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

<table>
<thead>
<tr>
<th>op-code</th>
<th>symbolic</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>stop</td>
</tr>
<tr>
<td>1xx</td>
<td>add</td>
</tr>
<tr>
<td>2xx</td>
<td>subtract</td>
</tr>
<tr>
<td>3xx</td>
<td>store</td>
</tr>
<tr>
<td>5xx</td>
<td>load</td>
</tr>
<tr>
<td>901</td>
<td>get</td>
</tr>
<tr>
<td>902</td>
<td>put</td>
</tr>
</tbody>
</table>

ASSEMBLY LANG. (Source Program)

<table>
<thead>
<tr>
<th>STEP</th>
<th>INSTRUCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>get</td>
</tr>
<tr>
<td>01</td>
<td>store w</td>
</tr>
<tr>
<td>02</td>
<td>get</td>
</tr>
<tr>
<td>03</td>
<td>store b</td>
</tr>
<tr>
<td>04</td>
<td>load w</td>
</tr>
<tr>
<td>05</td>
<td>add b</td>
</tr>
<tr>
<td>06</td>
<td>put</td>
</tr>
<tr>
<td>07</td>
<td>stop</td>
</tr>
</tbody>
</table>

MACHINE LANG. (Object Program)

<table>
<thead>
<tr>
<th>LOC</th>
<th>INSTRUCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>901</td>
</tr>
<tr>
<td>01</td>
<td>398</td>
</tr>
<tr>
<td>02</td>
<td>901</td>
</tr>
<tr>
<td>03</td>
<td>399</td>
</tr>
<tr>
<td>04</td>
<td>598</td>
</tr>
<tr>
<td>05</td>
<td>199</td>
</tr>
<tr>
<td>06</td>
<td>902</td>
</tr>
<tr>
<td>07</td>
<td>000</td>
</tr>
</tbody>
</table>

Source: S. Madnick, MIT
Machine language

- Binary
- Machine dependent
- Stored in the computer when the program is running
- Example:
  01110110001010010010 ....
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")

Get X
Add Y
Store Z
...

Assembler

Machine language program ("object code")

11001000100
01100100011
10001011001
...

...
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

`AVERRAGING 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; 
}
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")

Compiler

Machine language program ("object code")
“Interpreting” high level languages

High-level language program ("source code")

Interpreter

\[ Z = X + Y \]

...
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

Bank account
Owner
Balance
-----
Open
Deposit
Withdrawal

Credit card account
Owner
Balance
Credit limit
-----
Open
Deposit
Withdrawal
Authorize charge

Checking account
Owner
Balance
Minimum balance: $100
-----
Open
Deposit
Withdrawal

Joe’s checking account
Owner: Joe
Balance: $400
Minimum balance: $100
-----
Open
Deposit
Withdrawal

Joe’s checking account
Owner: Joe
Balance: $400
Minimum balance: $100
-----
Open
Deposit
Withdrawal

Joe’s checking account
Owner: Joe
Balance: $400
Minimum balance: $100
-----
Open
Deposit
Withdrawal

Joe’s checking account
Owner: Joe
Balance: $400
Minimum balance: $100
-----
Open
Deposit
Withdrawal

Joe’s checking account
Owner: Joe
Balance: $400
Minimum balance: $100
-----
Open
Deposit
Withdrawal

Joe’s checking account
Owner: Joe
Balance: $400
Minimum balance: $100
-----
Open
Deposit
Withdrawal

Joe’s checking account
Owner: Joe
Balance: $400
Minimum balance: $100
-----
Open
Deposit
Withdrawal

Joe’s checking account
Owner: Joe
Balance: $400
Minimum balance: $100
-----
Open
Deposit
Withdrawal
What’s good about Java?

• Highly interactive
  – Traditional Web - application software runs on server
  – Java applets **dynamically** downloaded and run on client
    (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

Servers

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

Internet

Client environments

HTML page
- HTML Interpreter
  - Netscape (Win 2000)
  - Windows 2000 OS
  - PC Hardware

Java applet
- Java interpreter

Sun/UNIX

Windows 2000

Windows XP

Mac OS

PC Hardware

Netscape (Mac OS)

PowerMac Hardware
Java reality check

• Highly interactive $\rightarrow$ or too slow (interpretive)

• “Nice” programming language $\rightarrow$ or too limited

• Secure environment $\rightarrow$ or (1) not secure enough or (2) too secure (restrictive)

• Portable $\rightarrow$ only if consistent Java interpreter available (Java “dialects”)
Y2K problem

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