

Lecture 1: Overview of Java

What is java?

- Developed by Sun Microsystems (James Gosling)
- A general-purpose object-oriented language
- Based on C/C++
- Designed for easy Web/Internet applications
- Widespread acceptance

Java Features (1)

- **Simple**

- fixes some clumsy features of C++
- no pointers
- automatic garbage collection
- rich pre-defined class library <http://java.sun.com/j2se/1.4.2/docs/api/>

- **Object oriented**

- focus on the data (objects) and methods manipulating the data
- all functions are associated with objects
- almost all datatypes are objects (files, strings, etc.)
- potentially better code organization and reuse

Java Features (2)

- **Interpreted**

- java compiler generate byte-codes, not native machine code
- the compiled byte-codes are platform-independent
- java bytecodes are translated on the fly to machine readable instructions in runtime (Java Virtual Machine)

- **Portable**

- same application runs on all platforms
- the sizes of the primitive data types are always the same
- the libraries define portable interfaces

Java Features (3)

- **Reliable**

- extensive compile-time and runtime error checking
- no pointers but real arrays. Memory corruptions or unauthorized memory accesses are impossible
- automatic garbage collection tracks objects usage over time

- **Secure**

- usage in networked environments requires more security
- memory allocation model is a major defense
- access restrictions are forced (private, public)

Java Features (4)

- **Multithreaded**

- multiple concurrent threads of executions can run simultaneously
- utilizes a sophisticated set of synchronization primitives (based on monitors and condition variables paradigm) to achieve this

- **Dynamic**

- java is designed to adapt to evolving environment
- libraries can freely add new methods and instance variables without any effect on their clients
- interfaces promote flexibility and reusability in code by specifying a set of methods an object can perform, but leaves open how these methods should be implemented
- can check the class type in runtime

Java Disadvantages

- **Slower than compiled language such as C**

- an experiment in 1999 showed that Java was 3 or 4 times slower than C or C++

title of the article: "Comparing Java vs. C/C++ Efficiency Issues to Interpersonal Issues" (Lutz Prechelt)

- adequate for all but the most time-intensive programs

Environment Setup

- Sun Solaris OS JDK 1.4 (latest: J2SE 5.0)

- You can use the lab at CL112. Please follow the steps:

- log into the Unix environment
- subscribe to JDK-CURRENT when you log in for the first time (% is a prompt sign)

```
%> subscribe JDK-CURRENT
```

and then log out by clicking on the EXIT button on the control panel

Install Java™ 2 Platform on your machine

- Can be installed on different platforms:

- Unix/Linux
- Windows
- Mac OS

- Follow the on-line instructions:

<http://java.sun.com/docs/books/tutorial/getStarted/cupojava/index.html>

Getting Started: (1)

- (1) Create the source file:

- open a text editor, type in the code which defines a class (*HelloWorldApp*) and then save it in a file (*HelloWorldApp.java*)
- file and class name are case sensitive and must be matched exactly (except the .java part)

Example Code: [HelloWorldApp.java](#)

```
/**
 * The HelloWorldApp class implements an application
 * that displays "Hello World!" to the standard output
 */
public class HelloWorldApp {
    public static void main(String[] args) {
        // Display "Hello World!"
        System.out.println("Hello World!");
    }
}
```

★ Java is CASE SENSITIVE!

Getting Started: (2)

(2) Compile the program:

- compile HelloWorldApp.java by using the following command:

```
javac HelloWorldApp.java
```

it generates a file named HelloWorldApp.class

☹ **'javac' is not recognized as an internal or external command, operable program or hatch file.**

javac: Command not found

if you see one of these errors, you have two choices:

- 1) specify the full path in which the javac program locates every time.

For example:

```
C:\j2sdk1.4.2_09\bin\javac HelloWorldApp.java
```

- 2) set the PATH environment variable

Getting Started: (3)

(3) Run the program:

- run the code through:

```
java HelloWorldApp
```

- Note that the command is `java`, not `javac`, and you refer to `HelloWorldApp`, not `HelloWorldApp.java` or `HelloWorldApp.class`

☹ **Exception in thread "main" java.lang.NoClassDefFoundError:
HelloWorldApp**

if you see this error, you may need to set the environment variable `CLASSPATH`.

Language basics (1)

- Data types

- 8 primitive types:

- boolean, byte, short, int, long, float, double, char

- Class types, either provided by Java, or made by programmers

- String, Integer, Array, Frame, Object, Person, Animal, ...

- Array types

- Variables

- *dataType identifier [= Expression]:*

- Example variable declarations and initializations:

<pre>int x; x=5; boolean b = true; Frame win = new Frame(); String x = "how are you?";</pre>	<pre>int[] intArray; intArray = new int[2]; intArray[0] = 12; intArray[1] = 6; Person pArray = new Person[10];</pre>
---	--

Language basics (2)

- Flow of control

- if, if-else, if-else if
 - switch
 - for, while, do-while
 - break
 - continue

Supplemental reading

- ***Getting Started***

<http://java.sun.com/docs/books/tutorial/getStarted/index.html>

- ***Nuts and bolts of the Java Language***

<http://java.sun.com/docs/books/tutorial/java/nutsandbolts/index.html>

- ***Compiling and Running a Simple Program***

<http://developer.java.sun.com/developer/onlineTraining/Programming/BasicJava1/compile.html>