to the Advanced Computing Training School



	ALL ABOUT ACTS
	Objectives ,
	Information About ACTS DIPLOMA and Courses Offered
	Faculty
	Find Your Dream Job
	Centres



(c) Rajkumar

- **URLs- The Hypertext links we use today are known as Universal Resource Locator**
- **URLs-Each name is unique across the Internet**
- **An URL looks like this**
<http://system.domain.ext:999/dir1/dir2/dir3/file.html?blue#>
- **Parts of a URL are,**
 - Service type, System Name, Port, Directory path, Filename, Search Components or Variables**
- **Service type, System Name, Directory path are the required parts of the URL**



(c) Rajkumar



CGI (Common Gateway Interface)

- **CGI makes the Web a Two-way interface**
- **CGI -lets the user run a script when a web page is accessed**
- **Information from the Web Client is received through simple 'fill-in-the-forms' kind of interface**
- **FORMS - Integrates data sheets, menus, check boxes**
- **CGI makes the Web interactive**
- **CGI -complicated to setup, requires PERL knowledge**
- **HTML books talk less about CGI**



Authoring tools and Filters

(c) Rajkumar

- **Authoring tools- Editors for HTML documents**
- **Editors similar to WYSIWYG word processing programs**
- **Semi-WYSIWYG or completely WYSIWYG**
- **Provide syntax checking and correction**
- **Filters -Convert legacy documents to HTML format**
- **Filters are useful when the documents already exist**
- **Authoring tools- HoTMetaL, HTML Assistant**
-Shareware



Preconfigured v/s Integrated Internet Products

(c) Rajkumar

- **Integrated Internet Products- From multiple vendors**
- **Preconfigured Systems- Web Server and a Client ready to use**
- **Sun's Netra Internet Server**
- **SGI's WebFORCE Indy and WebFORCE Challenge S**
- **Apple's Internet Server Solution**
- **DEC's Internet AlphaServer**
- **Integraph's Web Server 10**



Future Directions

(c) Rajkumar

- **Additions to HTML (Grammar, Maths, Display control)**
- **VRML (Virtual Reality Markup Language)**
- **Security - Using Scrambling and Encryption**
- **Common Client Interface (CCI) Allows Clients to pass information back and forth between the Browser and the External Viewer**
- **Charge Mechanisms**
- **Performance Enhancements- Sending a page and graphics for that page in one connection**



Interesting URLs

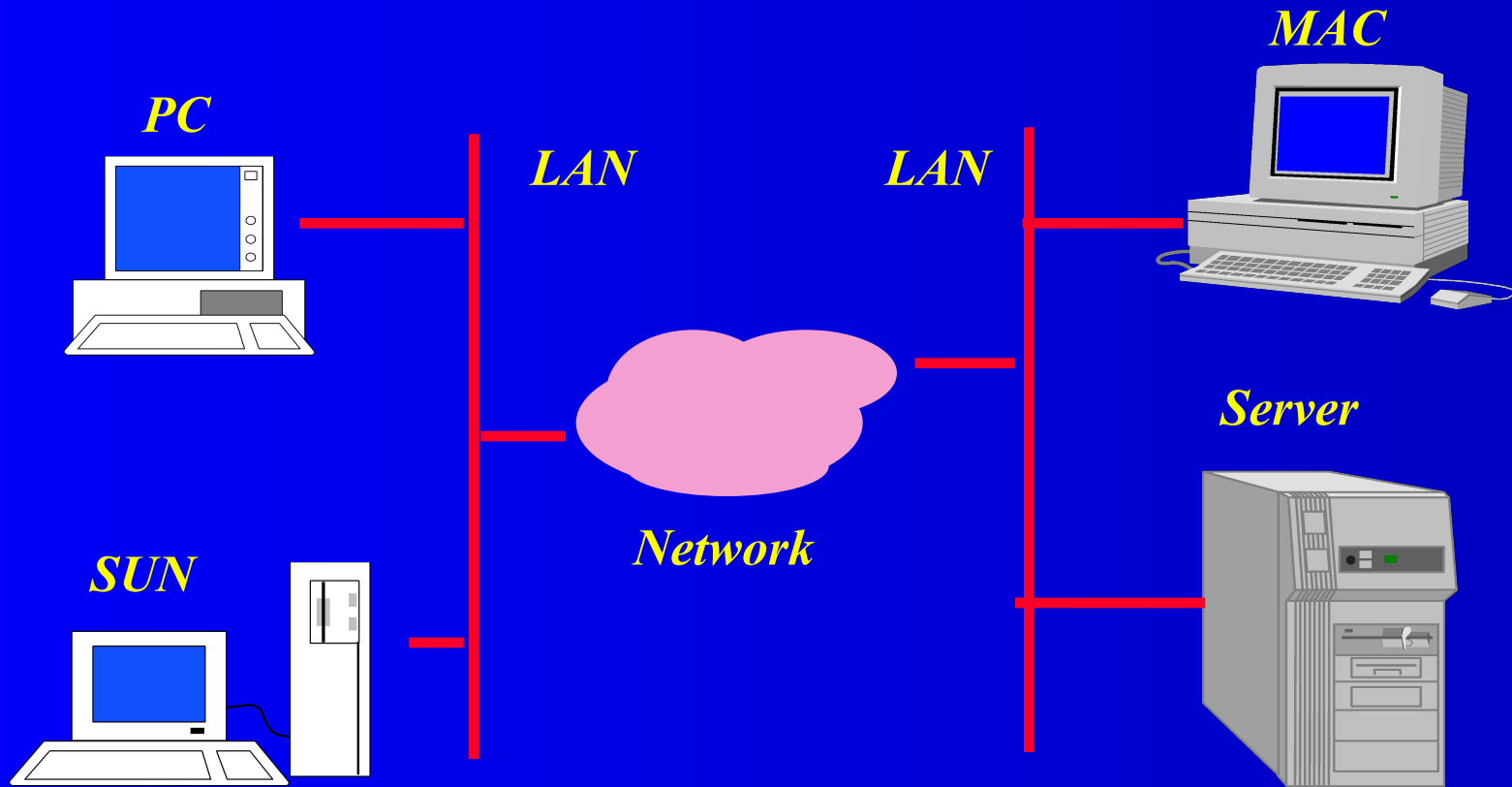
(c) Rajkumar

- <http://www.whitehouse.gov> (The WhiteHouse)
- <http://www.w3.org> (Everything about the WWW)
- <http://sunsite.unc.edu> (Software on Sun)
- <http://www.indnet.org> (India Net Foundation Services)
- <telnet://www.arbornet.org> (Free Public Access Unix System)
- <http://www.infoseek.com> (Search engines, [Add URL](#))
- <http://www.infophil.com> (World Alumni on the net)
- <http://www.rocketmail.com> (Free Email)
- <http://members.tripod.com> (Free Website,2MB space)
- <http://www.bangaloreonline.com> (Offers virtual web servvices for compinies to host their website).
- <http://www.prajavani.com> (Kannada news paper on web)



API Bottleneck

(c) Rajkumar



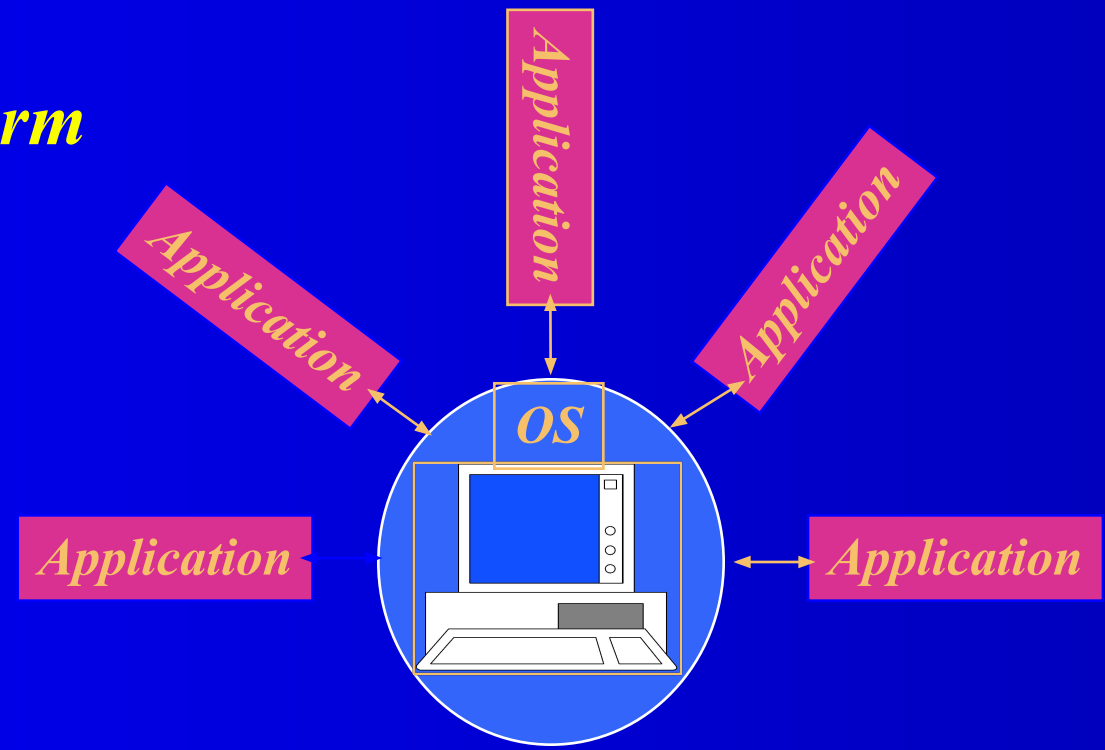


The OS - Platform lock

(c) Rajkumar

Applications tied to OS

OS tied to Platform

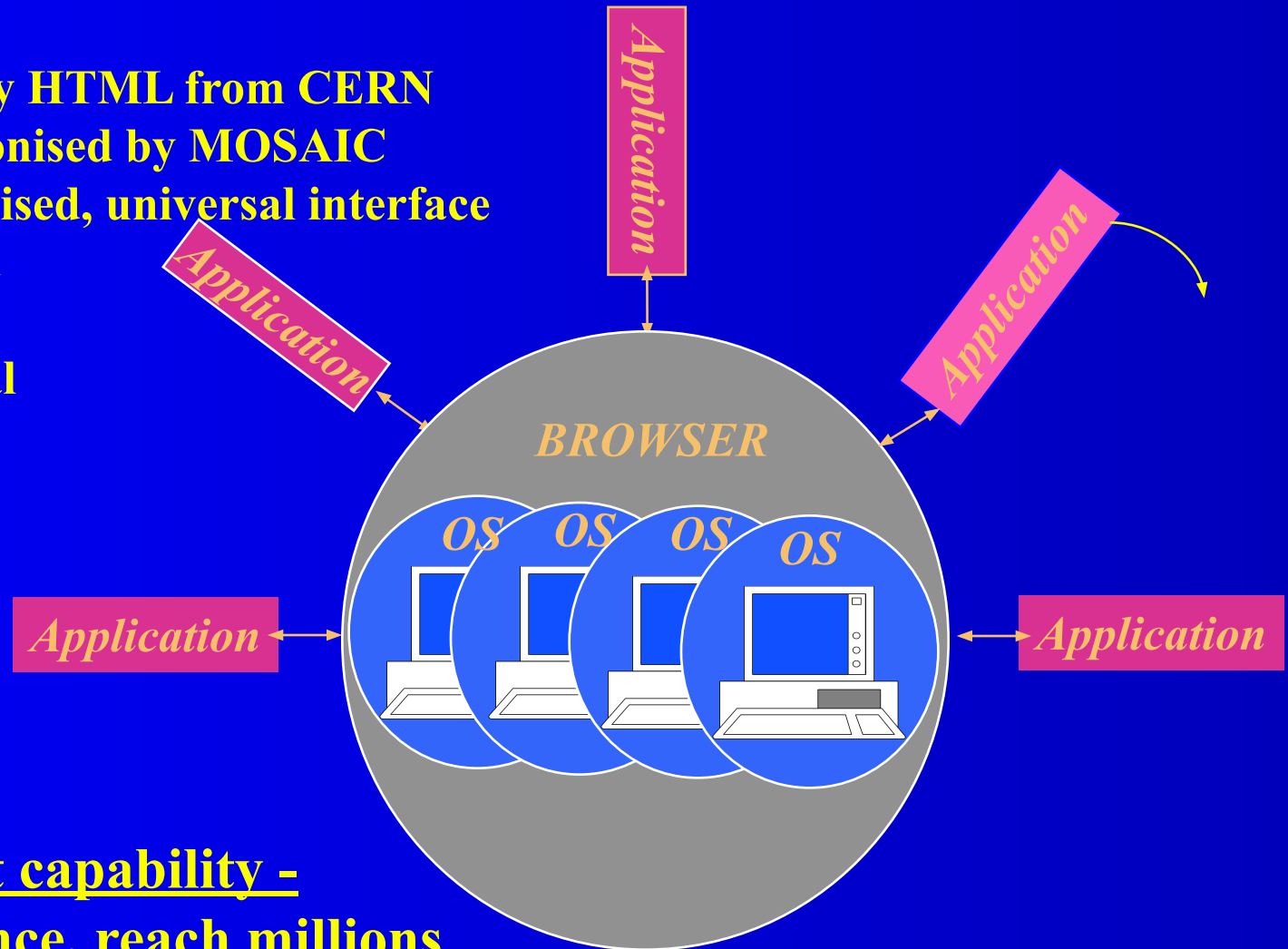




The Web

(c) Rajkumar

- * Seeded by HTML from CERN
- * Revolutionised by MOSAIC
- * Standardised, universal interface to data
- * Graphical



* Broadcast capability -
publish once, reach millions



Making life easier!

(c) Rajkumar

- **Data on the web**
- **Browser platform independent**
- **Click on application - run on any machine**
- **Java the programming language of the 21 century**



Java and Java Computing

(c) Rajkumar



Java Computing



Java - An Introduction

(c) Rajkumar

- Java - The new programming language from Sun Microsystems
- Java - Allows anyone to publish a web page with Java code in it
- Java - CPU Independent language
- Created for consumer electronics
- Java - James , Arthur Van , and others
- Java -The name that survived a patent search
- Oak -The predecessor of Java
- Java is "C++ -- ++ "



Java From 10,000 Ft.

(c) Rajkumar

- According to the world, Java is...
- According to Sun, Java is...
- On closer inspection, Java is



According to the World, Java Is...

(c) Rajkumar

- Snazzy Web pages
- The cross-platform language we want
- The rest-of-the-worlds answer to Bill
- The C++ replacement we need
- The C++ replacement we dont need
- A bunch of hype



According to Sun, Java is...

(c) Rajkumar

- Simple and Powerful
- Object Oriented
- Portable
- Architecture Neutral
- Distributed
- Multi-threaded
- Robust, Secure/Safe
- Interpreted
- High Performance
- Dynamic programming language/platform.

**Buzzword
compliant!**



On Closer Inspection, Java is...

(c) Rajkumar

- **Simple**
- **Pure**
- **Portable**
- **Surprisingly effective**



As a whole, Java is a Comprehensive Programming Solution

(c) Rajkumar

- Object Oriented
- Portable
- High Performance
- Geared for Distributed Environments
- Secure



Java as Object Oriented

(c) Rajkumar

- “Objects all the way down”
- Simple and Familiar: “C++ Lite”
- No Pointers!
- Garbage Collector
- Dynamic Binding
- Single Inheritance with “Interfaces”



Java as Portable

(c) Rajkumar

- Unlike other language compilers, Java compiler generates code (byte codes) for Universal Machine.
- Java Virtual Machine (JVM): Interprets bytecodes at runtime
- Architecture Neutral
- No Link Phase
- Higher Level Portable Features: AWT, Unicode



Total Platform Independence

(c) Rajkumar

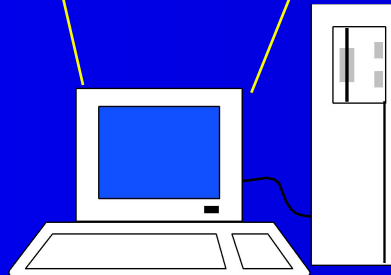
JAVA COMPILER
(translator)

JAVA BYTE CODE
(same for all platforms)

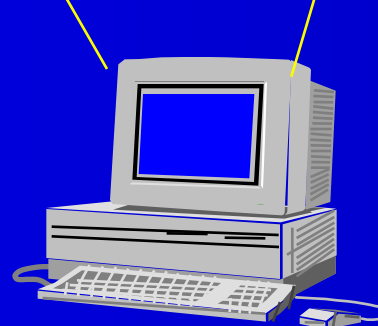
JAVA INTERPRETER
(one for each different system)



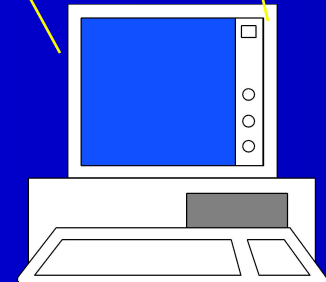
Windows 95



Macintosh



Solaris



Windows NT 4



(c) Rajkumar

Java

Write Once, Run Anywhere



Architecture Neutral & Portable

(c) Rajkumar

- **Java Compiler -Java source code to *bytecode***
- ***Bytecode* - an intermediate form, closer to machine representation**
- **A virtual machine on any target platform interprets the *bytecode***
- **Porting the java system to any new platform involves writing an interpreter that supports the *Java Virtual Machine***
- **The interpreter will figure out what the equivalent machine dependent code to run**



Java as High Performance

(c) Rajkumar

- **JVM uses “lean and mean” bytecodes**
- **Small binary class files**
- **Just-in-time Compilers**
- **Multithreading**
- **Native Methods**



Java in the World of Distributed Computing

(c) Rajkumar

- **Class Loader**
- **Lightweight Binary Class Files**
- **Multithreading**
- **Dynamic**
- **Good communication constructs**
- **Secure**



Java as Secure

(c) Rajkumar

- Language designed as safe
- Strict compiler
- Dynamic Runtime Loading (Verifier)
- Runtime Security Manager



Object Oriented Languages -a Comparison

(c) Raikumar

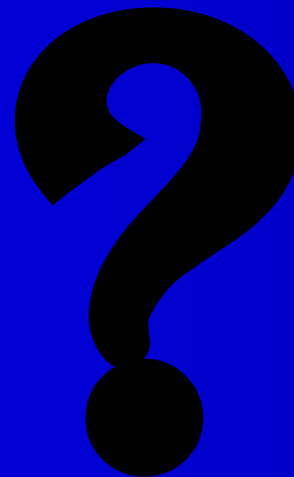
Feature	C++	Objective C	Ada	Java
Encapsulation	Yes	Yes	Yes	Yes
Inheritance	Yes	Yes	No	Yes
Multiple Inherit.	Yes	Yes	No	No
Polymorphism	Yes	Yes	Yes	Yes
Binding (Early/Late)	Both	Both	Early	Late
Concurrency	Poor	Poor	Difficult	Yes
Garbage Collection	No	Yes	No	Yes
Genericity	Yes	No	Yes	No
Class Libraries	Yes	Yes	Limited	Yes



Java better than C++ ?

