15.561
Information Technology Essentials

Session 5
Programming Languages

Acknowledgments:
Slides marked "SM" are adapted from Stuart Madnick, MIT.

Outline

- Types of software
- Types of programming languages
- Examples
  - Java
  - Y2K

Types of software

- System software
  - Operating systems
  - Programming languages
  - Database systems

- Application software
  - General office tasks (word processing, etc.)
  - Accounting
  - Design
  - Factory automation
  - …

Programming languages

- Machine language
- Assembly language
- High-level languages
- Fourth-generation languages

A sample LMC program

INSTRUCTIONS

op-code	symbolic
000	**stop**
1xx	**add**
2xx	**subtract**
3xx	**store**
5xx	**load**
901	**get**
902	**put**

① ASSEMBLY LANG → ② MACHINE LANG.

(Source Program) (Object Program)

STEP	INSTRUCTION	LOC	INSTRUCTION
00	get	00	901
01	store w	01	398
02	get	02	901
03	store b	03	399
04	load w	04	598
05	add b	05	199
06	put	06	902
07	stop	07	000

Source: S. Madnick, MIT

Machine language

- Binary
- Machine dependent
- Stored in the computer when the program is running
- Example:

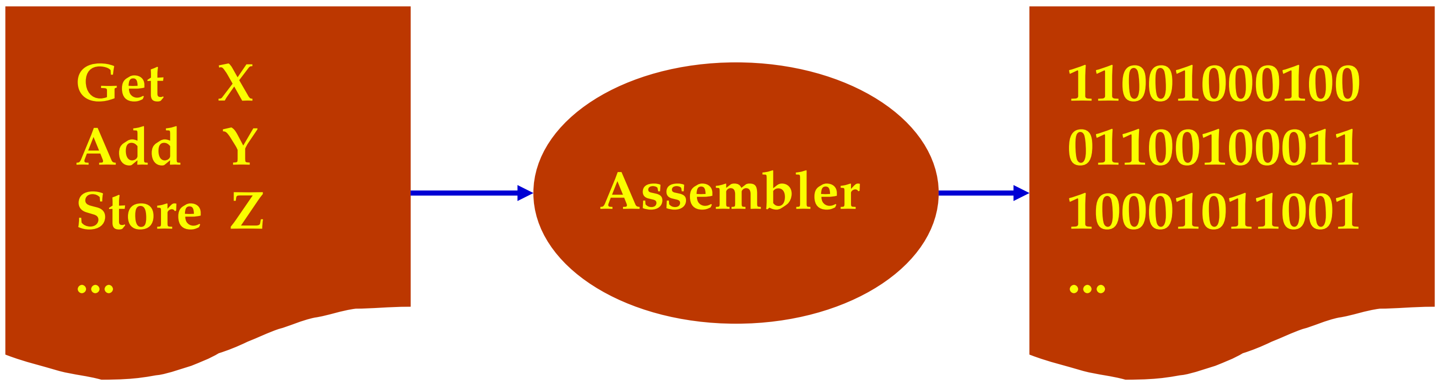
  0111011000101001001 0 ….

Assembly language

- Mnemonic
- Symbolic addressing
- One-to-one correspondence with machine language
- Example:

```
Get X
Add Y
Store Z
```

Automatically translating assembly language to machine language



Assembly language program ("source code")

Machine language program ("object code")

High-level languages

- Closer to how people think about their problems
- No one-to-one correspondence to machine language
- General purpose
- Example:

  Z = X + Y

High-level languages - Examples

- Fortran
- Basic
- Visual Basic
- C
- C++
- Java

Example 1 – Basic

```
'AVERAGING INTEGERS ENTERED THROUGH THE KEYBOARD
CLS
PRINT "THIS PROGRAM WILL FIND THE AVERAGE OF INTEGERS YOU ENTER"
PRINT "THROUGH THE KEYBOARD. TYPE 999 TO INDICATE THE END OF DATA."
PRINT
SUM=0
COUNTER =0
PRINT "PLEASE ENTER A NUMBER"
INPUT NUMBER
DO WHILE NUMBER <> 999
    SUM=SUM+NUMBER
    COUNTER=COUNTER+1
    PRINT "PLEASE ENTER THE NEXT NUMBER"
    INPUT NUMBER
LOOP
AVERAGE=SUM/COUNTER
PRINT "THE AVERAGE OF THE NUMBER IS"; AVERAGE
END
```

