



Java Programming:

Object-Oriented Programming in Java

Syllabus

Introduction to Java





Administrivia

- Resources
 - Web pages + Syllabus
 - Mailing list
 - Contact info
- Schedule
 - Lecture schedule
 - Office hours
 - I always respond to e-mail
 - IM

Expectations

- My responsibility
 - Organize
 - Motivate
 - Connect
- Your Responsibility
 - Think
 - Interact
 - Get involved, participate
 - Get excited

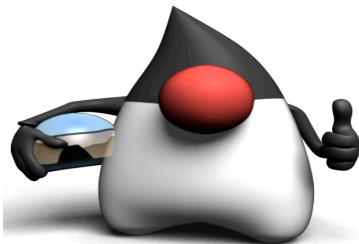
Assignment Review

- NO picture taking in class during assignment Review
- The intent of sharing the solution is to help you in understanding it not copy it.
- Assignments now due at 6pm on day due.

Topics Covered

- OO Concepts
- Classes and Objects
- Object Collections
- Libraries and Documentation
- Class Design
- Inheritance
- Abstraction
- Error handling

History of Java



Tumbling Dice

In the Beginning

- Oak (1990), originally designed and named by James Gosling
- Developed at Sun for small programmable devices and appliances
 - Language
 - Runtime Environment
- Released in 1995
- Demonstrated how to deliver executable content on the Web

History of JDK Release

- JDK 1.0
 - 8 packages, 212 classes, 2125 members
- JDK 1.1.x
 - 23 packages, 504 classes, 5478 members
- SDK 1.2
 - 62 packages, 1592 classes, 18837 members
- SDK 1.3
- SDK 1.4
- J2SE V5 - now calling JDK 5.0 but bits are still 1.5.0
- Java SE V6 - called JDK 6.0 but bits are still 1.6.0
 - 272 packages, 4597 classes

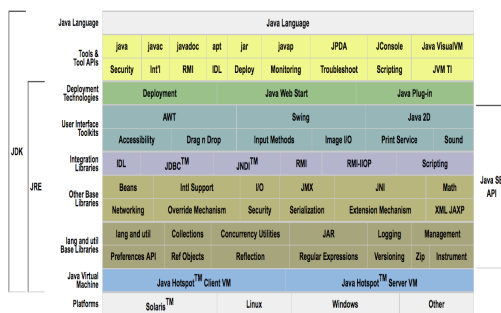
Tools in the Java Developers Kit (JDK)

- | | |
|---|---|
| • javac - Java compiler | • javap - Java disassembler |
| • java - Java interpreter | • javadoc - Java documentation generator |
| • appletviewer - viewer for Java applets | • Documentation for the JDK can be explored with your Web browser |
| • jdb - Java debugger | |

Java Platform Today

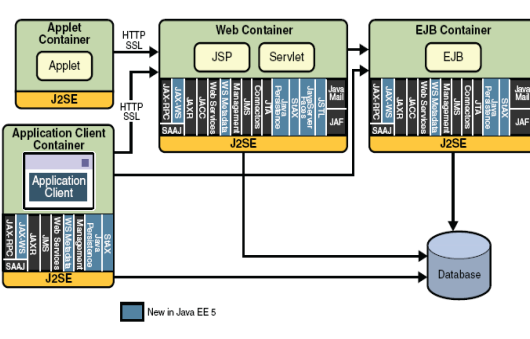
- Sun has developed subsets of the Java platform
 - Java Enterprise Edition – Java EE
 - Java Standard Edition – Java SE
 - Java Micro Edition – Java ME