(c) Rajkumar

- **No Typedefs, Defines, or Preprocessor**
- **No Global Variables**
- **No Goto statements**
- **No Pointers**
- **No Unsafe Structures**
- **No Multiple Inheritance**
- **No Operator Overloading**
- **No Automatic Coercions**
- **No Fragile Data Types**





Basic Data Types

(c) Rajkumar

- **Types**

- boolean either true or false

- char 16 bit Unicode 1.1

- byte 8-bit integer (signed)

- short 16-bit integer (signed)

- int 32-bit integer (signed)

- long 64-bit integer (signed)

- float 32-bit floating point (IEEE 754-1985)

- double 64-bit floating point (IEEE 754-1985)

- **String** (class for manipulating strings)

- **Java uses Unicode to represent characters internally**



(c) Rajkumar

**Java Integrates
Power of Compiled Languages
and
Flexibility of Interpreted
Languages**



Two Types of JavaApplications

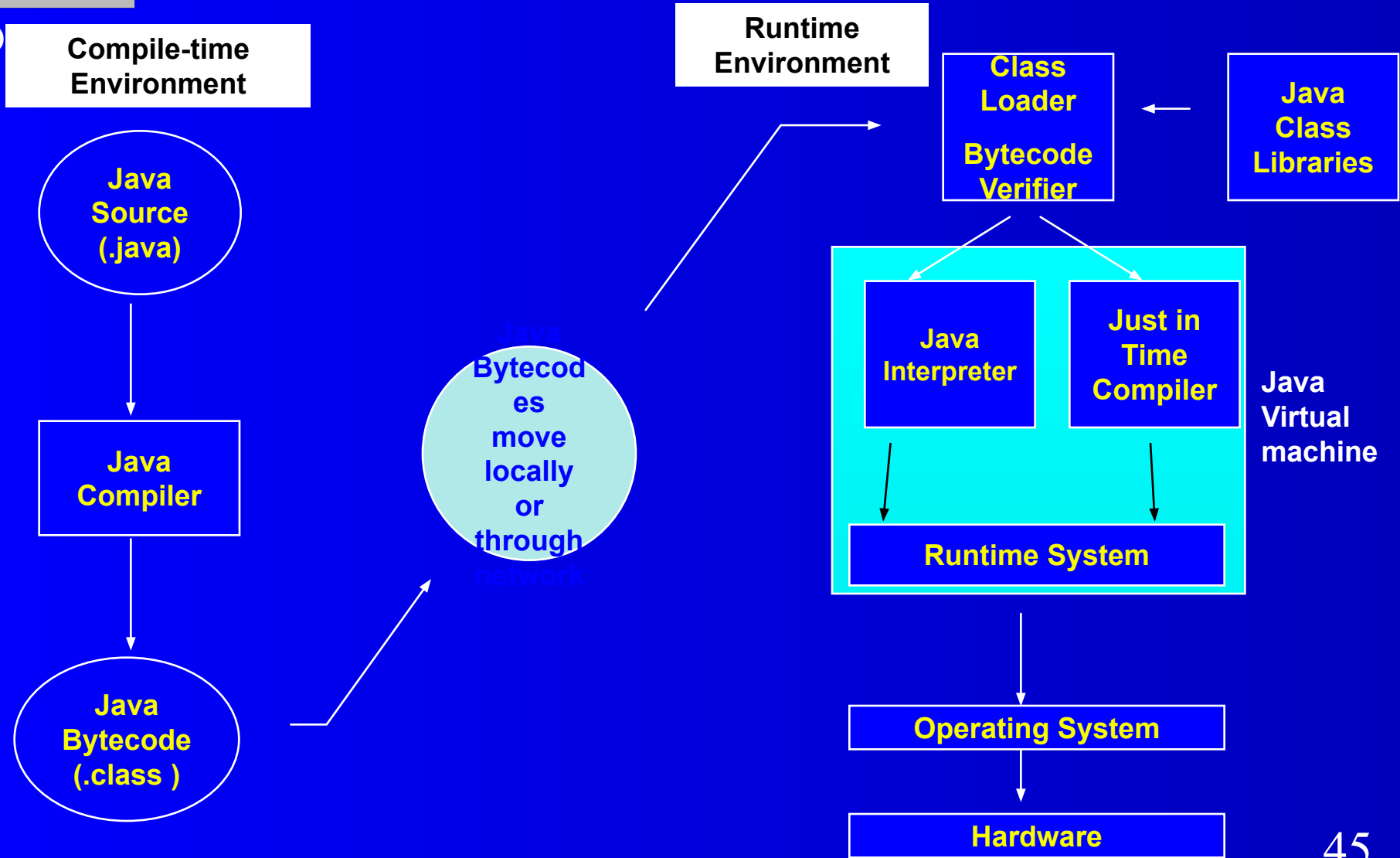
(c) Rajkumar

- Different ways to write/run a Java codes are:
 - Application- A stand-alone program that can be invoked from command line . A program that has a “main” method
 - Applet- A program embedded in a web page , to be run when the page is browsed . A program that contains no “main” method
- Application -Java interpreter
- Applets- Java enabled web browser (Linked to HTML via <APPLET> tag. in html file)



Java Environment/ Life Cycle of Java Code

(c)



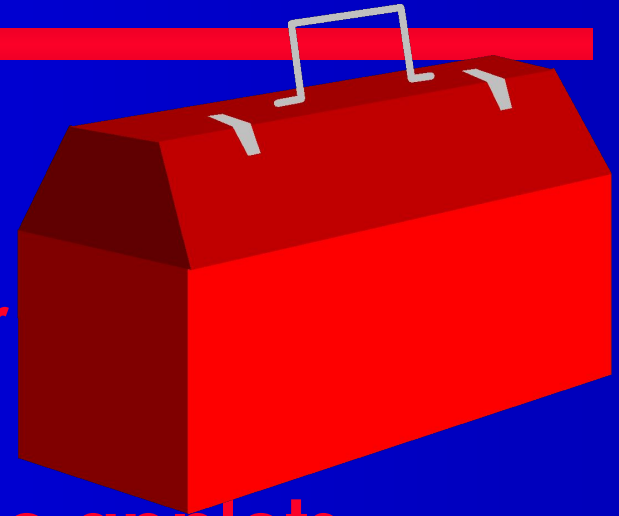


Java Development Kit

(c) Rajkumar

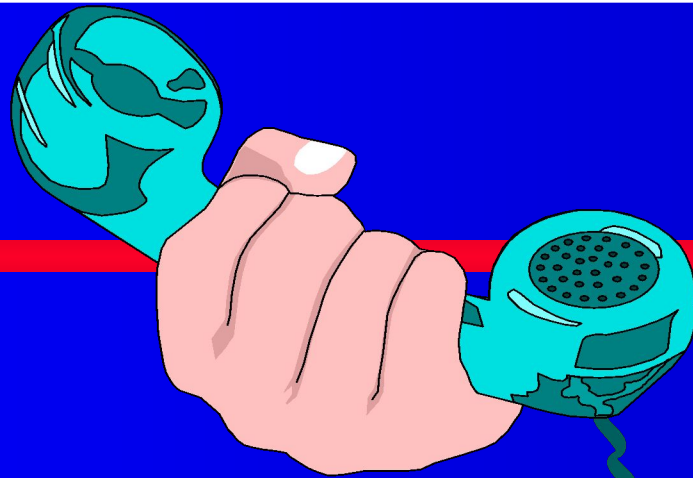
- **javac** - The Java Compiler
- **java** - The Java Interpreter
- **jdb**- The Java Debugger
- **appletviewer** -Tool to run the applets

- **javap** - to print the Java bytecodes
- **javaprof** - Java profiler
- **javadoc** - documentation generator
- **javah** - creates C header files





(c) Rajkumar



Hello Internet

```
// hello.java: Hello Internet program
class HelloInternet
{
    public static void main(String args[])
    {
        System.out.println("Hello Internet");
    }
}
```



Program Processing

(c) Rajkumar

- **Compilation**

```
# javac hello.java  
results in HelloInternet.class
```

- **Execution**

```
# java HelloInternet  
Hello Internet  
#
```




Simple Java Applet

(c) Rajkumar

```
// HelloWorld.java: A sample applet
import java.applet.Applet;
public class HelloWorld extends Applet {
    public void paint(Graphics g)
    {
        g.drawString("Hello World !",25,25);
    }
}
```



Calling an Applet

(c) Rajkumar

```
<HTML>  
<TITLE> Hello Worls Applet </TITLE>  
<APPLET code="HelloWorld.class" width=500 height=500>  
</APPLET>  
</HTML>
```



Execution of Applets

(c) Rajkumar

1

**APPLET
Development
"hello.java"
AT
CDAC-India**

2

**hello.class
AT C-DAC'S
WEB
SERVER**

3

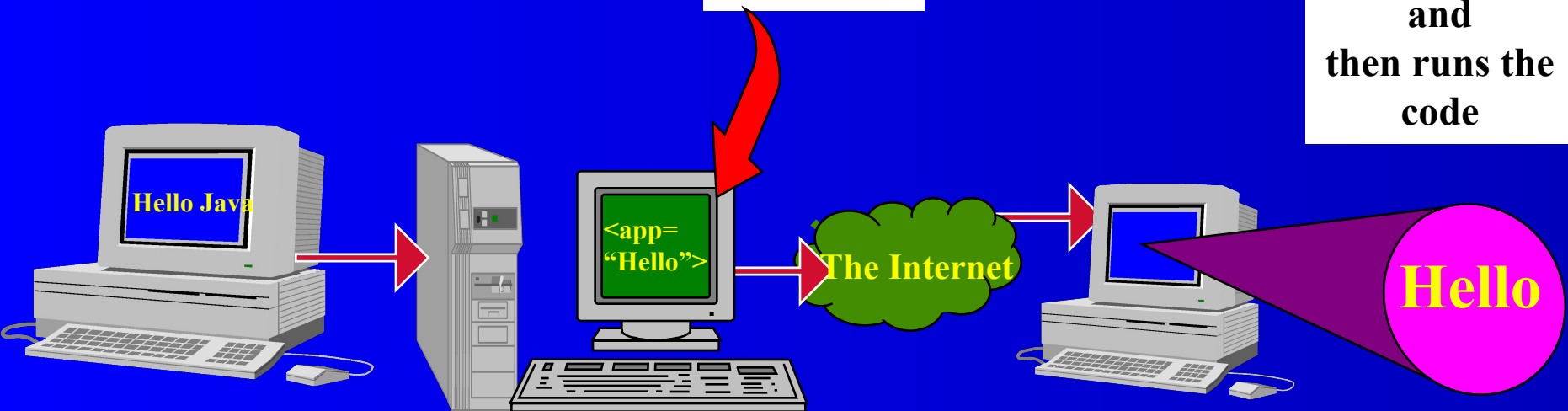
**Create
Applet
tag in
HTML
document**

4

**Accessing
from
CRAY Corp.
(USA)**

5

**The browser
creates
a new
window and
a new thread
and
then runs the
code**





Web Perspective

(c) Rajkumar

- How did Web interactions work?
- How do they work with Java?
- Distributed Java objects and the Web

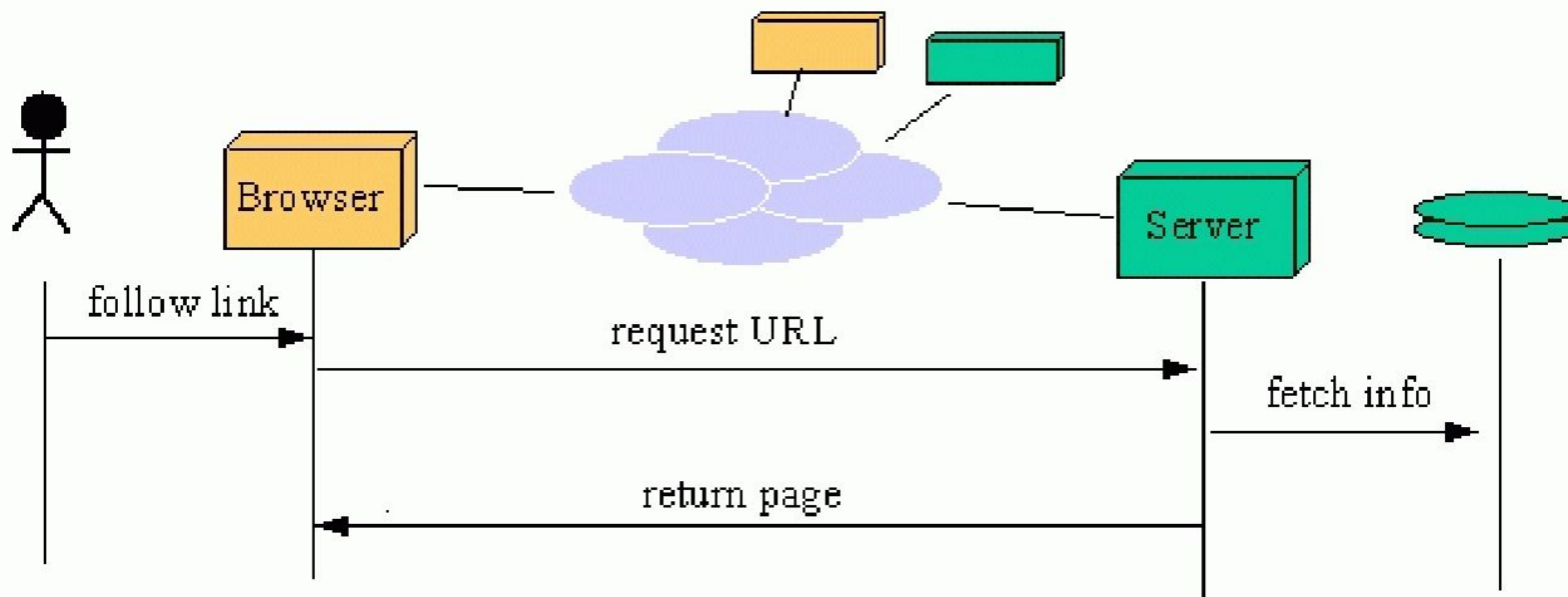


Classical Web Perspective

(c) Rajkumar

Classic Web Interactions

- Page-level updates and interactivity

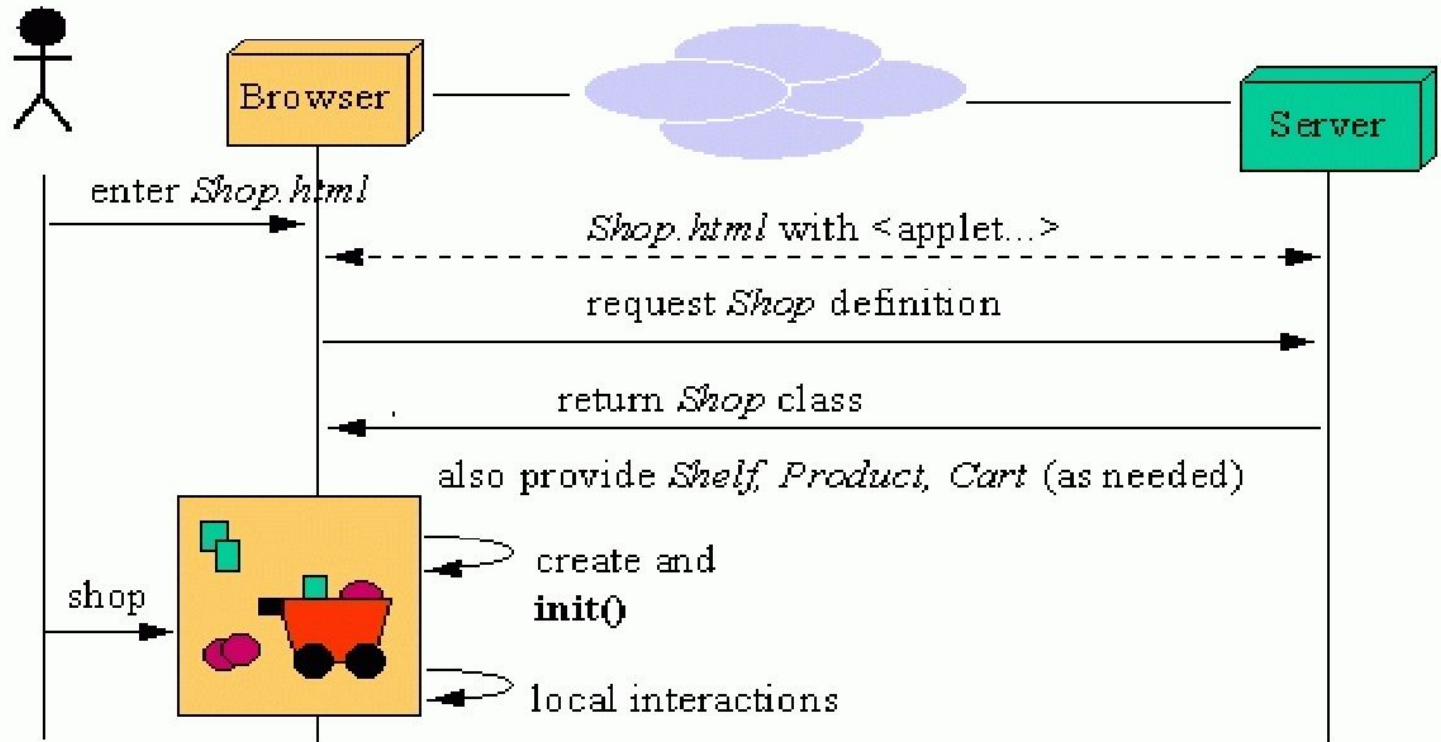




Java Web Perspective

(c) Rajkumar

Web Interactions With Java



- ❑ Automatic download of applet and supporting code
- ❑ Client-side instances and interactions