Example 2 – C++

```
// AVERAGING INTEGERS ENTERED THROUGH THE KEYBOARD

#include <iostream.h>
main ( )
{
     float average;
     int number, counter = 0;  int sum = 0;
     cout << “THIS PROGRAM WILL FIND THE AVERAGE OF INTEGERS YOU ENTER \n”;
     cout << “THROUGH THE KEYBOARD. TYPE 999 TO INDICATE END OF DATA. \n”;
     cout << “PLEASE ENTER A NUMBER”;
     cin   >> number;
     while (number !=999)
         {

               sum =sum + number;
               counter++;
               cout << “\nPLEASE ENTER THE NEXT NUMBER”;
               cin   >>number;
         }

      average = sum / counter;
      cout << “\nTHE AVERAGE OF THE NUMBERS IS “<< average;

}
```

Example 3 – Java

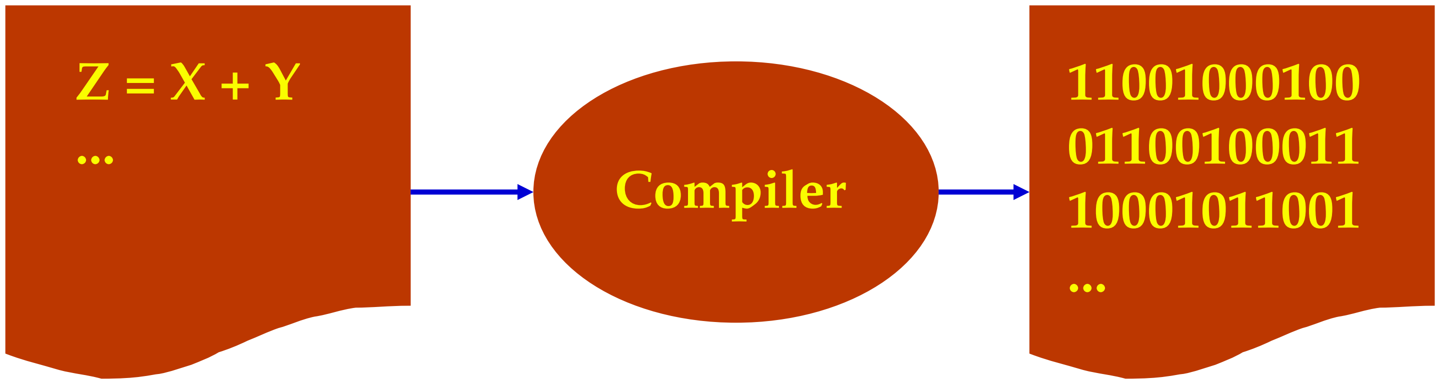
```
import java.io.*;
import java.lang.*;
/**
 ** Prompts user for list of numbers and outputs average
**/

class AverageNumbers {
   public static void main (String[] args) {
     float sum = 0;
     float average = 0;
     int counter = 0;

     System.out.println("THIS PROGRAM WILL FIND THE
     AVERAGE OF THE INTEGERS YOU ENTER
     THROUGH THE KEYBOARD.  TYPE 999 TO
     INDICATE END OF DATA.");

     try
     {
        BufferedReader in = new BufferedReader(new
     InputStreamReader(System.in));
        String cin = "0";
        System.out.println("Please enter a number.");
        while (!(cin=in.readLine()).equals("999"))
              {
                 sum = sum + Integer.parseInt(cin);
                 counter = counter + 1;
                 System.out.println("Please enter another
     number.");
              }
        in.close();
        average = sum/counter;
        System.out.println("The average of the numbers is :
     "+average);
     }
     catch (IOException e)
     {
              System.out.println("Ooops..");
     }
   }
}
```

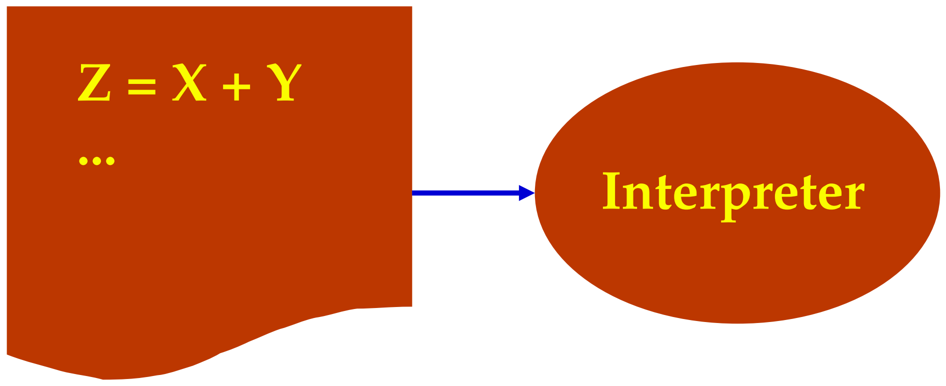
Automatically translating high-level language to machine language



High-level language program ("source code")

Machine language program ("object code")

