

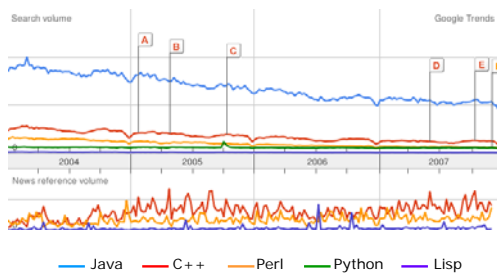
## Overview

### Lecture 1

## What is Java?

- Developed by Sun Microsystems
  - James Gosling
  - Birth: 1994 (progenesis from Oak)
- Based on C/C++
  - Similar syntax, control, data structures
  - Imperative, object-oriented
- Designed for building Web/Internet applications
- Widespread acceptance

## Google Searches

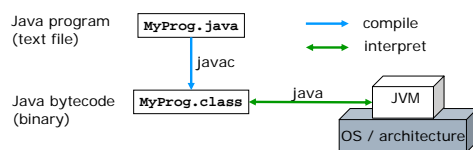


## Resources

- On line tutorials from Sun ("trails")
  - <http://java.sun.com/docs/books/tutorial>
- On line API documentation
  - <http://java.sun.com/j2se/1.5.0/docs/api>
- Class website
  - Handouts, lecture notes, lab assignments
  - Pointers to more resources

## The Java Virtual Machine (JVM)

- An abstract computer architecture
  - The software that executes Java programs
  - Part of Java Runtime Environment (JRE)
- Java program compiled into bytecode
- Java bytecode then interpreted by JVM



## Implications of JVM

- Portability
  - Sun slogan: "Write once, run anywhere"
  - JVM is ubiquitous
- Environment configuration
  - path variable
    - for shell to find java / javac executables
  - classpath variable
    - for JVM to find bytecode at execution time
- Dynamic extensibility
  - JVM can find bytecode on-the-fly
- Performance
  - Extra layer comes at (small) penalty in performance

## Environment Setup: JDK 1.5

- Version 1.5 == version 5
- Lab: CL 112 (& Baker 310 if available)  
<http://www.cse.ohio-state.edu/cs/labs.shtml>
- Follow these steps:
  - log into the unix or linux (ie stdsun or stdlogin) environment
  - subscribe to JDK-CURRENT  
% **subscribe JDK-CURRENT**
  - log out and log back in
- Confirm set-up  
% **java -version**  
**java version "1.5.0\_08"**  
. . .

## Install Java Platform at Home

- Can be installed on different platforms:
  - Solaris, Windows, Linux, ...
- Trail: Getting Started > "Hello World!"
  - Download OS-specific Java Development Kit (JDK)
    - Tools for program development (eg javac)
    - JRE
  - Create simple program (with a text editor)
  - Compile (with javac)
  - Run (with java)
- Make sure to download:
  - J2SE JDK (not J2EE, not JRE)
  - Version 5 (1.5.0\_15, ie JDK 5.0 Update 15. Do NOT download Version 6)

## Getting Started: 1. Creating Source File

- Using any text editor:
  - Create a file `HelloWorldApp.java`
  - Copy the following code into this file:

```
public class HelloWorldApp {
    public static void main(String[] args) {
        // Display "Hello World!"
        System.out.println("Hello World!");
    }
}
```
- Note:
  - Class name must match file name
  - Java is CASE SENSITIVE!

## Getting Started: 2. Compiling the Program

- Compile using javac

```
% javac HelloWorldApp.java
```
- Generates a file named `HelloWorldApp.class`

```
% ls
HelloWorldApp.class HelloWorldApp.java
```
- Problem  
`javac: command not found`
- Cause
  - Shell can not find javac executable
- Solutions
  - Use full path on command line

```
% /usr/local/jdk1.5.0_08/bin/javac HelloWorldApp.java
```
  - Set path environment variable

```
% set path=(%path %usr/local/jdk1.5.0_08/bin/javac)
```

## Getting Started: 3. Running the Program

- From same directory, run using java

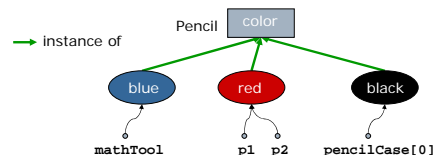
```
% java HelloWorldApp
Hello World!
```
- Note:
  - argument is `HelloWorldApp`, *not* a file (`.java` or `.class`)
- Problem  
Exception in thread "main" `java.lang.NoClassDefFoundError: HelloWorldApp`
- Cause
  - JVM can not find `HelloWorldApp` bytecode (ie `.class` file)
- Solutions
  - Explicitly set classpath on command line

```
% java -classpath ~/code/example HelloWorldApp
```
  - Set classpath using environment variable

```
% setenv CLASSPATH ~/code/example
```