Significance of downloading Applets

(c) Rajkumar

- **Interactive WWW**
- **Flashy animation instead of static web pages**
- **Applets react to users input and dynamically change**
- **Display of dynamic data**
- **WWW with Java - more than a document publishing medium**

<http://www.javasoft.com/applets/alpha/applets/StockDemo/standalone.html>



Power of Java and the Web

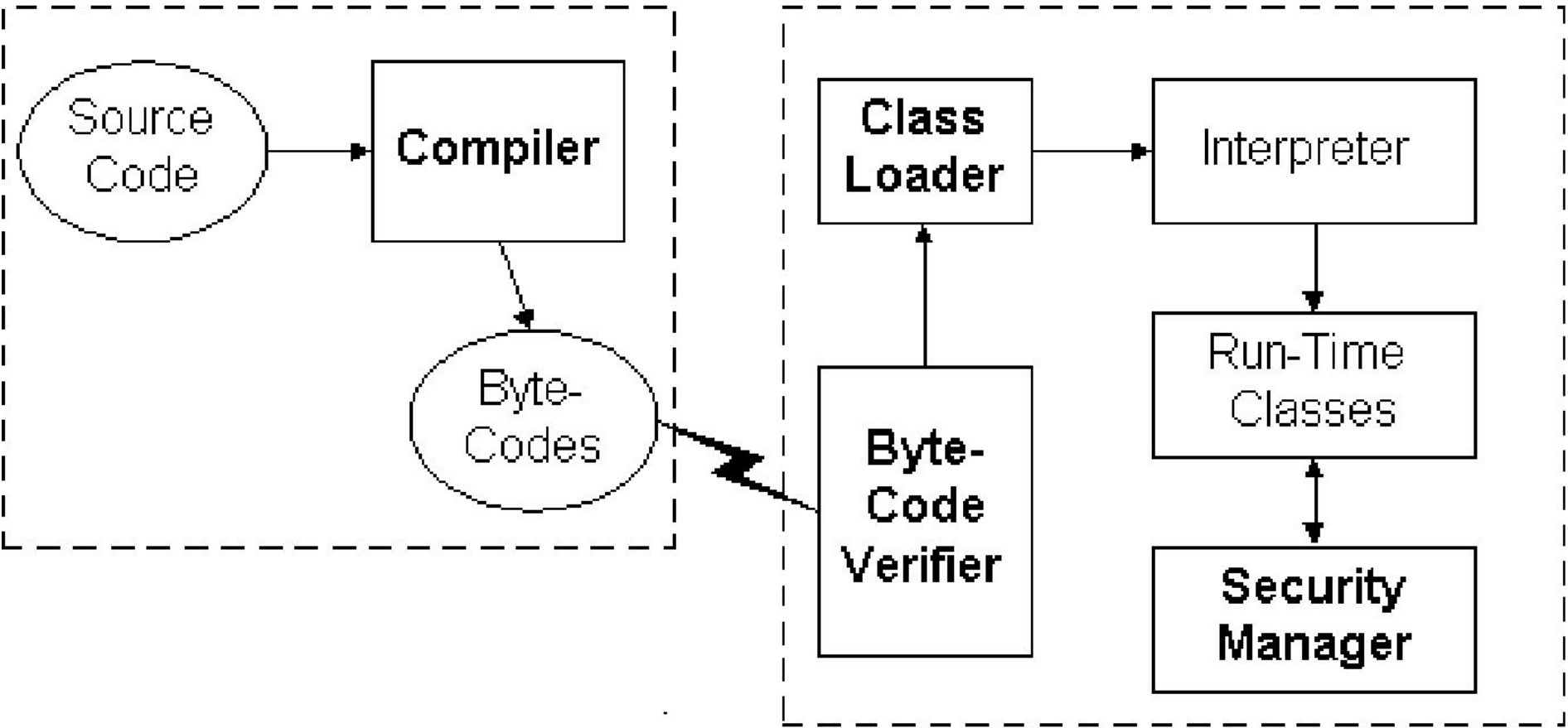
(c) Rajkumar

- **Deliver applications, not just information**
- **Eliminate porting**
- **Eliminate end-user installation**
- **Slash software distribution costs**
- **Reach millions of customers - instantly**



Lifecycle of Java Code

(c) Rajkumar





Bytecode Verifier

(c) Rajkumar

- Called when class is first loaded in runtime environment
- Verifies bytecodes meet certain set of properties
- Verifier uses Theorem Prover
- Verified code runs faster
- After verification, interpreter defines memory layout



Class Loader

(c) Rajkumar

- **Unique “Namespace” for each origin**
- **Local namespace classes are called “built-ins”**
- **Prevents class “spoofing”**



Security Manager

(c) Rajkumar

- Prevents unauthorized disk read/writes
- Restricts network access
- Other access restrictions (native methods)
- Implementation is browser dependent



General Language Features

(c) Rajkumar

- **C/C++ like syntax**
- **No pointers**
- **Objects all the way down**
- **Objects request services of other objects through *messages***
- **Messages result in invocation of class *methods***



Removed From C++

(c) Rajkumar

- **Operator overloading**
- **Pointers and Array/pointers**
- **Multiple-inheritance of implementation**
- **Enum, typedef, #define**
- **Copy constructors, destructors**
- **Templates**
- **And other stuff....**



Added or Improved over C++

(c) Rajkumar

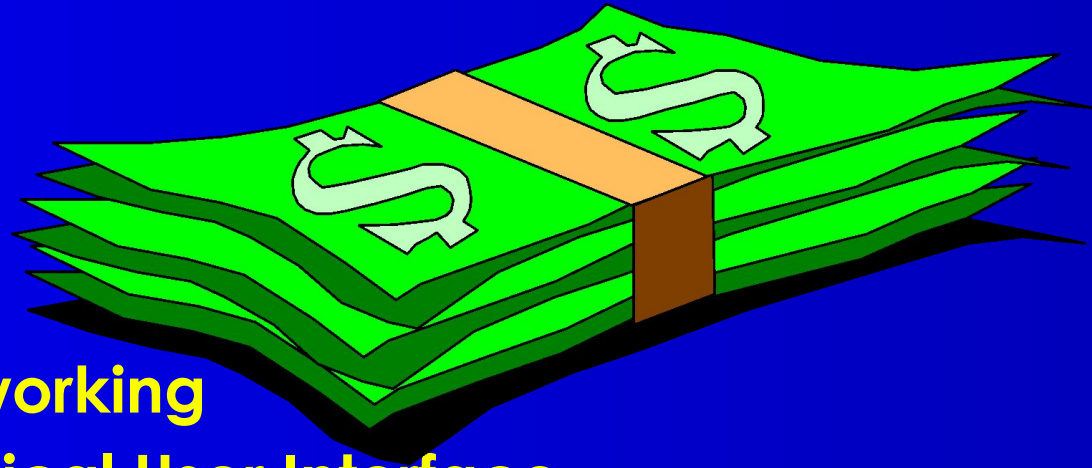
- **Interfaces: type Vs. class**
- **Garbage collection**
- **Exceptions (More powerful than C++)**
- **Strings**
- **Instanceof**
- **Package**
- **Multi-threads**



Rich Object Environment

(c) Rajkumar

- **Core Classes**
 - language
 - Utilities
 - Input/Output
 - Low-Level Networking
 - Abstract Graphical User Interface
- **Internet Classes**
 - TCP/IP Networking
 - WWW and HTML
 - Distributed Programs





Main Packages

(c) Rajkumar

- `java.lang`
- `java.util`
- `java.io`
- `java.awt`
- `java.awt.image`
- `java.applet`
- `java.net`



(c) Rajkumar

Java Fundamentals

Constructs

Graphics

Multithreading

Streams and Networking

Networking



Unit I--Java Constructs

(c) Rajkumar

- **what is Java, basic constructs, including**
 - classes and objects
 - constructors,
 - this and super keywords,
 - inheritance,
 - abstract classes, interfaces,
 - inner classes,
 - exceptions.



Unit II--Graphics Programming

(c) Rajkumar

- **How to build Graphical User Interfaces in Java:**
 - GUI components,
 - event handling,
 - layout management.



Unit III--Advanced Features

(c) Rajkumar

- Applets,
- Threads,
- Streams I/O,
- Networking



Unit I -- What is Java ?

(c) Rajkumar

- **A programming language:**
 - Object oriented (no friends, all functions are members of classes, no function libraries -- just class libraries)
 - simple (no pointer arithmetic, no need for programmer to deallocate memory)
 - platform independent
 - dynamic
 - interpreted



(c) Rajkumar

- **Eight basic types**

- 4 integers (byte, short, int, long) [int a;]
- 2 floating point (float, double) [double a;]
- 1 character (char) [char a;]
- 1 boolean (boolean) [boolean a;]

- **Everything else is an object**

- String s;



Classes and objects

(c) Rajkumar

- **declaring a class**

```
class MyClass {  
    member variables;  
  
    ...  
    member functions () ;  
  
    ...  
} // end class MyClass
```




Java programs

(c) Rajkumar

- **Two kinds**

- Applications

- have main()
 - run from the OS prompt

- Applets

- have init(), start(), stop(), paint(), update(), repaint(), destroy()
 - run from within a web page



The first Java Application

(c) Rajkumar

```
class MyApp {  
    public static void main(String s [ ] ) {  
        System.out.println("Hello World");  
    }  
} // end class MyApp
```



Declaring and creating objects

(c) Rajkumar

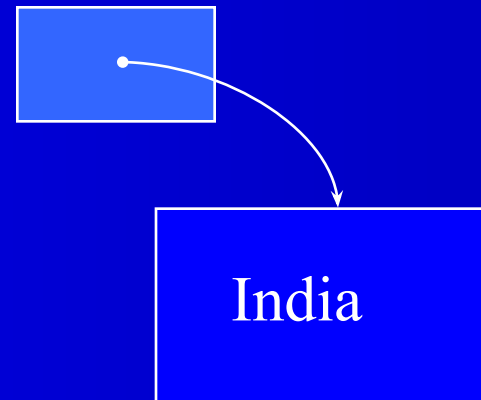
- **declare a reference**

- String s;



- **create/define an object**

- s = new String (“India”);





Arrays (are objects in Java)

(c) Rajkumar

- **declare**

- `int a [] ; // 1-dim`

- `int [] b ; // 1-dim`

- `int [] c [] ; // 2-dim`

- `int c [] [] ; // 2-dim`

- **allocate space**

- `a = new int [7];`

- `c = new int [7][11];`



Arrays have length

(c) Rajkumar

- **used to retrieve the size of an array**
 - `int a [] = new int [7]; // 1-dim`
 - `System.out.println(a.length);` will print '7'
 - `int b [] [] = new int [7] [11];`
 - `System.out.println(a.length);` will print '7'
 - `System.out.println(b.length * b[0].length);` will print '77'



... this is because

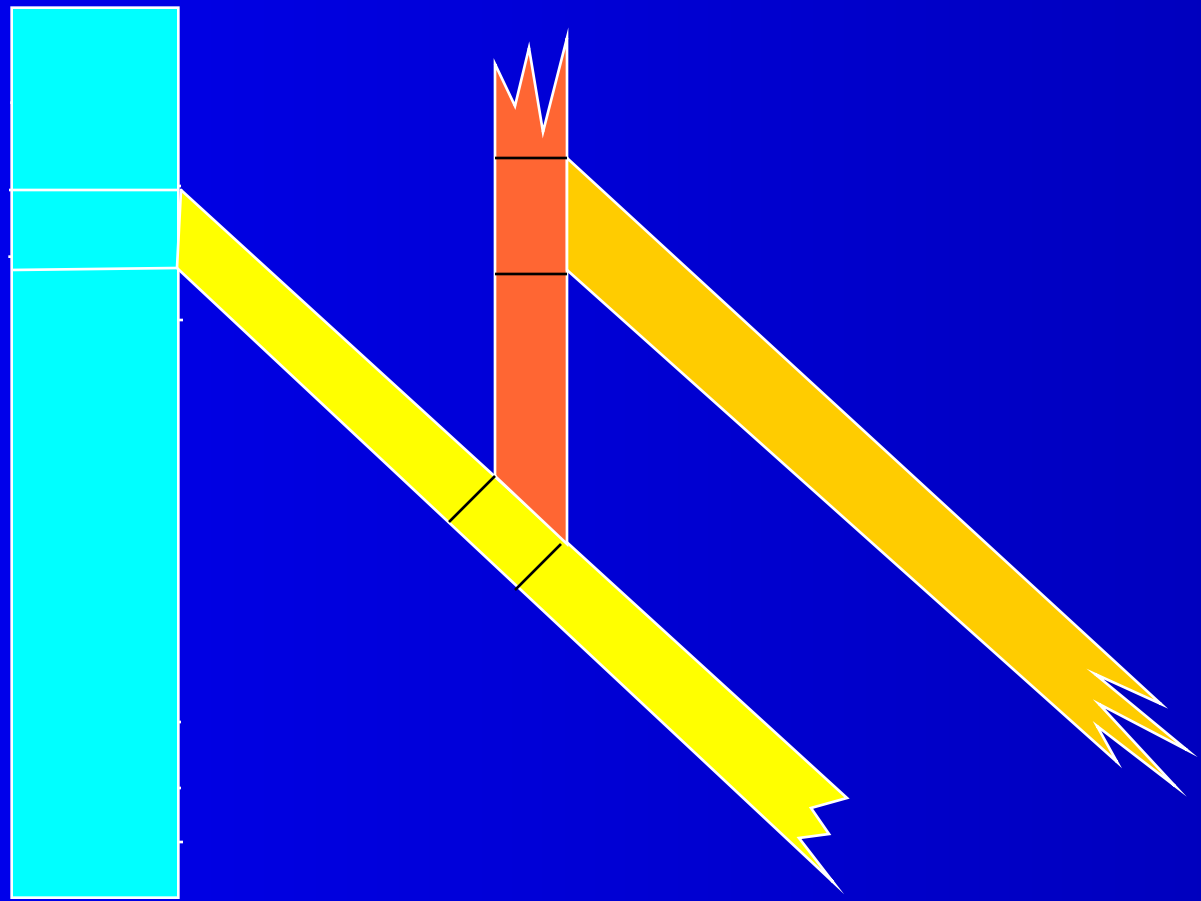
(c) Rajkumar

- Let `int [][][] array = new int [7][11][10][21]` , then ...
- `array.length * array[3].length * array[3][5].length * array[3][5][2].length` is `7 x 11 x 10 x 21`



... this is because

(c) Rajkumar





Constructors

(c) Rajkumar

- All objects are created through constructors
- They are invoked automatically

```
class Weight {  
    int lb; int oz;  
    public Weight (int a, int b ) {  
        lb = a; oz = b;  
    }  
}
```




this keyword

(c) Rajkumar

- refers to “this” object (object in which it is used)
- usage:
 - with an instance variable or method of “this” class
 - as a function inside a constructor of “this” class
 - as “this” object, when passed as parameter



this :: with a variable

(c) Rajkumar

- refers to “this” object’s data member

```
class Weight {  
    int lb; int oz;  
    public Weight (int lb, int oz ) {  
        this.lb = lb; this.oz = oz;  
    }  
}
```



this :: with a method

(c) Rajkumar

- refers to another method of “this” class

```
class Weight {  
    public int m1 (int a) {  
        int x = this.m2(a); return x;  
    }  
    public int m2(int b) { return b*7 ; }  
}
```



this :: as a function inside a constructor of “this” class

(c) Rajkumar

- must be used with a constructor

```
class Weight {  
    int    lb, oz;  
    public Weight (int a, int b) { lb = a; oz = b; }  
    }  
    public Weight (int x) { this( x, 0); }  
}
```

Constructor is also overloaded (Java allows overloading of all methods, including constructors)



this :: as “this” object, when passed as parameter

(c) Rajkumar

- refers to the object that used to call the calling method

```
class MyApp {  
    int a;  
  
    public static void main(String [] s ) { new MyApp().myMethod(); }  
  
    public void myMethod() { yourMethod(this); }  
  
    public void yourMethod(MyApp inMyApp) { inMyApp.a = 77; }  
}
```



static keyword

(c) Rajkumar

- means “global” --all all objects refer to the same storage.
- applies to variables or methods”
- usage:
 - with an instance variable of a class
 - with a method of a class



static keyword (with variables)

(c) Rajkumar

```
class PurchaseOrder {  
    private static int POCCount; // var. 'a' is shared by all objects of this class  
  
    public static void main(String [] s ) {  
        PurchaseOrder po1 = new PurchaseOrder();  
        po1.updatePOCount();  
    }  
  
    public void updatePOCount() { POCCount++; }  
}
```



static keyword (w/ methods)

(c) Rajkumar

```
class Math {  
    public static double sqrt(double x) {  
        // calculate  
        return result;  
    }  
}  
class MyApp {  
    public static void main(String [] s ) {  
        double dd;  
        dd = Math.sqrt(7.11);  
    }  
}
```




Inheritance (subclassing)

(c) Rajkumar

```
class Employee {  
    protected String name;  
    protected double salary;  
    public void raise(double dd) {  
        salary += salary * dd/100;  
    }  
    public Employee ( ... ) { ... }  
}
```



Manager can be made a sub/derived-class of Employee

(c) Rajkumar

```
class Manager extends Employee {  
    private double bonus;  
    public void setBonus(double bb) {  
        bonus = salary * bb/100;  
    }  
    public Manager ( ... ) { ... }  
}
```



Overriding (methods)

(c) Rajkumar

```
class Manager extends Employee {  
    private double bonus;  
    public void setBonus(double bb) { ... }  
    public void raise(double dd) {  
        salary += salary * dd/100 + bonus;  
    }  
    public Manager ( ... ) { ... }  
}
```