“Interpreting” high level languages



High-level language program (“source code”)

Interpreting high level languages

- Advantages
  - Can give machine independence
    - (e.g., one machine can “look” like another)
  - Can be easier to debug and modify
  - Can give more flexibility at “run time”
- Disadvantages
  - Slower

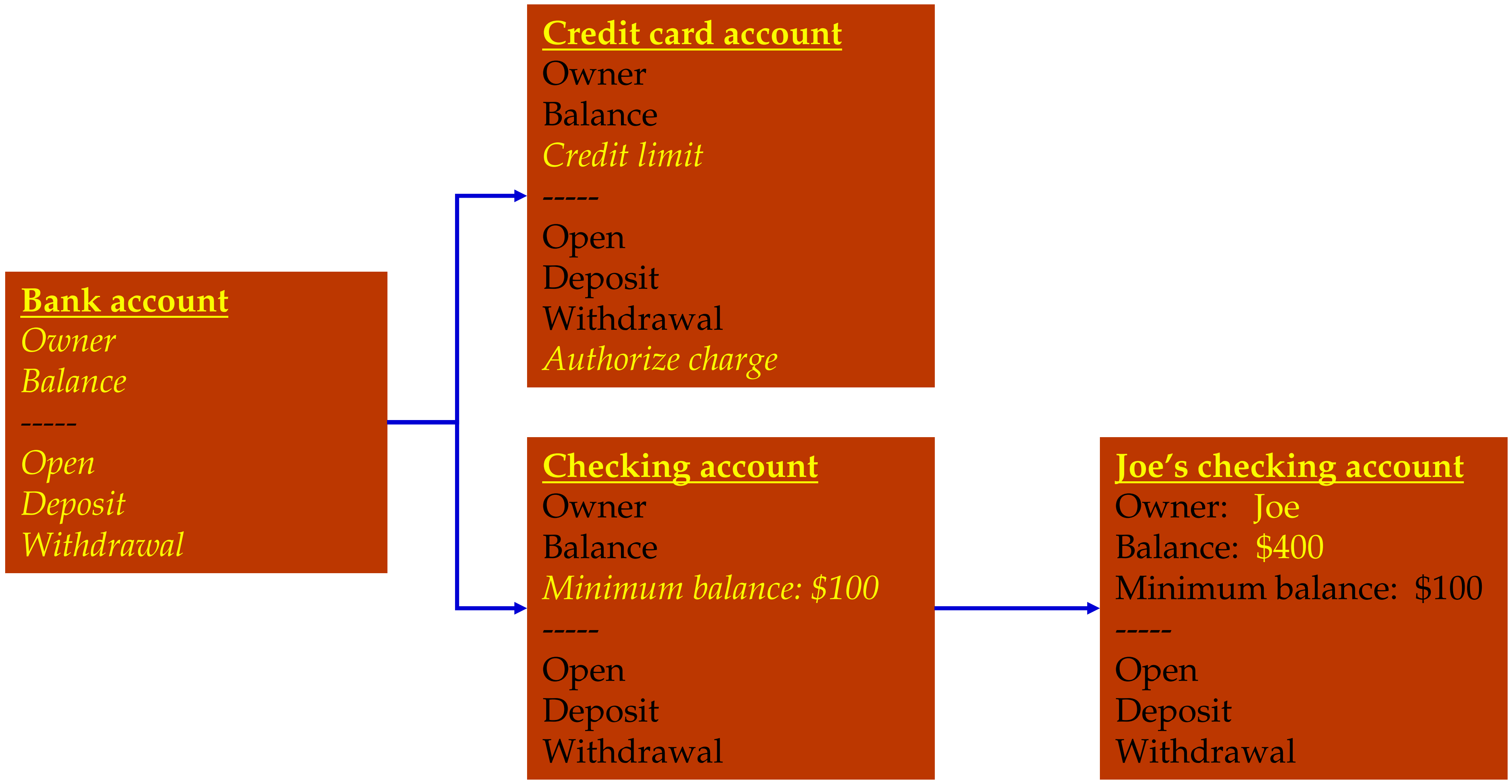
“Fourth-generation” languages

- Even closer to how people think about their problems
- Special purpose
- Examples:
  - Scripting languages
    - » FIND ALL RECORDS WHERE NAME IS “SMITH”
  - Spreadsheet formulas?

