Advanced Java Programming
Introduction

Ohad Barzilay,
Tel-Aviv University
Spring ‘06

Java – The History

- 1991 – Sun Microsystems initiates project “Green” with the intent to develop a programming language for digitally controlled consumer devices and computers.

- The language OAK was developed by James Gosling with the goal of using C++’s popularity but without many of the annoying features.

The Java Platform

- OAK was later renamed to JAVA and released in 1995.
- Design goals:
  - Object oriented language
  - Allow an automatic garbage collection mechanism
  - Security
  - Distributed programming
  - Threading support
  - Easily portable on any environment and devices.

Java Meets the Internet

- In 1994 java engineers retarget their effort for internet technology that was growing rapidly.
- The new design goals were to create a platform independent language, that can be used to execute remote code and generate dynamic web contents easily.

Java Meets the Internet

- 1995 Netscape announces that it's browser will support java technology.
  - Java was to contain language facilities and libraries for networking.
  - Java was designed to execute code from remote sources securely.

Simplicity

- Java would be a simple language to use:
  - No Pointers arithmetic's
  - No operator overloading
  - No Multiple Inheritance
  - No automatic conversion of types
  - No need for manual memory allocation (uses a garbage collection mechanism)

Object Oriented

- Java is an almost pure object oriented programming language.
- A programming task is about defining new types as classes.
- An execution of a program is about creating instances (Objects) from the classes, and letting them interact.

Object Oriented Concepts

- Size does matter!
- “Making it work” is not enough
- Maintenance of software is too expensive
- Some principals need to be adopted to be able to write and maintain large software systems

Object Oriented Concepts

Most of the course will deal with these features of java:

- Abstraction
- Objects
- Classes and Interfaces
- Encapsulation / Information hiding
- Messages
- Generalization and Inheritance
- Aggregation
- Polymorphism

Easy Application Development

- Java contains a rich set of built in classes that can be used in programs. The classes are divided into packages according to topics such as:

  `java.lang, java.awt,`

  `javax.swing, java.net,`

  `java.io, java.util, java.applet`

Platform Independence

- Programs in Java must run similarly on diverse hardware. *"Write Once – Run Anywhere"*.
- Java is compiled "halfway" into byte code, which is then run on a virtual machine.
- The virtual machine is written in native code on the target machine and translates the byte code into usable code on the hardware.

Platform Independence

- At first the virtual machine interpreted the byte code. Interpreted code runs slower than compiled programs, so java suffered a reputation for producing slow programs.
- Today the Java VM produces programs that run much faster, using multiple techniques as the *just-in-time* compiler or "JIT".
- JIT compiles the Java bytecodes into native code at the time the program is run.

Platform Independence

- Platform independent Java is very successful with server side applications, such as web services, servlets, and Enterprise Java Beans.

Network Savvy

- Java has an extensive library for coping easily with TCP/IP protocols like HTTP and FTP.
- Easy to creating network connections. Java applications can open and access objects across the net via URLs with the same ease that programmers are used to when accessing a local file system.

Network Savvy

Other networking features include:

- RMI – remote method invocation, use of remote objects and methods.
- Applets – small java applications that run within the browser, and are loaded via the web. Allow dynamic and interactive content to be generated.
- Servlets – Enhance the capabilities of web servers and allow the creating of dynamic web content.

Secure execution of remote code

- Java provides support for the execution of code from remote sources.
- An applet runs within a browser, executing code downloaded from a remote HTTP server.
- The remote code runs in a highly restricted "sandbox", which protects the user from misbehaving or malicious code
- Applets can be digitally signed as "safe", giving them permission to break out of the sandbox and access local filesystem and network.

Robust

- Static typing.
- Mechanism for preventing overwriting memory and data corruption.
- Programs cannot gain unauthorized access to memory.
- No pointer arithmetic.
- True arrays

Multi Threading

- Built in language support for concurrent programming allowing to perform more than a single task at once.
- Useful for animation, GUI design, playing sounds, interactive systems and distributed algorithms.

Comparing Java with C++

C++	Java
Arrays and Strings can be manipulated by the user	All non primitive types are objects
Signed and unsigned numeric types	All numeric types are signed
Primitive type size varies by platform	Fixed length primitive types for all platforms
Integer results may be interpreted as Boolean conditions	Conditions must be Boolean expressions.

Comparing Java with C++

C++	Java
Can have stand-alone functions.	All methods are part of a class
Can have Global Variables	No global variables
Multiple inheritance	No multiple inheritance
Operator Overloading	No Operator Overloading (besides string + and array [])
Static function binding by default	All methods (except final and static method) are dynamically bound.
No automatic garbage collection	Automatic garbage collection
Heavy Reliance on preprocessor	No Preprocessor

Writing a java program

- We begin by writing a java program and saving it in a file using the `.java` extension.
- The `.java` file is a text file containing text which is of correct java syntax.
- This text file can be easily edited by us or other programmers, but it makes no sense for the computer.