Inheritance and Constructors

(c) Rajkumar

```
class First {  
    public First() { System.out.println(" First class "); }  
}  
public class Second extends First {  
    public Second() { System.out.println("Second class"); }  
}  
public class Third extends Second {  
    public Third() {System.out.println("Third class");}  
}
```

First class
Second class
Third class

Topmost class constructor is invoked first
(like us ...grandparent-->parent-->child->)



access modifiers

(c) Rajkumar

- **private**
 - same class only
- **public**
 - everywhere
- **protected**
 - same class, same package, any subclass
- **(default)**
 - same class, same package



super keyword

(c) Rajkumar

- refers to the superclass (base class)
- usage:
 - with a variable or method (most common with a method)
 - as a function inside a constructor of the subclass



super :: with a method

(c) Rajkumar

```
class Manager extends Employee {  
    private double bonus;  
    public void setBonus(double bb) { ... }  
    public void raise(double dd) { //overrides raise() of  
        Employee  
        super.raise(dd); // call Employee's raise()  
        salary += bonus;  
    }  
    public Manager ( ... ) { ... }  
}
```



super :: as a function inside a constructor of the subclass

(c) Rajkumar

```
class Manager extends Employee {  
    private double bonus;  
    public void setBonus(double bb) { ...}  
    public Manager ( String name, double salary, double bonus ) {  
        super(name, salary);  
        this.bonus = bonus;  
    }  
}
```




final keyword

(c) Rajkumar

- means “constant”
- applies to
 - variables (makes a var. constant), or
 - methods (makes a method non-overridable), or
 - classes (makes a class non-subclassable means “objects cannot be created”).



final keyword with a variable

(c) Rajkumar

```
class Math {  
  
    public final double pi = 3.1412;  
    public static double method(double x) {  
        double x = pi * pi;  
    }  
}
```

note: variable **pi** is made “read-only”



final keyword with a method

(c) Rajkumar

```
class Employee {  
    protected String name;  
    protected double salary;  
    public final void raise(double dd) {  
        salary += salary * dd/100;  
    }  
    public Employee ( ... ) { ... }  
}
```

then: cannot override method raise() inside the Manager class



final keyword with a class

(c) Rajkumar

```
final class Employee {  
    protected String name;  
    protected double salary;  
    public void raise(double dd) {  
        salary += salary * dd/100;  
    }  
    public Employee ( ... ) { ... }  
}
```

then: cannot create class Manager as a subclass of class Employee (all are equal)



abstract classes and interfaces

(c) Rajkumar

- **abstract classes**
 - may have both implemented and non-implemented methods
- **interfaces**
 - have only non-implemented methods
- **(concrete classes)**
 - have all their methods implemented



sample abstract class

(c) Rajkumar

```
abstract class TwoDimensionalGeoFigure {  
    public abstract double area();  
    public abstract double perimeter();  
    public abstract void printInfo();  
    public void setOutlineColor(Color cc) {  
        // code to set the color  
    }  
    public void setInsideColor(Color cc) {  
        // code to set the color  
    }  
}
```



sample interface

(c) Rajkumar

```
interface ResponceToMouseClicked {  
    public void mouseDown();  
    public void mouseUp();  
    public void mouseDoubleClick();  
}
```

```
class ConcreteMouseClicked implements  
ResponseToMouse Click {  
    // all above methods implemented here  
}
```



Exceptions (error handling)

(c) Rajkumar

A nice way to handle errors in Java programs

code without exceptions:

```
...
int a = 7, b = 0, result;
if ( b != 0) {
    result = a/b;
}
else {
    System.out.println("b is zero");
}
...
```

code with exceptions:

```
...
int a = 7, b = 0, result;
try {
    result = a/b;
}
catch (ArithmeticException e) {
    System.out.println("b is zero");
}
...
```




Exceptions (cont'd)

(c) Rajkumar

```
...
int a = 7, b = 0, result;
try {
    result = a/b;
    /// more code .. reading from a file
}
catch (ArithmeticException e ) {
    System.out.println("b is zero");
}
catch (IOException e ) {
    System.out.println("Can't read");
}
finally {
    Sysytem.out.println("Closing file");
    /// code to close file
}
...
```



methods throwing exceptions

(c) Rajkumar

```
public int divide (int x, int y ) throws ArithmeticException {  
  
    if (y == 0 ) {  
        throw new ArithmeticException();  
    }  
    else {  
        return a/b ;  
    }  
} // end divide()
```



Defining your own exceptions

(c) Rajkumar

```
class MyException extends ArithmeticException  
{  
    frm
```

```
public int divide (int x, int y ) throws MyException {  
  
    if (y == 0 ) {  
        throw new MyException();  
    }  
    else {  
        return a/b ;  
    }  
} // end divide()
```



(c) Rajkumar

GUI Programming in Java

(AWT and Event Handling)



AWT - Abstract Windowing Toolkit

(c) Rajkumar

- **Single Windowing Interface on Multiple Platforms**
- **Supports functions common to all window systems**
- **Uses Underlying Native Window system**
- **AWT provides**
 - **GUI widgets**
 - **Event Handling**
 - **Containers for widgets**
 - **Layout managers**
 - **Graphic operations**



AWT - Abstract Window Toolkit

(c) Rajkumar

- **Portable GUI - preserves native look & feel**
- **Standard GUI Components (buttons...)**
- **Containers - Panels, Frames, Dialogs**
- **Graphics class for custom drawing**
- **Layouts responsible for actual positioning of components:**
 - **BorderLayout, GridLayout, FlowLayout, null layout**



Adding Components via Layouts

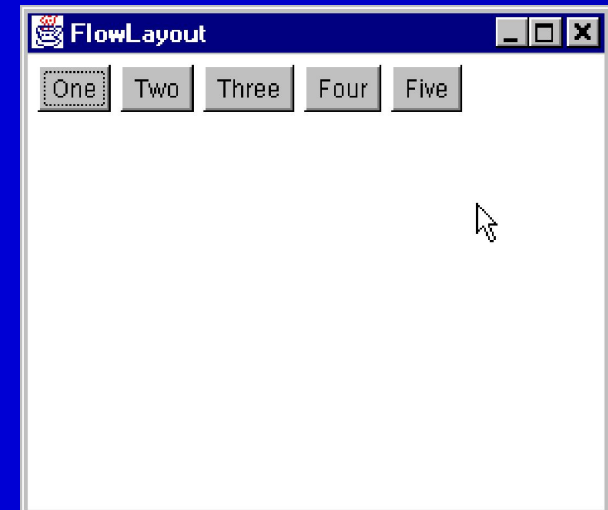
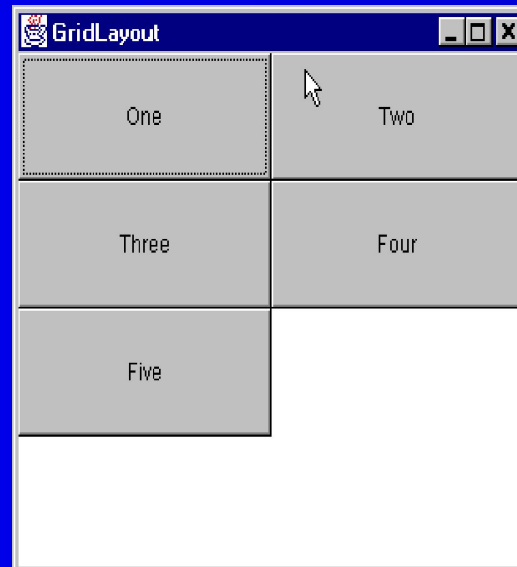
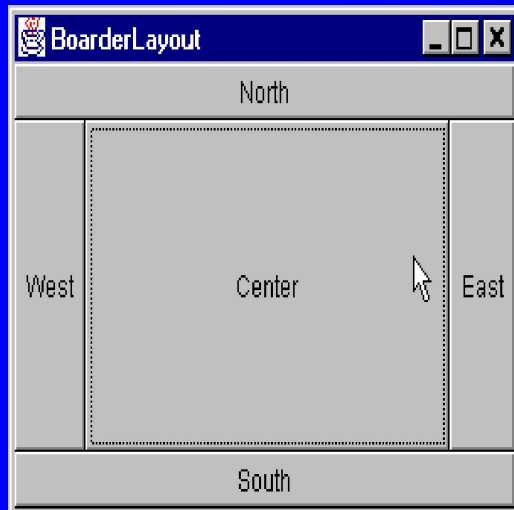
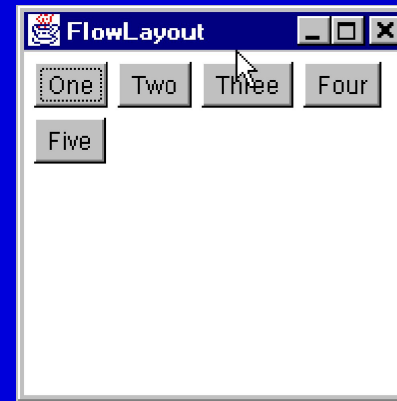
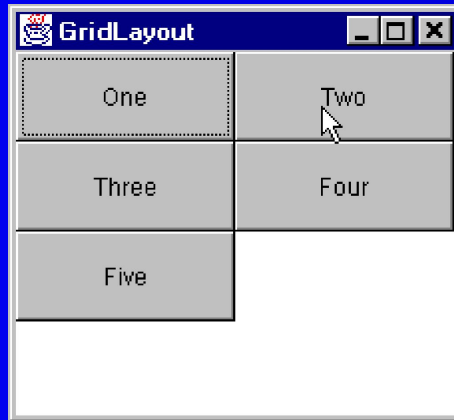
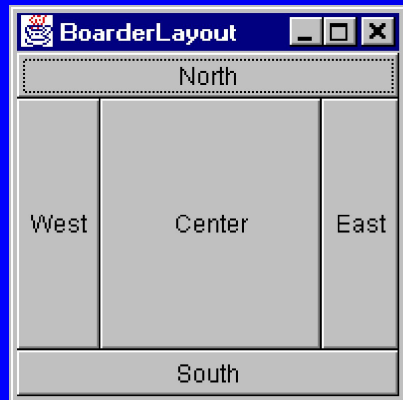
(c) Rajkumar

```
setLayout(new BorderLayout());  
// Add text field to top  
add("North",new TextField());  
// Create the panel with buttons at the bottom...  
Panel p = new Panel(); // FlowLayout  
p.add(new Button("OK"));  
p.add(new Button("Cancel"));  
add("South",p);
```



Adding Components via Layouts

(c) Rajkumar





Building Graphical User Interfaces

(c) Rajkumar

- **import java.awt.*;**
- **Assemble the GUI**
 - use GUI components,
 - basic components (e.g., Button, TextField)
 - containers (Frame, Panel)
 - set the positioning of the components
 - use Layout Managers
- **Attach events**



A sample GUI program

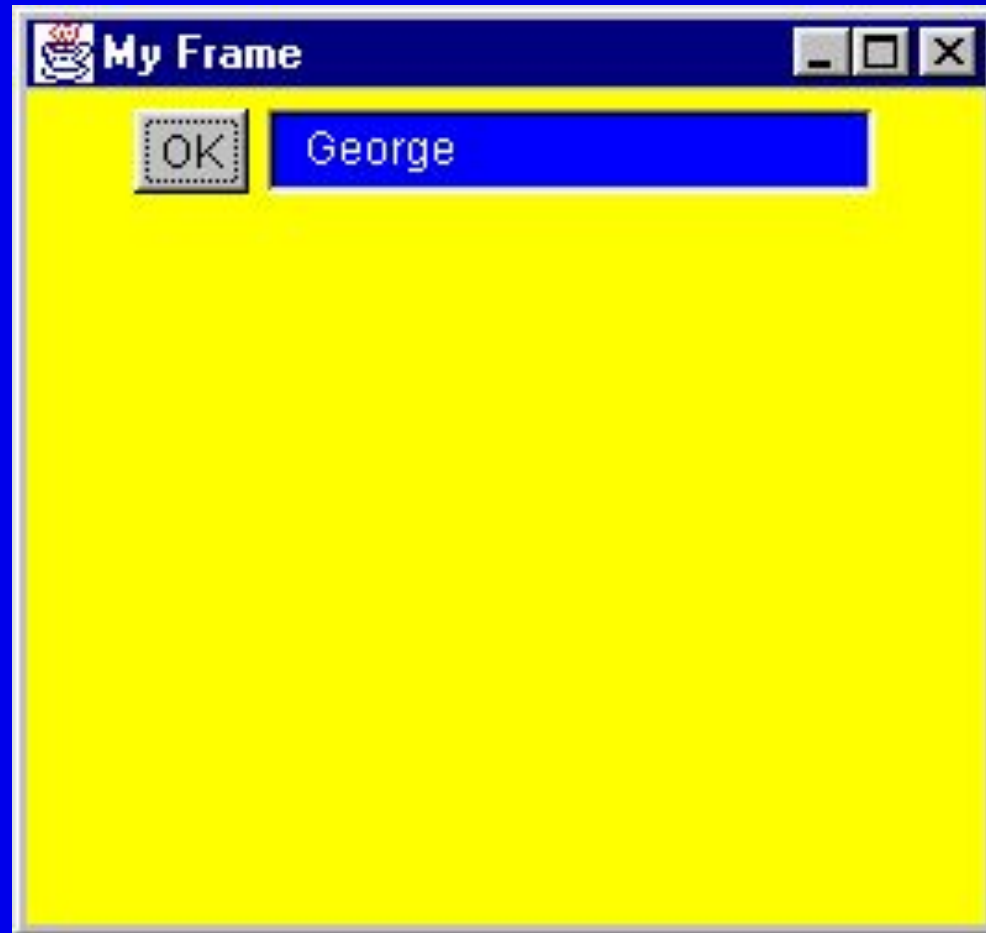
(c) Rajkumar

```
Import java.awt.*;
class MyGui {
    public static void main(String [] s ) {
        Frame f = new Frame (“My Frame”);
        Button b = new Button(“OK”);
        TextField tf = new TextField(“George”, 20);
f.setLayout(new FlowLayout());
        f.add(b);
        f.add(tf);
        f.setSize(300, 300);
        f.setVisible(true);
    }
}
```



output

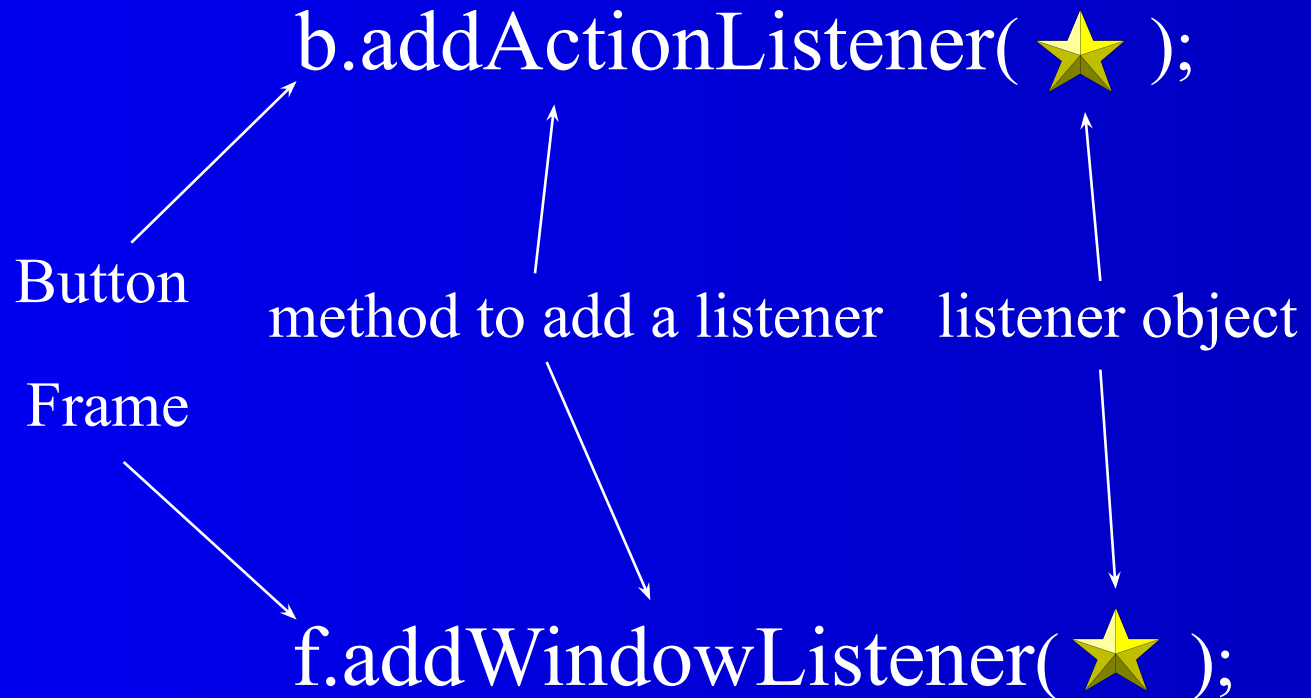
(c) Rajkumar





Events

(c) Rajkumar





(c) Rajkumar

- Each GUI component (e.g., a Button) that wishes to respond to an event type (e.g., click), must register an event handler, called a Listener.
- The listener is an object of a "Listener" interface.
- A Listener class can be created by subclassing (through "implements") one of Listener interfaces (all listener interfaces are in the java.awt.event package => must import java.awt.event.*;)
- The registration of the listener is done by a call to a method such as addActionListener(<Listener Object>). Each GUI component class has one or more such add...() methods, where applicable.



Listener Interfaces

(c) Rajkumar

INTERFACE NAME (IN JAVA.AWT.EVENT)

- [1] ActionListener
- [2] ItemListener
- [3] MouseMotionListener
- [4] MouseListener
- [5] KeyListener
- [6] FocusListener
- [7] AdjustmentListener
- [8] ComponentListener
- [9] WindowListener
- [10] ContainerListener
- [11] TextListener



Listener Interfaces

(c) Rajkumar

Each listener interface has methods that need to be implemented for handling different kinds of events.

For example, the MouseEventListener interface has two methods:

- 1) **mouseDragged(MouseEvent)** - Invoked when a mouse button is pressed on a component and then dragged.
- 2) **mouseMoved(MouseEvent)** - Invoked when the mouse button has been moved on a component (with no buttons down).