Object-oriented programming

- A special kind of high-level language
- Can increase programming efficiency and software re-use
- Combines procedures and data into “objects”
- Arranges objects in “class hierarchies”
- “Inherits” properties of objects in this hierarchy

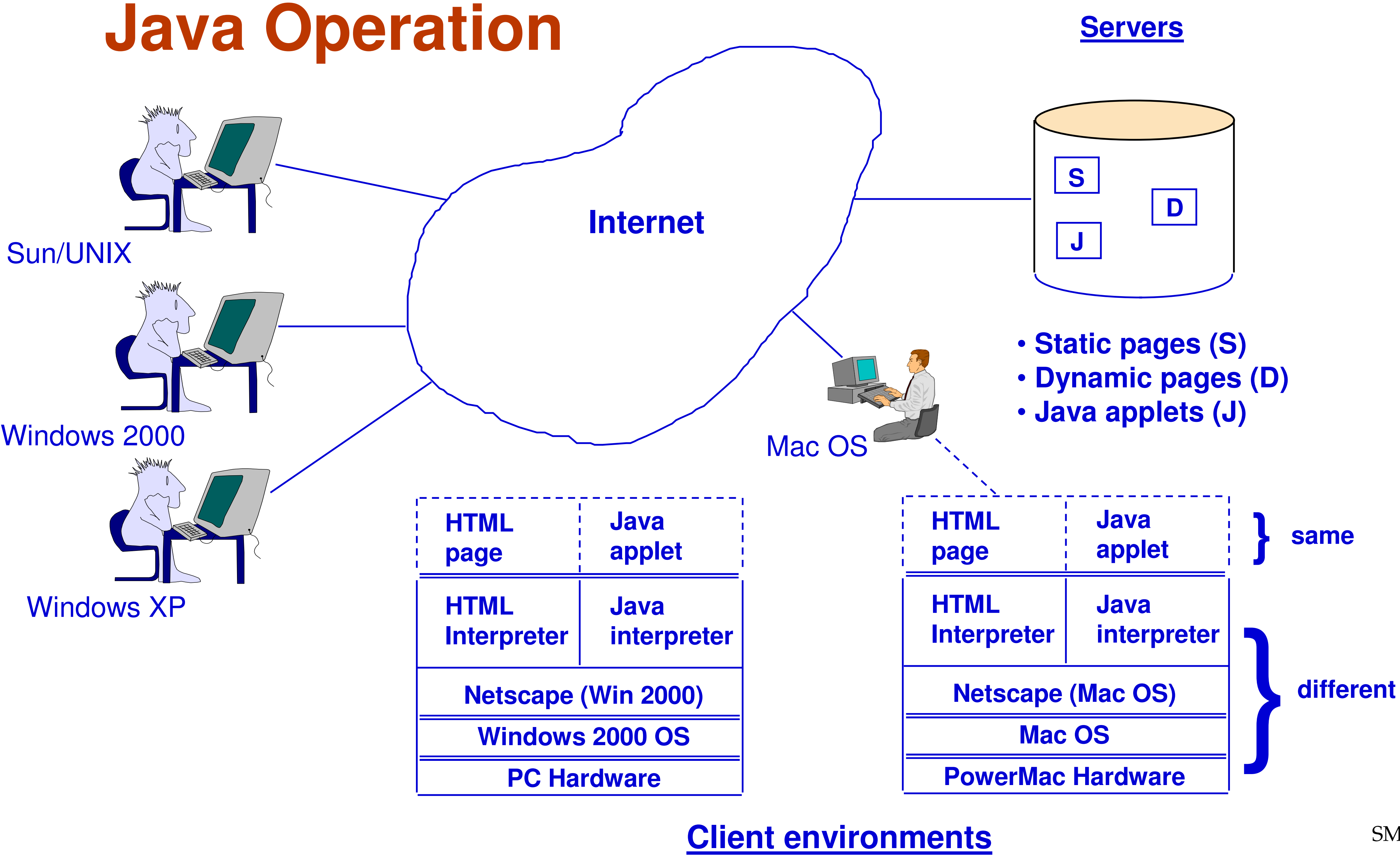
Class inheritance in object-oriented programming



What's good about Java?

- **Highly interactive**
  - Traditional Web - application software runs on server
  - Java applets <u>dynamically</u> downloaded and run on clien
    (e.g., input data validation)
- **"Nice" programming language**
  - Simpler than C/C++
  - Object-oriented
- **Secure programming environment**
  - "Sandbox" approach
- **Portable ("write once, run anywhere")**
  - Based on Java byte-code interpreter

Java Operation



- Static pages (S)
- Dynamic pages (D)
- Java applets (J)

Java reality check

- **Highly interactive** $\longrightarrow$ or too slow (interpretive)
- **“Nice” programming language** $\longrightarrow$ or too limited
- **Secure environment** $\longrightarrow$ or (1) not secure enough or (2) too secure (restrictive)
- **Portable** $\longrightarrow$ only if consistent Java interpreter availabl
  (Java “dialects”)

What will happen with Java?

Y2K problem

- Why was this a hard problem?
- Was money wasted?