Compiling the java program

- Once a program is saved into a `.java` file, the compiler is used to translate the Java source code into Java bytecode.

```
> javac fileName.java
```

Compiling the java program

- Once the program is compiled a new file is created with the same name and the `.class` extension
- This file contains the translated byte-code from the program we compiled.
- Java byte code is platform independent. It must be translated to the local machine language in order to be executed.

Executing the program

- The java byte code will be executed by the interpreter (the java virtual machine).

- The java interpreter name is java and it will be called

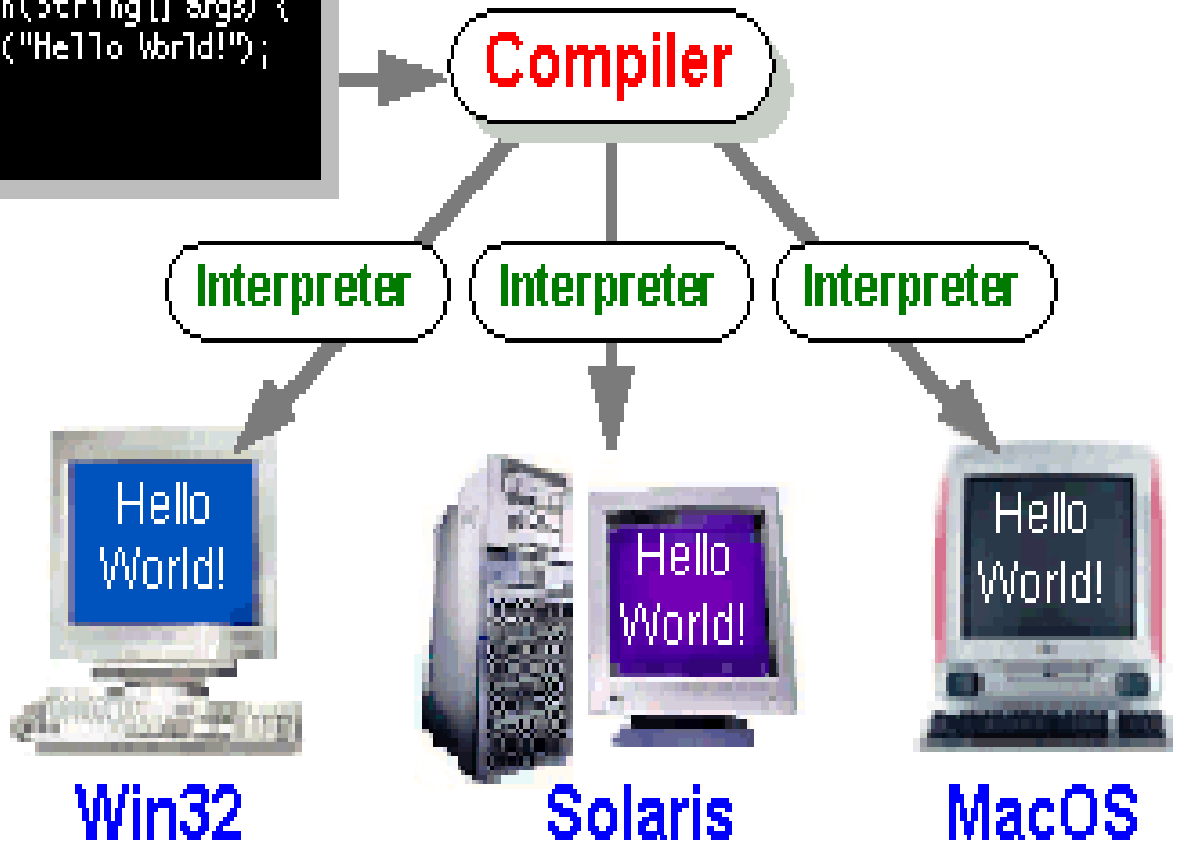
```
> java <filename_of_bytecode>
```

Java Code

Java Program

```
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

HelloWorldApp.java



An Example java Program

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

Java ByteCode

```
class Hello extends java.lang.Object {
    Hello();
    public static void main(java.lang.String[]);
}
Method Hello()
   0 aload_0
   1 invokespecial #1 <Method java.lang.Object()>
   4 return
Method void main(java.lang.String[])
   0 getstatic #2 <Field java.io.PrintStream out>
   3 ldc #3 <String "Hello World">
   5 invokevirtual #4 <Method void println(java.lang.String)>
   8 return
```

Java Development Kit

- The JDK provides all the necessary components for developing a java program.

- The JVM (`java`), java compiler (`javac`), `javadoc`, `jar` utility and many more are part of the JDK.

Setting the Environmental variables

- The `CLASSPATH` variable indicates the JVM and the java compiler where classes your program uses can be found.

JAR Utility

- The Jar utility (Java Archive) enables us to wrap and compress folders containing code and accessories needed to execute our java programs.
- A jar file is basically a zip file.

IDE

- An **I**ntegrated **D**evelopment **E**nvironment contains many tools needed for the development process:
  - Editor, Compiler, JRE (VM), Debugger, Browser, and more…
- Our recommended development environment is called Eclipse:
  - It is free. It is open source. It is supported by IBM
  - Download it today! www.eclipse.org
  - See the course web site for more useful links and tutorials