... the WindowListener interface has seven methods:

(c) Rajkumar

- 1) **windowActivated**(WindowEvent) - Invoked when a window is activated.
- 2) **windowClosed**(WindowEvent) - Invoked when a window has been closed.
- 3) **windowClosing**(WindowEvent) - Invoked when a window is in the process of being closed.
- 4) **windowDeactivated**(WindowEvent) - Invoked when a window is de-activated.
- 5) **windowDeiconified**(WindowEvent) - Invoked when a window is de-iconified.
- 6) **windowIconified**(WindowEvent) - Invoked when a window is iconified.
- 7) **windowOpened**(WindowEvent) - Invoked when a window has been opened.



How to create an object of a listener interface ?

(c) Rajkumar

Interfaces cannot be instantiated.

Therefore, cannot do new WindowListener():

Instead, have to subclass the interface and then create object of the subclass



Implementing the ActionListener Interface and attaching an event handler to a button

(c) Rajkumar

```
class MyApp implements ActionListener {
    Button b = new Button("OK");
    public static void main(String [] s ) {
        (new MyApp()).go();
    }
    public void go() {
        b.addActionListener( this );
    }
    public void actionPerformed(ActionEvent e ) {
        // what to do when the button is clicked
        if( e.getSource() == b )
        {   System.out.println("OK pressed"); }
    }
}
```



Implementing 2 interfaces

(c) Rajkumar

```
class MyApp implements ActionListener, WindowListener {
    Button b = new Button("OK");
    Frame f = new Frame("My Frame");
    public static void main(String [] s ) {(new MyApp()).go(); }
    public void      go() {
        b.addActionListener( this );
        f.addWindowListener( this );
    }
    public void      actionPerformed(ActionEvent e ) { ... }
    public void      windowActivated(WindowEvent e ) { ... }
    public void      windowClosed(WindowEvent e ) { ... }
    public void      windowClosing(WindowEvent e ) { ... }
    public void      windowDeactivated(WindowEvent e ) { ... }
    public void      windowDeiconified(WindowEvent e ) { ... }
    public void      windowIconified(WindowEvent e ) { ... }
    public void      windowOpened(WindowEvent e ) { ... }
}
```



or ... use Adapters

(c) Rajkumar

```
class MyApp extends WindowAdapter {  
    Button b = new Button("OK");  
    Frame f = new Frame("My Frame");  
    public static void main(String [] s ) {(new MyApp()).go(); }  
    public void go() {  
        f.addWindowListener( this );  
    }  
    public void windowClosing(WindowEvent e ) { ... }  
}
```

Need only implement the method(s) that are required,
instead of all seven methods of the WindowListener
interface



But, we can only use one Adapter at a time (no multiple inheritance)

(c) Rajkumar

I.e., cannot have :

```
class MyApp extends WindowAdapter,  
MouseAdapter, ... {  
    .....  
}
```



However ... can use inner classes instead !!!

(c) Rajkumar

```
class MyApp {
    Button b = new Button("OK");
    Frame f = new Frame("My Frame");
    public static void main(String [] s ) {
        ((new MyApp()).go()); }
    public void go() {
        f.addWindowListener( new FrameHandler() );
        b.addMouseListener( new ButtonHandler() );
    }
    class ButtonHandler extends MouseAdapter {
        public void mousePressed (MouseEvent e ) { ... }
    }
    class FrameHandler extends WindowAdapter {
        public void windowClosing (WindowEvent e ) { ... }
    }
}
```



Popup Menu and Event Handling...

(c) Rajkumar

```
//popup.java: popup menu and event handling
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;
public class popup extends Frame implements ActionListener, MouseListener
{
    TextField text1;
    PopupMenu popup;
    MenuItem menuitem1, menuitem2, menuitem3;
    public popup()
    {
        super( "Popup Menu" );
        setLayout(new FlowLayout());
        setBounds(10, 10, 300, 200 );
        setVisible(true);
        init();
    }
    public void init()
    {
        popup = new PopupMenu("Resource Usage" );
```



Popup Menu and Event Handling...

(c) Rajkumar

```
menuItem1 = new MenuItem("CPU");
menuItem1.addActionListener(this);
menuItem2 = new MenuItem("Disk");
menuItem2.addActionListener(this);
menuItem3 = new MenuItem("Memory");
menuItem3.addActionListener(this);
popup.add(menuItem1);
popup.add(menuItem2);
popup.add(menuItem3);
add(popup);
text1 = new TextField(20);
text1.setBounds(20, 40, 120, 30);
add(text1);
addMouseListener(this);
}
public void mousePressed(MouseEvent e)
{
    if( e.getModifiers() != 0 )
        popup.show(this, e.getX(), e.getY());
}
```




Popup Menu and Event Handling

(c) Rajkumar

```
public void mouseReleased( MouseEvent e )
{ System.out.print("Mouse Released\n" ); }
public void mouseEntered( MouseEvent e )
{ System.out.print("Mouse Entered\n" ); }
public void mouseExited( MouseEvent e )
{ System.out.print("Mouse Exited\n" ); }
public void actionPerformed( ActionEvent e )
{
    if( e.getSource() == menuitem1 )
    { text1.setText("CPU"); }
    if( e.getSource() == menuitem2 )
    { text1.setText("Disk"); }
    if( e.getSource() == menuitem3 )
    { text1.setText("Memory"); }
}
public static void main( String args[] )
{
    popup p = new popup();
}
}
```



(c) Rajkumar

Applets and GUI



(c) Rajkumar

AWT & Applets

An Applet is a Java program capable of running from within a web page (HTML document)

Steps to incorporate and run an applet:

- Have MyApplet.java
- javac MyApplet.java
- Have MyApplet.class
- Create **MyApplet.html**

```
<applet code = MyApplet.class width = 200 height = 300 >  
</applet>
```

- appletviewer MyApplet.html (or open MyApplet.html in browsers like Netscape/IE).



Applet methods

(c) Rajkumar

Unlike Applications, Applets do not have `main()`.

Instead, they have : `init()`, `start()`, `stop()`, `paint()`, `update()`, `repaint()`, `destroy()`.

All methods need not be implemented -- there are default versions for all of them.

- **AppletContext**
 - “Applet” derived from AWT Panel
 - Hooks into Browser environment
 - Can be used to link to another Web page



A sample Applet

(c) Rajkumar

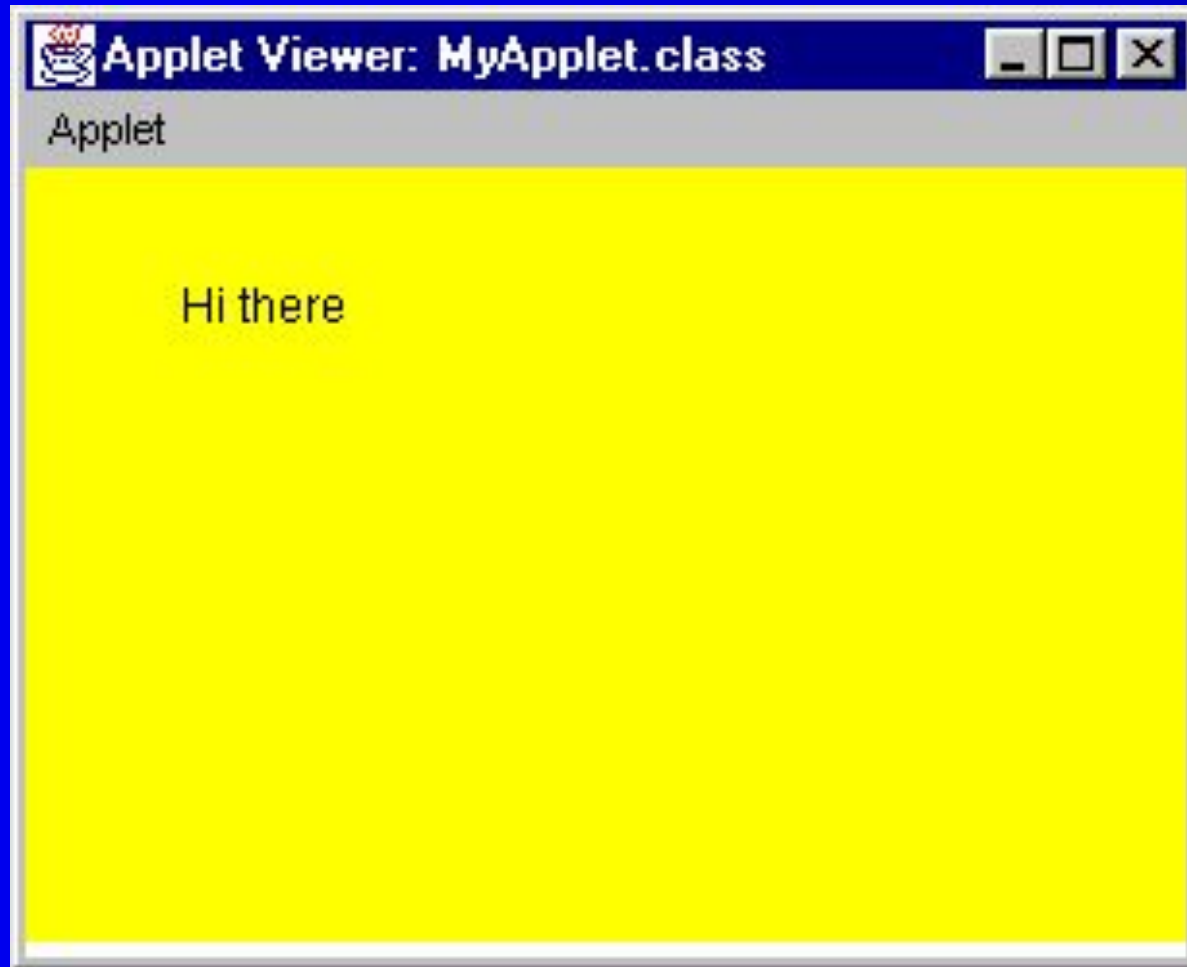
```
// HelloApplet.java: for processing applet methods
import java.awt.*;
import java.applet.*;
public class HelloApplet extends Applet
{

    public void init()    {
        setBackground(Color.yellow);
        System.out.println("init() method invoked");
    }
    public void start()
    {
        System.out.println("start() method invoked");
    }
    public void paint( Graphics g )
    {
        System.out.println("paint() method invoked");
        g.drawString( "Hi there", 24, 25 );
    }
    public void stop()
    {
        System.out.println("stop() method invoked");
    }
}
```



sample Applet

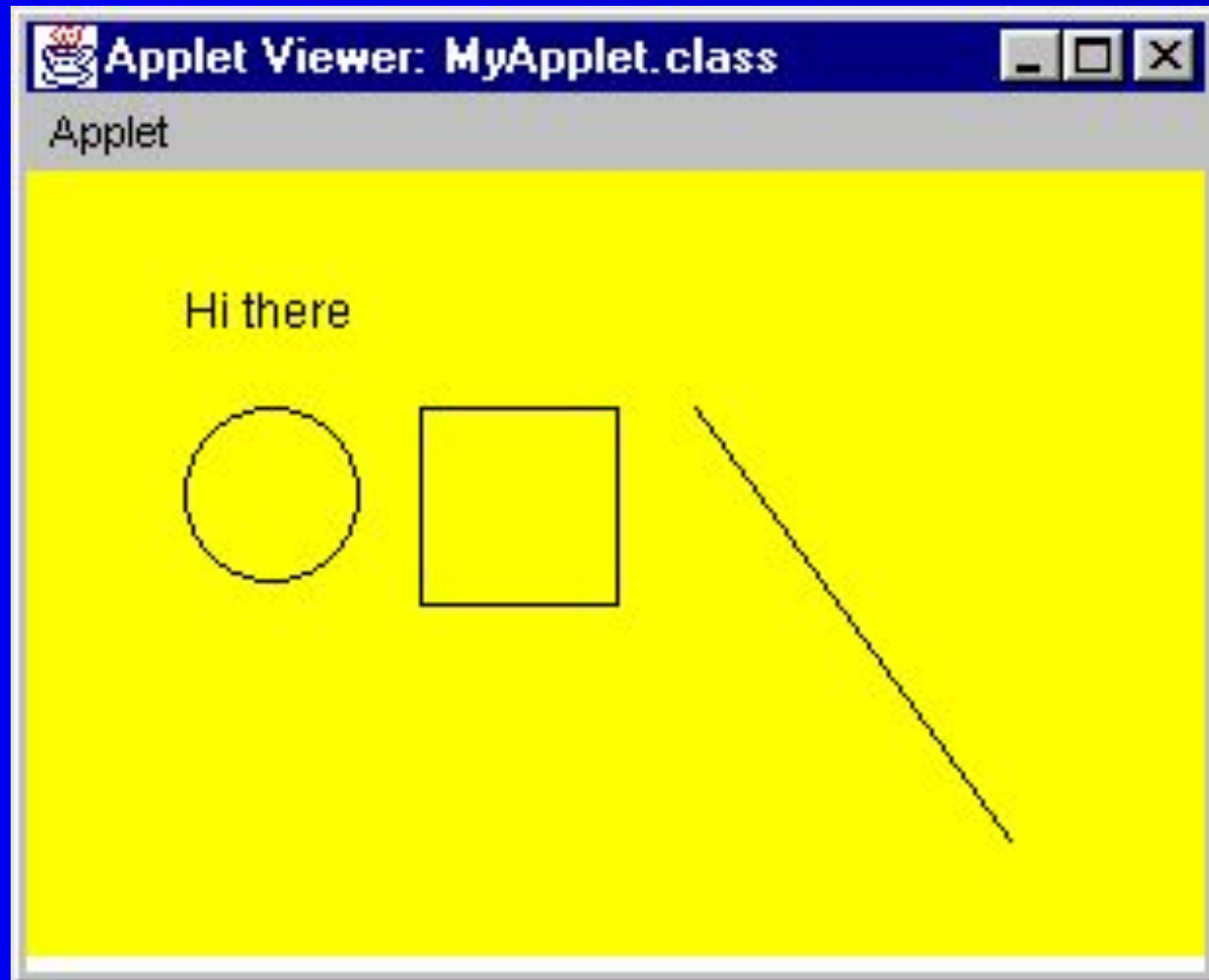
(c) Rajkumar





another sample Applet (run in Applet Viewer)

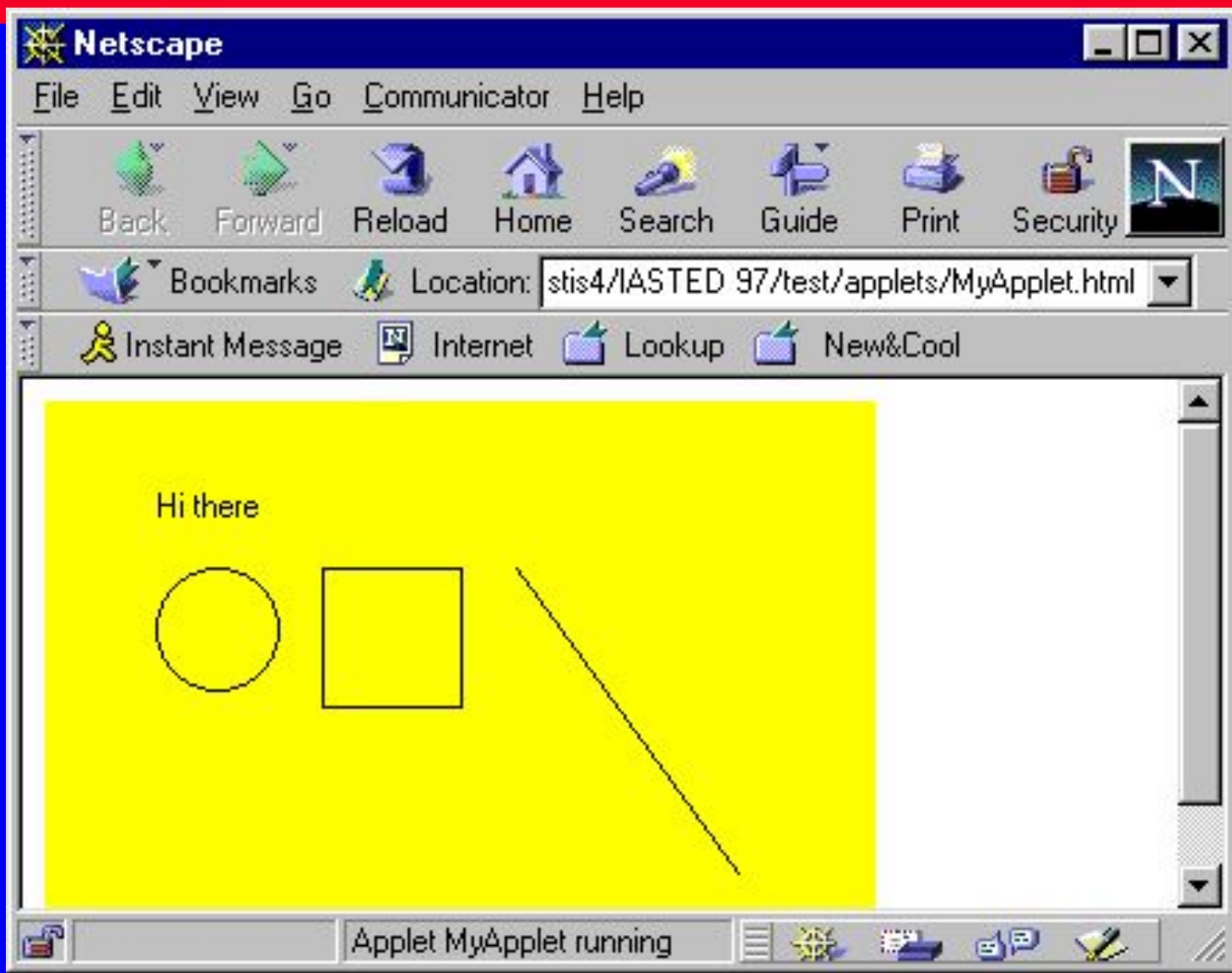
(c) Rajkumar





sample Applet running within Netscape

(c) Rajkumar





sample Applet code

(c) Rajkumar

```
import java.applet.*; // for Applet class
import java.awt.*;    // for Graphics class
public class MyApplet extends Applet {
    public void paint( Graphics g ) {
        g.drawString("Hi there", 40, 40);
        g.drawOval(40, 60, 45, 45);
        g.drawRect(100, 60, 50, 50);
        g.drawLine(170, 60, 250, 170);
    } // end paint()
    public void init() {
        setBackground(Color.yellow);
    }
} // end class MyApplet
```



Another example

(c) Rajkumar

```
// MyApplet.java: draws rectangle with yellow color fill
import java.applet.*;
import java.awt.*;
public class MyApplet extends Applet
{
    public synchronized void paint(Graphics g)
    {
        int x,y,width,height;
        Dimension dm = size();
        x = dm.width/4;
        y = dm.height / 4;
        width = dm.width / 2;
        height = dm.height / 2;
        // Draw the rectangle in the center with colors!
        g.setColor(Color.blue);
        g.drawRect(x,y,width,height);
        g.setColor(Color.yellow);
        g.fillRect(x + 1,y + 1,width - 2,height - 2);
    }
}
```



order of Applet method execution

(c) Rajkumar

As soon as the browser (or Appletviewer) accesses the page that contains the applet:

It calls `init()`, first

It calls `start()`, second.