## Object-Oriented Programming

- Fundamental component is an *object*
  - Encapsulates state and behavior
  - Contains *fields* and *methods*
- Each object is an instance of a *class*
  - Class declaration is a blueprint for objects



## Object Creation and Deletion

- Class constructor initializes fields

```
class Pencil {
    String color;
    Pencil (String c) {
        color = c;
    }
}
```
- Explicit object creation with new();

```
java.util.Date d = new java.util.Date();
Integer count = new Integer(34);
Pencil p1 = new Pencil("red");
```
- Unlike C/C++, references not explicitly freed
  - References go out of scope (become null)
  - *Automatic* garbage collection (eventually) deletes unreachable objects

## Sample Class Declaration

```
class Pencil {
    String color;
    int length = 14;

    Pencil (String c) {
        color = c;
    }

    void setColor (String newColor) {
        color = newColor;
    }

    String getDescription () {
        if (length < 15)
            { return "small: " + color; }
        else
            { return "large: " + color; }
    }
}
```

## Language Basics: Statements

- Similar to C/C++
- Control flow:
  - if, if-else, if-else if
  - switch
  - for, while, do-while
  - break
  - continue
- Statements
  - Separation with ;
  - Blocks with { ... }
- Comments with // or /\* ... \*/
- Operators
  - arithmetic: + - \* / % ++ -- ...
  - logical (for booleans): & | ^ ! && ||
  - bit (for integer types): & | ^ ~ << >> >>>
  - relational: == != < > <= >=

## Language Basics: Types

- Primitive types
  - boolean, byte, short, int, long, float, double, char
  - Size of each defined by language (eg int: 4 bytes)
- Reference types, provided by:
  - Java standard libraries
    - String, Integer, Date, System, ...
  - Programmer
    - Person, Animal, HelloWorldApp, ...
- Arrays
  - Can contain primitive or class types
    - int[], float[], String[], ...
  - Indexed starting from 0

## Language Basics: Variables

- Declaration

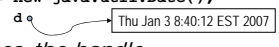
```
int counter; boolean isDone;
String firstName; Date d;
float[] measurements; Animal[] zoo;
```
- Explicit initialization with declaration

```
int counter = 3;
boolean isDone = false;
String firstName = "Nima";
java.util.Date d = new java.util.Date();
Integer c = new Integer(34);
float[] measurements = {3, 5, 6.5};
Animal[] zoo = new Animal[50];
```
- Implicit initialization (for *data members* only, lecture 2)
  - Language-defined values for all primitive types
    - false for boolean, 0 for int, ...
  - Special value (null) for class types

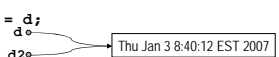
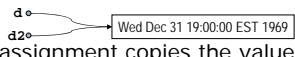
## References: Aliasing

- A reference is a handle to some data

```
java.util.Date d = new java.util.Date();
```


- Assignment copies *the handle*
  - Creates an alias

```
java.util.Date d2 = d;
```


  - `d.setTime(0);`
- cf. Basic types: assignment copies the value

```
int i = 3;
int j = i;
i++; //now i==4 and j==3
```

## Best Practices: Files and Classes

- Declare one class per file
- Give file the same name as the class declaration it contains
  - class HelloWorldApp declaration appears in HelloWorldApp.java
  - class Pencil is defined in Pencil.java

## Good Practice: Single-Statement Conditionals

- Always include body of if-else in braces, even if it is a single statement
- The following is correct, but discouraged:

```
if (!isDone)
    retry = true;
```

- Instead, write:

```
if (!isDone) {
    retry = true;
}
```

## Supplemental Reading

- Sun trails
  - Getting Started
  - Learning the Java Language > Language Basics
- Java overview white paper
  - <http://java.sun.com/docs/white/langenv/>
- Another walk-through of simple application
  - "Essentials of the Java Programming Language, Part 1"
  - <http://developer.java.sun.com/developer/onlineTraining/Programming/BasicJava1/compile.html>
  - Lessons 1 and 2

## Summary

- JVM
  - .java (source) vs .class (bytecode)
  - javac (compiler) vs java (interpreter)
- Environment configuration
  - Setting class and classpath
- Sample program: Hello World
- Language basics
  - Statements
  - Types
  - Variables