It calls `paint()`, third.



order of Applet method execution (cont'd)

(c) Rajkumar

After the above three initial calls, invocation of the other methods depends on user's activity while in the browser:
no activity => none of the methods is invoked

leave to a different URL => stop() is invoked (and if later come back to this URL, then start() will be invoked).

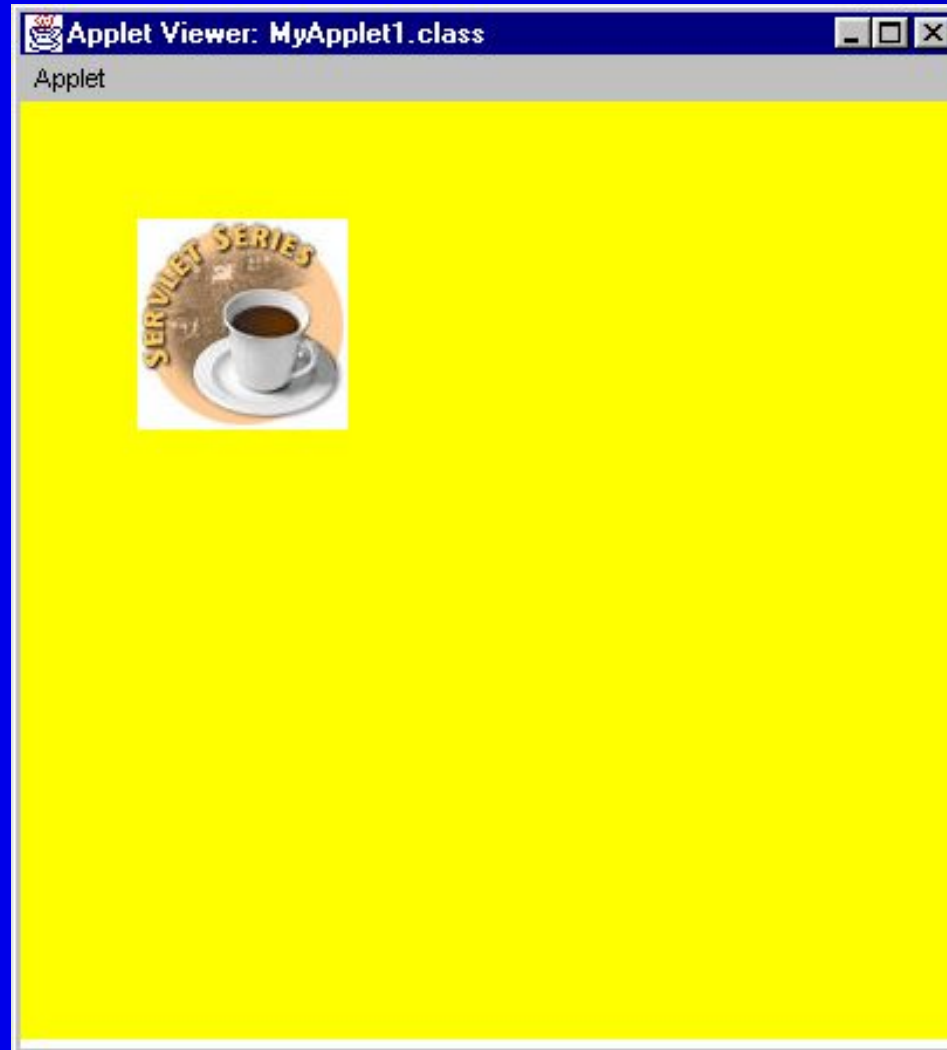
close down the browser => destroy() is invoked

none of the above => either paint() or update() or repaint() is invoked.



Incorporating Images and sound in Applets

(c) Rajkumar





sample Applet with sound

(c) Rajkumar

... .. (MyAppletSound.java)



how to do that

(c) Rajkumar

Step 1 : LOAD (image of sound file)

Step 2 : DISPLAY -or- PLAY



(c) Rajkumar

Applet that displays image

```
import java.applet.*;
import java.awt.*;
public class MyApplet1 extends Applet {
    Image im;
    public void init () {
        // load
        im = getImage(getDocumentBase(),"BOTTOMDOLLAR.JPG");
        setBackground(Color.yellow);
    }
    public void paint(Graphics g ) {
        g.drawImage(im, 50, 50, this); // display
    }
} // end class MyApplet1
```




(c) Rajkumar

Applet that plays sound

```
import java.applet.*;  
import java.awt.*;  
public class MyAppletSound extends Applet {  
    AudioClip ac;  
    public void init () {  
        // load  
        ac = getAudioClip(getDocumentBase(), "chirp1.au");  
    }  
    public void start() {  
        ac.loop();    // play  
    }  
    public void stop() {  
        ac.stop();    // stop the sound upon leaving this web page  
    }  
} // end class MyAppletSound
```



(c) Rajkumar

Multithreading in Java

(A built-in feature in Java)

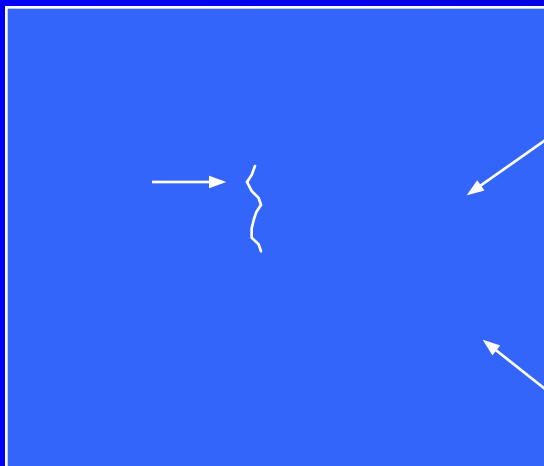


Single and Multithreaded Processes

(c) Rajkumar

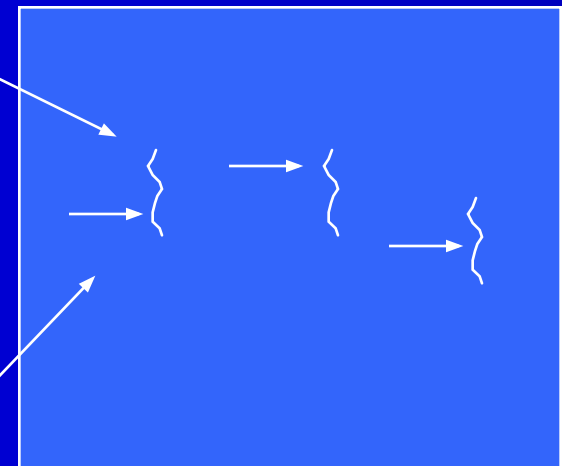
threads are light-weight processes within a process

Single-threaded Process



Single instruction stream

Multiplethreaded Process



Multiple instruction stream

Threads of Execution

Common Address Space



Threads

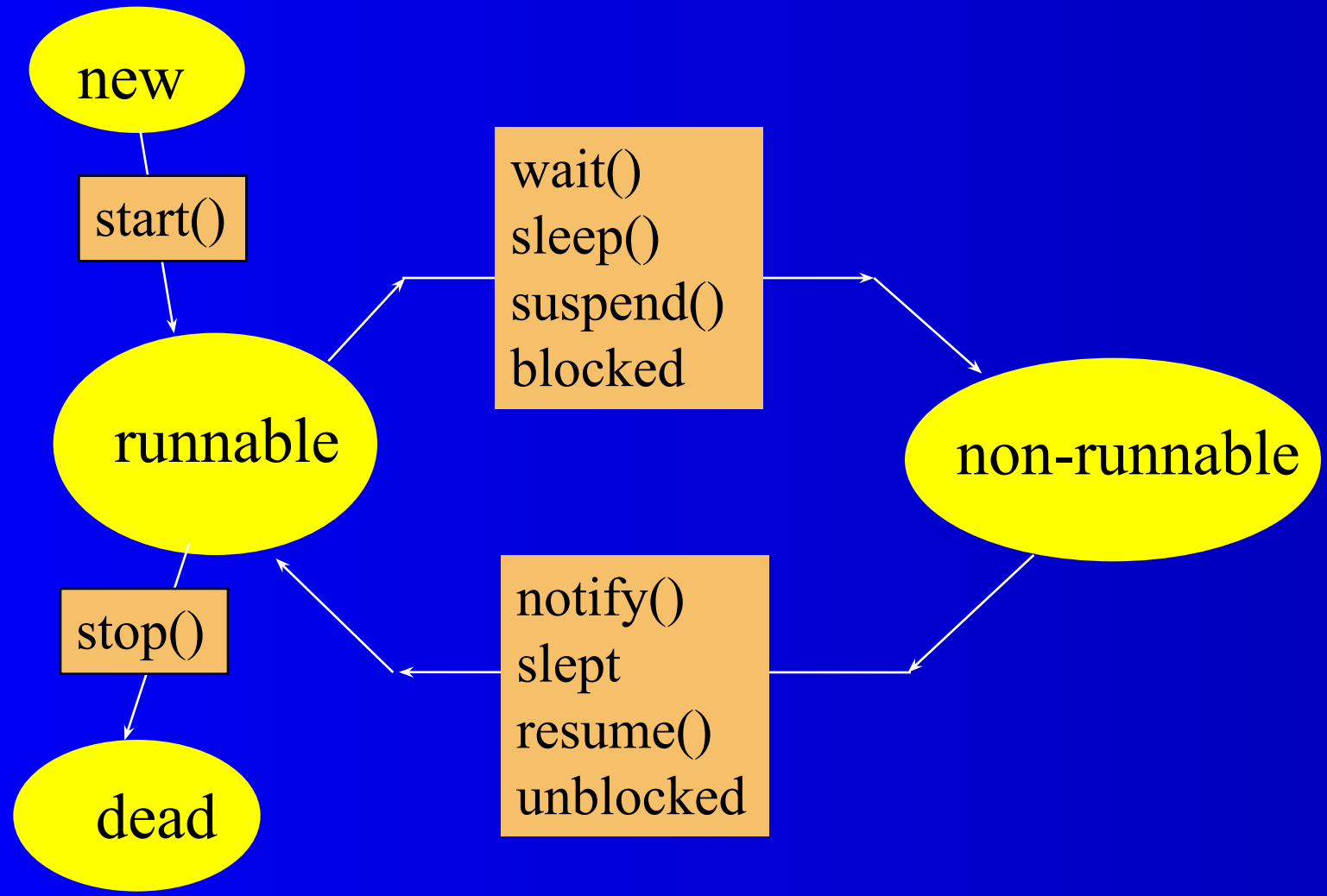
(c) Rajkumar

- Java has built in thread support for Multithreading
- Synchronization
- Thread Scheduling
- Inter-Thread Communication:
 - currentThread
 - start
 - setPriority
 - yield
 - run
 - getPriority
 - sleep
 - stop
 - suspend
 - resume
- Java Garbage Collector is a low-priority thread



Thread states

(c) Rajkumar

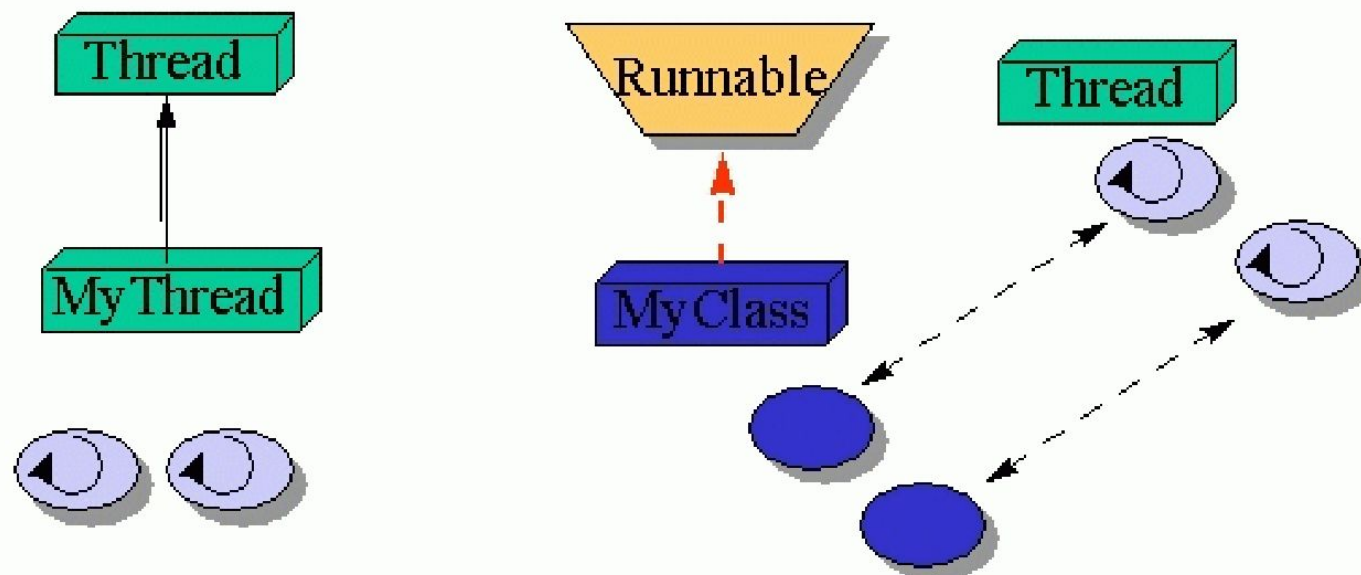




Threading Mechanisms...

- (c) Rajkumar
- Create a class that extends the Thread class
 - Create a class that implements the Runnable interface

Threading Mechanisms





1st method: Extending Thread class

(c) Rajkumar

1st Method: Extending the Thread class

```
class MyThread extends Thread
{
    public void run()
    {
        // thread body of execution
    }
}
```

- **Creating thread:**

```
MyThread thr1 = new MyThread();
```

- **Start Execution:**

```
thr1.start();
```



An example

(c) Rajkumar

```
class MyThread extends Thread { // the thread
    public void run() {
        System.out.println(" this thread is running ... ");
    }
} // end class MyThread

class ThreadEx2 { // a program that utilizes the thread
    public static void main(String [] args ) {
        // note, the created object myThreadObject IS A Thread as well.
        MyThread t = new MyThread();
        // due to extending the Thread class (above)
        // I can call start(), and this will call
        // run(). start() is a method in class Thread.
        t.start();
    } // end main()
} // end class ThreadEx2
```




2nd method: Threads by implementing Runnable interface

(c) Rajkumar `class MyThread implements Runnable`

```
{  
    .....  
    public void run()  
    {  
        // thread body of execution  
    }  
}
```

- **Creating Object:**

```
MyThread myObject = new MyThread();
```

- **Creating Thread Object:**

```
Thread thr1 = new Thread( myObject );
```

- **Start Execution:**

```
thr1.start();
```



An example

(c) Rajkumar

```
class MyThread implements Runnable {
    public void run() {
        System.out.println(" this thread is running ... ");
    }
} // end class MyThread

class ThreadEx21 {
    public static void main(String [] args ) {
        Thread t = new Thread(new MyThread());
        // due to implementing the Runnable interface
        // I can call start(), and this will call run().
        t.start();
    } // end main()
} // end class ThreadEx2
```



A program with two threads

```
(c) Rajkumar class MyThread implements Runnable {
    public void run() { System.out.println("This is 'MyThread' ); }
}

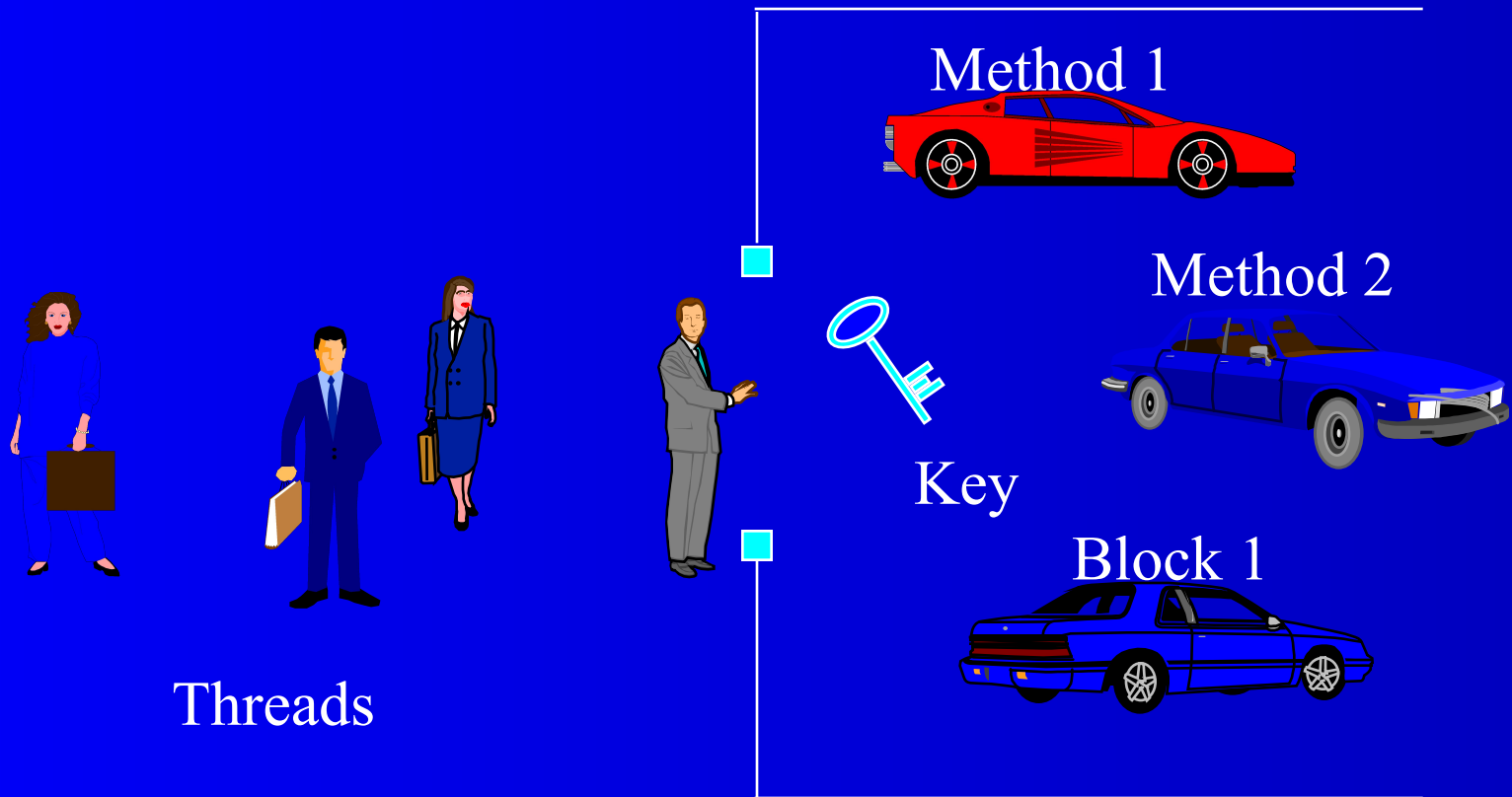
class YourThread implements Runnable {
    public void run() { System.out.println("This is 'YourThread'); }
}

class ThreadEx4 {
    public static void main(String [] args ) {
        Thread t1 = new Thread(new MyThread());
        Thread t2 = new Thread(new YourThread());
        t1.start();
        t2.start();
    }
} // end class ThreadEx4
```



Monitor model (for Synchronisation)

(c) Rajkumar



Monitor (synchronised) solves race-condition problem



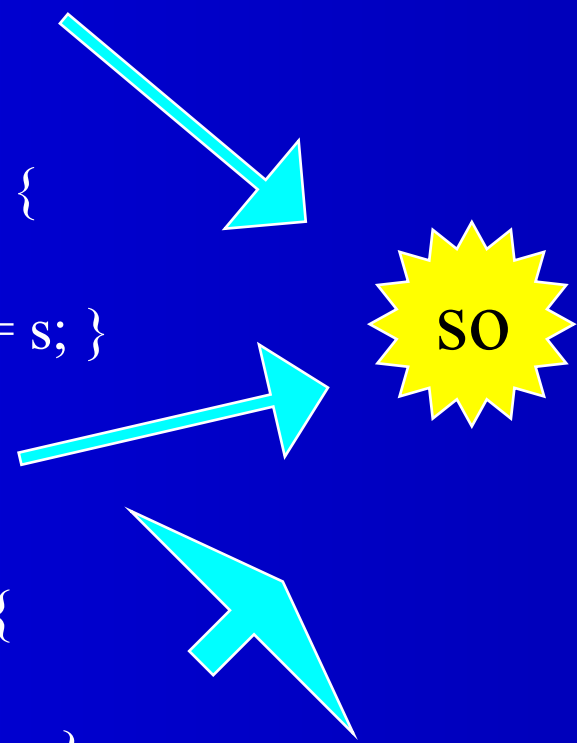
(c) Rajkumar

examples :: program with two threads and shared object

```
class MyThread implements Runnable {  
    Shared so;  
    public MyThread (Shared s) { so = s;}  
    public void run() { so.method1(); }  
} // end class MyThread
```

```
class YourThread implements Runnable {  
    Shared so;  
    public YourThread (Shared s) { so = s; }  
    public void run() { so.method2(); }  
} // end class YourThread
```

```
class HerThread implements Runnable {  
    Shared so;  
    public HerThread (Shared s) { so = s; }  
    public void run() { so.method3(); }  
} // end class HerThread
```





the monitor (shared object)

```
class Shared { // the 'monitor'
```

```
// if 'synchronized' is removed, the outcome is unpredictable
```

```
public synchronized void method1() {  
    for (int i = 0; i < 200; i++) { System.out.print(" [1] :: " + i ); }  
}
```

```
// if the 'synchronized' is removed, the outcome is unpredictable
```

```
public synchronized void method2() {  
    for (int i = 0; i < 200; i++) { System.out.print(" [2] :: " + i ); }  
}
```

```
// if the 'synchronized' is removed, the outcome is unpredictable
```

```
public synchronized void method3() {  
    for (int i = 0; i < 200; i++) { System.out.print(" [3] :: " + i ); }  
}
```

```
} // end class Shared
```

(c) Rajkumar



the driver

(c) Rajkumar

```
class MyMainClass {  
    public static void main(String [] args ) {  
        Shared sharedObject = new Shared ();  
        Thread t1 = new Thread(new MyThread(sharedObject));  
        Thread t2 = new Thread(new YourThread(sharedObject));  
        Thread t3 = new Thread(new HerThread(sharedObject));  
  
        t1.start();  
        t2.start();  
        t3.start();  
  
    } // end main()  
  
} // end class ThreadEx5
```

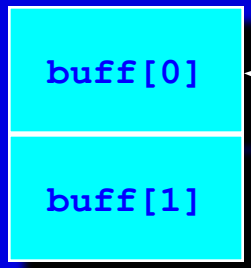


Threads in Action...

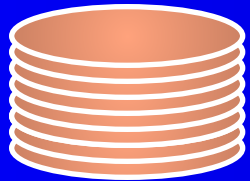
Cooperative threads - File Copy

(c) Rajkumar

```
reader ()  
{  
  -----  
  -  
  lock (buff [i]);  
  read (src ,buff [i]);  
  unlock (buff [i]);  
  -----  
  -  
}
```



```
writer ()  
{  
  -----  
  lock (buff [i]);  
  write (src ,buff [i]);  
  unlock (buff [i]);  
  -----  
}
```



**Cooperative Parallel Synchronized
Threads**



(c) Rajkumar

Streams and I/O



Streams and I/O

(c) Rajkumar

- **basic classes for file IO**
 - FileInputStream, for reading from a file
 - FileOutputStream, for writing to a file

- **Example:**

Open a file "myfile.txt" for **reading**

```
FileInputStream fis = new FileInputStream("myfile.txt");
```

Open a file "outfile.txt" for **writing**

```
FileOutputStream fos = new FileOutputStream("myfile.txt");
```



Display File Contents

(c) Rajkumar

```
import java.io.*;
public class FileToOut1 {
    public static void main(String args[]) {
        try {
            FileInputStream infile = new FileInputStream("testfile.txt");
            byte buffer[] = new byte[50];
            int nBytesRead;
            do {
                nBytesRead = infile.read(buffer);
                System.out.write(buffer, 0, nBytesRead);
            } while (nBytesRead == buffer.length);
        }
        catch (FileNotFoundException e) {
            System.err.println("File not found");
        }
        catch (IOException e) { System.err.println("Read failed"); }
    }
}
```



(c) Rajkumar

- Once a stream (e.g., file) has been opened, we can attach filters
- Filters make reading/writing more efficient
- Most popular filters:
 - For basic types:
 - `DataInputStream`, `DataOutputStream`
 - For objects:
 - `ObjectInputStream`, `ObjectOutputStream`



Writing data to a file using Filters

(c) Rajkumar

```
import java.io.*;
public class GenerateData {
    public static void main(String args[]) {
        try {
            FileOutputStream fos = new FileOutputStream("stuff.dat");
            DataOutputStream dos = new DataOutputStream(fos);
            dos.writeInt(2);
            dos.writeDouble(2.7182818284590451);
            dos.writeDouble(3.1415926535);
            dos.close(); fos.close();
        }
        catch (FileNotFoundException e) {
            System.err.println("File not found");
        }
        catch (IOException e) {
            System.err.println("Read or write failed");
        }
    }
}
```



Reading data from a file using filters

(c) Rajkumar

```
import java.io.*;
public class ReadData {
    public static void main(String args[]) {
        try {
            FileInputStream fis = new FileInputStream("stuff.dat");
            DataInputStream dis = new DataInputStream(fis);
            int n = dis.readInt();
            System.out.println(n);
            for( int i = 0; i < n; i++ ) { System.out.println(dis.readDouble());
            }
            dis.close(); fis.close();
        }
        catch (FileNotFoundException e) {
            System.err.println("File not found");
        }
        catch (IOException e) { System.err.println("Read or write failed");
        }
    }
}
```



Object serialization

(c) Rajkumar

Write objects to a file, instead of writing primitive types.

Use the **ObjectInputStream**, **ObjectOutputStream** classes, the same way that filters are used.



(c) Rajkumar

Write an object to a file

```
import java.io.*;
import java.util.*;
public class WriteDate {
    public WriteDate () {
        Date d = new Date();
        try {
            FileOutputStream f = new FileOutputStream("date.ser");
            ObjectOutputStream s = new ObjectOutputStream (f);
            s.writeObject (d);
            s.close ();
        }
        catch (IOException e) { e.printStackTrace(); }

    public static void main (String args[]) {
        new WriteDate ();
    }
}
```




Read an object from a file

(c) Rajkumar

```
import java.util.*;
public class ReadDate {
    public ReadDate () {
        Date d = null;
        ObjectInputStream s = null;
        try { FileInputStream f = new FileInputStream ("date.ser");
            s = new ObjectInputStream (f);
        } catch (IOException e) { e.printStackTrace(); }
        try { d = (Date)s.readObject (); }
        catch (ClassNotFoundException e) { e.printStackTrace(); }
        catch (InvalidClassException e) { e.printStackTrace(); }
        catch (StreamCorruptedException e) { e.printStackTrace(); }
        catch (OptionalDataException e) { e.printStackTrace(); }
        catch (IOException e) { e.printStackTrace(); }
        System.out.println ("Date serialized at: "+ d);
    }
    public static void main (String args[]) { new ReadDate (); }
}
```



(c) Rajkumar

Network/Socket Programming in Java



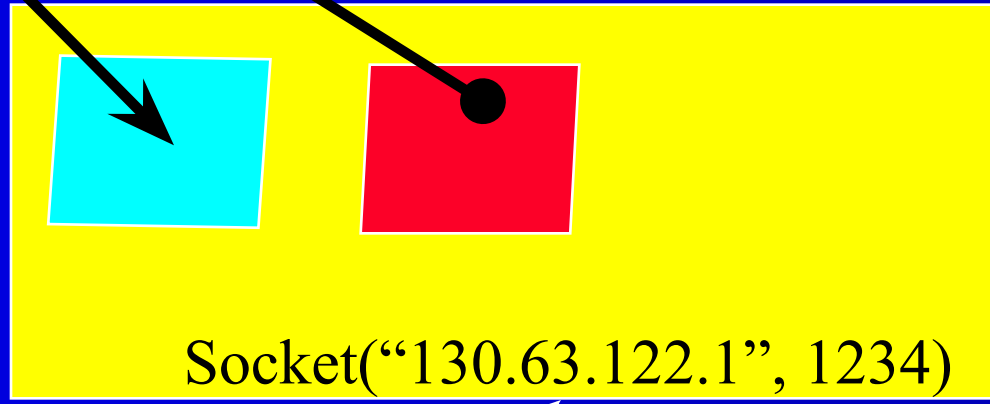
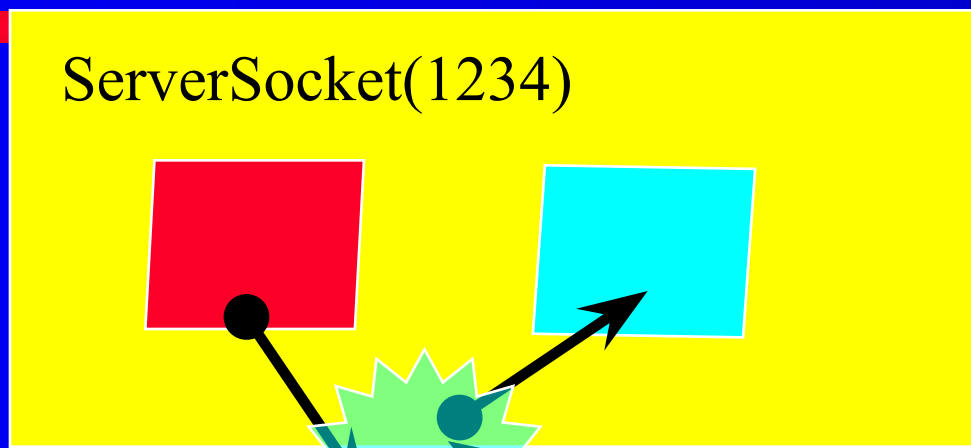
(c) Rajkumar



- **Used to manage:**
 - **URL streams**
 - **Client/server sockets**
 - **Datagrams**



Part III - Networking

(c) Rajkumar



 Output/write stream
 Input/read stream

Server_name: "cdacb.ernet.in" ↗

Socket("130.63.122.1", 1234)



Server side Socket Operations

(c) Rajkumar

1. Open Server Socket:

```
String server; Socket s;
DataOutputStream os;
DataInputStream is;
server = new ServerSocket( PORT );
```

2. Wait for Client Request:

```
Socket client = server.accept();
```

3. Create I/O streams for communicating to clients

```
is = new DataInputStream( client.getInputStream() );
os = new DataOutputStream( client.getOutputStream() );
```

4. Perform communication with client

```
Receive from client: String line = is.readLine();
Send to client: os.writeBytes("Hello\n");
```

5. Close sockets: client.close();

For multithreaded server:

```
while(true) {
    i. wait for client requests (step 2 above)
    ii. create a thread with "client" socket as parameter (the thread creates streams (as in step (3) and does communication as stated in (4). Remove thread once service is provided.
}
```



Client side Socket Operations

(c) Rajkumar

1. Get connection to server:

```
client = new Socket( server, port_id );
```

2. Create I/O streams for communicating to clients

```
is = new DataInputStream( client.getInputStream() );
```

```
os = new DataOutputStream( client.getOutputStream() );
```

3. Perform communication with client

```
Receive from client: String line = is.readLine();
```

```
Send to client: os.writeBytes("Hello\n");
```

4. Close sockets: client.close();



A simple server (simplified code)

(c) Rajkumar

```
import java.net.*;
import java.io.*;
public class ASimpleServer {
    public static void main(String args[]) {
        // Register service on port 1234
        ServerSocket s = new ServerSocket(1234);
        Socket s1=s.accept(); // Wait and accept a connection
        // Get a communication stream associated with the socket
        OutputStream s1out = s1.getOutputStream();
        DataOutputStream dos = new DataOutputStream (s1out);
        // Send a string!
        dos.writeUTF("Hi there");
        // Close the connection, but not the server socket
        dos.close();
        s1out.close();
        s1.close();
    }
}
```



A simple client (simplified code)

(c) Rajkumar

```
import java.net.*;
import java.io.*;
public class SimpleClient {
    public static void main(String args[]) throws IOException {
        // Open your connection to a server, at port 1234
        Socket s1 = new Socket("130.63.122.1",1234);
        // Get an input file handle from the socket and read the input
        InputStream s1In = s1.getInputStream();
        DataInputStream dis = new DataInputStream(s1In);
        String st = new String (dis.readUTF());
        System.out.println(st);
        // When done, just close the connection and exit
        dis.close();
        s1In.close();
        s1.close();
    }
}
```




Echo Server Client..

(c) Rajkumar

```
//client.java: client interface to server
import java.io.*;
import java.net.*;
public class client
{
    int port_id;
    String server; Socket slink;
    DataOutputStream os;
    DataInputStream is;
    DataInputStream kbd;
    public client( String args[] )
    {
        server = args[0];
        port_id = Integer.valueOf(args[1]).intValue();
        try
        {
            slink = new Socket( server, port_id );
            os = new DataOutputStream( slink.getOutputStream() );
            is = new DataInputStream( slink.getInputStream() );
            kbd = new DataInputStream( System.in );
        }
    }
}
```



Echo Server Client..

(c) Rajkumar

```
catch( UnknownHostException e )
{
    System.err.println( "Don't know about host: " );
    System.exit(1);
}
catch( IOException e )
{
    System.err.println( "Could not get I/O for the connection to "+server);
    System.exit(1);
}
}
}
void communicate()
{
    while(true)
    {
        try {
            System.out.print("Enter Input <end to stop>: ");
            String line = kbd.readLine();
            os.writeBytes( line+"\n" );
        }
    }
}
```



Echo Server Client..

(c) Rajkumar

```
if( line.equals("end") )
{
    os.close(); is.close(); slink.close();
    break;
}
String line2 = is.readLine();
System.out.println("Output: "+line2);
}
catch( IOException e )
{
    System.out.println(e); }
}
public static void main( String [] args )
{
    if( args.length < 2 )
    {
        System.out.println("Usage: java client server_name port_id" );
        System.exit(1);
    }
    client cln = new client( args );
    cln.communicate();
}
}
```



Echo Server ...

(c) Rajkumar

```
// server.java: echo server
import java.io.*;
import java.net.*;
public class server
{
    // public final static int PORT = 4779;
    public static void main( String [] args )
    {
        ServerSocket server = null;
        DataOutputStream os = null;
        DataInputStream is = null;
        boolean shutdown = false;
        if( args.length < 1 )
        {
            System.out.println( "Usage: java server port_num" );
            System.exit( 1 );
        }
        int PORT = Integer.valueOf(args[0]).intValue();
        try {
            server = new ServerSocket( PORT );
        }
```



Echo Server ...

(c) Rajkumar

```
catch( IOException e )
{
    System.err.println( "Could not get I/O for the connection to: " );
}
while(!shutdown)
{
    if( server != null )
    {
        try
        {
            Socket client = server.accept();
            System.out.println("Connected");
            InetAddress cip = client.getInetAddress();
            System.out.println( "Client IP Addr: "+cip.toString());
            is = new DataInputStream( client.getInputStream() );
            os = new DataOutputStream( client.getOutputStream() );
            for(;;)
            {
                String line = is.readLine();
                if( line == null )
                    break;
            }
        }
    }
}
```



Echo Server ...

```
(c) Rajkumar  if( line.startsWith("end" ) )
                {
                    shutdown = true;
                    break;
                }
                os.writeBytes(line.toUpperCase());
                os.writeBytes("\n");
                System.out.println(line);
            }
            is.close(); client.close();
        }
        catch( UnknownHostException e )
        {
            System.err.println( "Server Open fails" );
        }
        catch( IOException e )
        {
            System.err.println( "Could not get I/O for the connection to:"+args[0]);
        }
    }
}
```



Echo Server

(c) Rajkumar

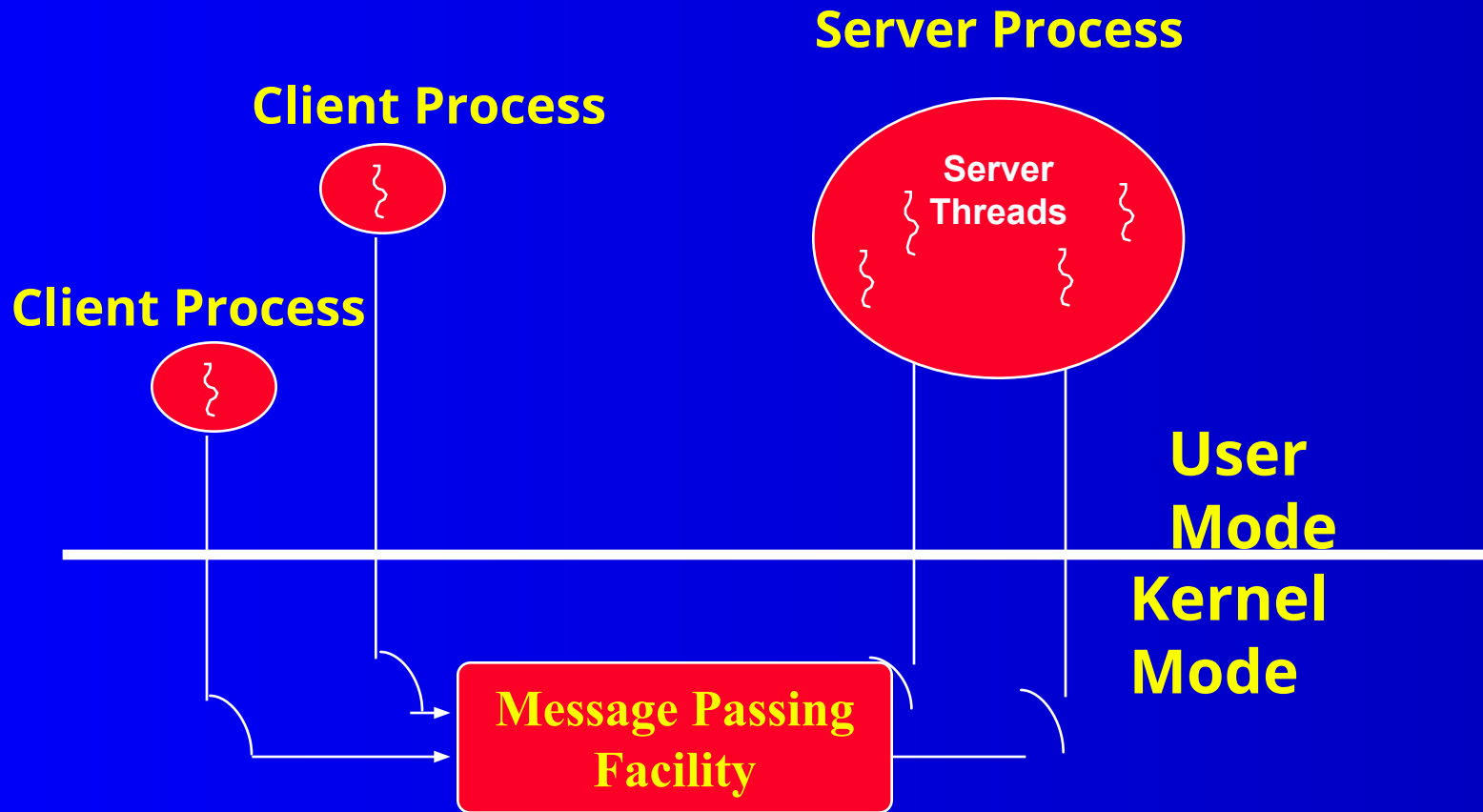
```
System.out.println( "Server Down" );  
    try {  
        server.close();  
    } catch(IOException e) {}  
}  
}
```



Threads in Action...

Multithreaded Server

(c) Rajkumar





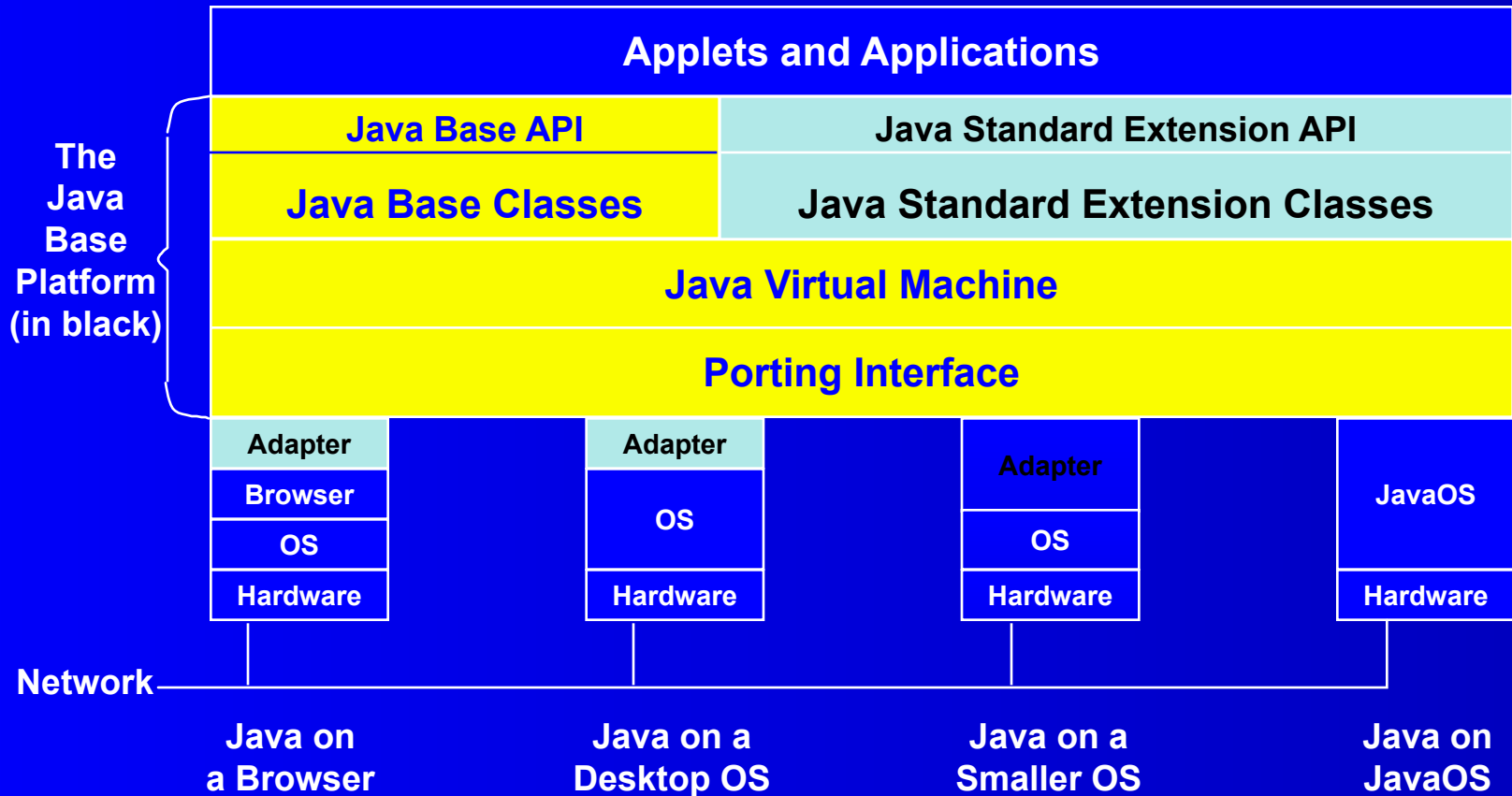
(c) Rajkumar

Java System Architecture & Availability



A Look Inside the Java Platform

(c) Rajkumar





Java Applications!

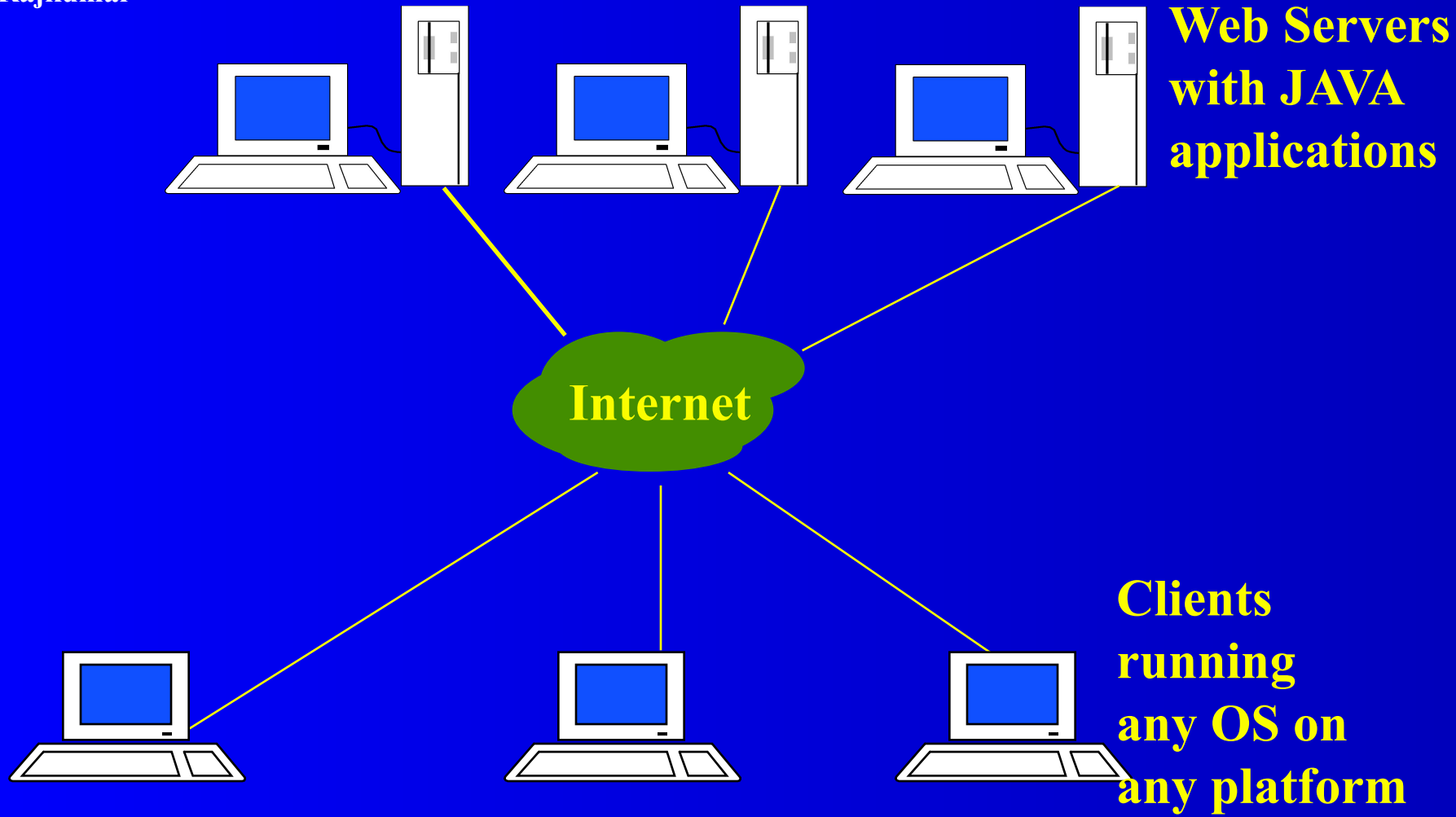
(c) Rajkumar

- Java applications are now available
- Cost of manfg zero, cost of distribution zero, cost of marketing zero!
- Hot Java is lean - loads everything else from the net.
- Java itself is small - 40k to 225k
- New class of small machines will emerge
- Java on cellular phones, credit cards, washing machines, and everywhere ?



Universal Interface

(c) Rajkumar





Java on my platform ?

(c) Rajkumar

- Sun (SPARC) <ftp://java.sun.com>
- Sun(x86) <ftp://xm.com:/pub/>
- IBM(Aix, OS/2) <ftp://ncc.hursley.ibm.com/javainfo>
- DEC(Alpha OSF/1)
<http://www.gr.osf.org:8001/projects/web/java/>
- SGI
<http://liawww.epfl.ch/~simon/java/irix-jdk.html>
- HP <http://www.gr.osf.org:8001/projects/web/java>
- Linux <http://www.blackdown.org>
- AT & T
<http://www.gr.osf.org:8001/projects/web/java>
- Windows 3.1 <http://www.alphaworks.ibm.com>



(c) Rajkumar

Java Development Tools (Present and Planned)



Sun's Java WorkShop

(c) Rajkumar

- **JDK:**

- **Compiler and runtime environment**
- **Class Libraries**
- **Documentation**
- **javadoc - Automated Documentation**
 - **Takes comments and converts to HTML**
- **IDE: Visual Java, and integrated tools, JavaBeans**
- **Other Products and API: JavaHelp, Java Card, Java Blend, JavaOS, Java Mail, Java Management, Java Electronic Commerce Framework**
- **Java Enterprise API: Java Naming and Directory Interface, Java IDL, JDBC, RMI and Object Serialization**



Symantec Cafe 1.0 (Released)

(c) Rajkumar

- Full IDE for Windows 95/NT
- Graphic Development Tools
- Two Compilers
- Debugger
- Class Browser



Microsoft Jakarta (Planned)

(c) Rajkumar

- **Visual C++ type interface**
- **Will Support ActiveX/COM**
- **Internet Explorer 3.0 will have Just-In-Time Java compiler**



Borland JBuilder

(c) Rajkumar

- Visual RAD workbench for maximum productivity.
- Rapid Application Development (RAD) and Open Component Architecture patterned after Delphi.
- 100+ JavaBean components, with source code, for drag-and-drop applications.
- Beans Express -- easiest way to create industry-standard Java-Bean components.
- DataExpress -- the fastest way to build business and database applications.
- Borland DataGateway for Java connectivity to all major database servers.
- Multi-tier applications with integrated RMI and CORBA.
- Versions: Standard, Professional, & Client/Server



Challenges & Possible Directions

(c) Rajkumar

- Performance
- AWT - need better GUI!
- Maintaining Interoperability
- Security - current restrictions limit what can be done
- Native Compilers
- Is Portability that Important?



Comments

(c) Rajkumar

- Java is a fun and easy programming language
- Portability = Mediocrity?
- Java will become a programming language of choice, but may take on a final form that will surprise many!



(c) Rajkumar

- **Java API for Relational Databases**
- **Being standardized by all major players**



Javascript and Java (Preview)

(c) Rajkumar

- Javascript can control Java applets
- Static data accessible as properties of applet
- `var i = Bank.Account.count`
- Public methods invocable on Java instances
- Provided those instances are accessible through the Applet
- Applet is accessible through document
- `document.applet_Name_Attribute.do_Stuff()`



Java for HPC!

(c) Rajkumar

- Many efforts are in progress for making java as a language for parallel programming.
- Java computing frameworks (HPC, numeric, data parallel)
- Java in distributed simulations and applications (e.g., real-world HPC, grand challenge)
- Source to source translators (C, Fortran, C++ to Java)
- Web based computation environment in Java
- Java for HPC conference:
<http://www.cs.ucsb.edu/conferences/java98>
- Java for Science and Engineering computing:
subscribe java-for-cse to majordomo@npac.syr.edu
- <http://www.jhpc.org>



How to Convert Programs to AWT 1.1...

(c) Rajkumar

1. Change source code so that it import event package:

```
import java.awt.event.*;
```

2. Figure out which component generates each event type: (1.0 uses `handleEvent()` and `action()`)

Button, List, MenuItem, TextField:

Interface: ActionListener

Method: `actionPerformed(ActionEvent event)`

Checkbox, CheckboxMenuItem, Choice:

Interface: ItemListener **Method:** `itemStateChanged(..)`

Dialog, Frame:

Interface: WindowListener

Method: `windowClosing()`, `windowOpened()`,...



How to Convert Programs to AWT 1.1...

(c) Rajkumar

**3. Change class declaration so that class implements
public class MyClass extends SomeComponent
implements ActionListener**

4. Register action Listener

```
newComponentObject.addActionListener(this);
```

5. Change event handling method:

Old: public boolean action(Event e, Object arg)

New: public void actionPerformed(ActionEvent e)



How to Convert Programs to AWT 1.1

(c) Rajkumar

- 6. Delete the event handling code in this way**
 - (a) Delete all return statements**
 - (b) Change `e.target` to `e.getSource()`**
 - (c) Delete all code the unnecessarily tests for which component the event come from**
 - (d) Perform any other modification require to make the program compile**



Just to Summarize

(c) Rajkumar

- Java as a Comprehensive Programming Solution
 - Object Oriented
 - Portable
 - High Performance
 - Geared for Distributed Environments
 - Secure
 - Highly suitable for Internet programming



Summary

(c) Rajkumar

- **Java is really very well poised**
- **Incredible leverage from the Web**
- **Will impact the C++ and Smalltalk markets**
- **Rate of progress is astonishingly high**
 - **Development environments**
 - **CORBA linkages**
 - **Components**
- **Fasten you seat-belts!**



(c) Rajkumar

Thank You ...