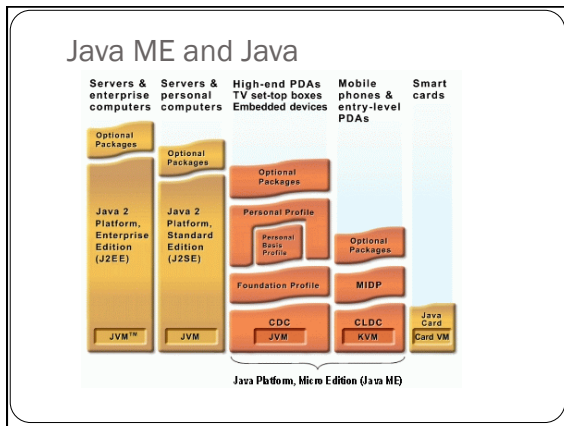
Java SE

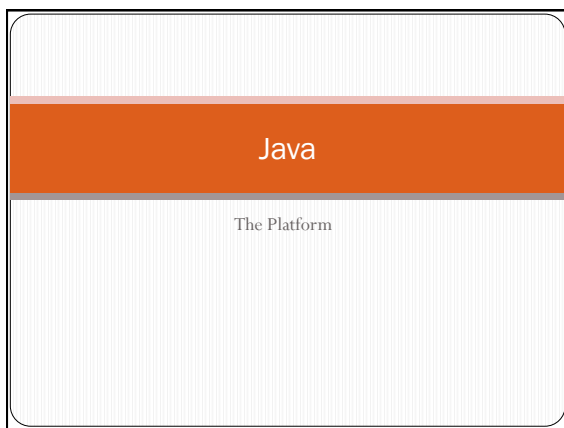


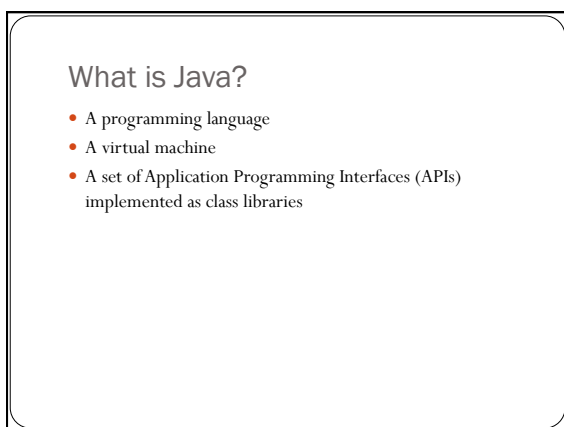
Other languages on JVM - <http://www.is-research.de/info/vmlanguages/>

Java EE









Java, The Language

- Based on many different languages
- OO Based
- Exceptions/Errors

Java, The Virtual Machine

- Hardware abstraction
- Many features of computer hardware: opcodes that represent fundamental computing tasks, assembly tools
- The Java VM executes opcodes stored in class files
- Note that class files could be (and are) generated by source in other languages

Java vs. Other Languages

- Java syntax is very much like C syntax
- Java does not support pointers or any other direct access to memory
- Java is automatically garbage-collected, so deallocating memory is not necessary
- Java is interpreted + HotSpot
- Java is dynamically linked, with run-time polymorphism.
- Java is more “purely” object-oriented than C, C++, Pascal, Visual Basic, COBOL, PL/I, etc.

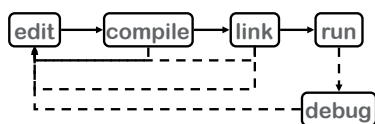
Java, The Libraries/APIs

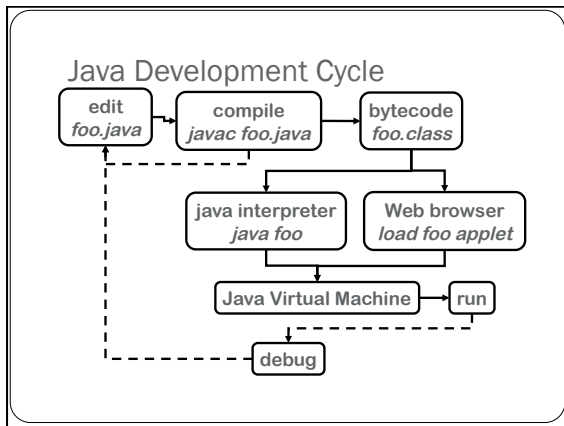
- `java.lang`
 - class, numbers, strings, System, Exceptions, Threads and more
- `java.io`
 - streams, readers, writer, files
- `java.net`
 - Sockets, URLs, datagrams, InetAddresses, connections
- Just the beginning – Look at the Javadoc!!!

Development Environments

- Simplest
 - Command line compiler from Sun
 - Notepad or your favorite text editor
- Programmer Editors
- IDEs

Typical Language Development Cycle





Demo

- IDE Tool

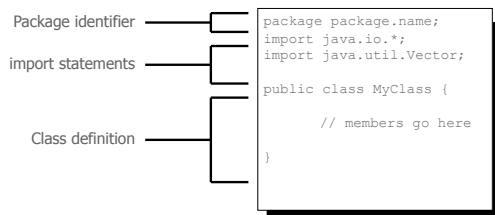
Getting Started

Source File Structure

- Simple structure in order
 - package definition (Optional)
 - package and/or class import statements (Optional)
 - Class definition (multiple are allowed but messy)

Source File Structure (Continued)

Three components to a Java source
file, in order



Source File Structure (Continued)

- Constraints
 - if more than 1 class, only one declared public.
 - source file must have same name as name of public class

Source File Structure (Continued)

Packages

- A way to group related classes
- More importantly, a key part of Java's encapsulation mechanism (we'll see this later in the course)
- Class is permanently associated with its package
- Period (.) separated name mirrors directory structure where classes are stored
- "Default" package is the current directory
- Classes without a package identifier are in the default package

Source File Structure (Continued)

importing classes

A class' full name includes its package. Example:
java.util.Vector or java.lang.String.

- Sometimes it is more convenient simply to use the class name without the package, e.g., Vector, String.
- The `import` statement allows this shortcutting
- Classes can be imported individually, or all classes in a package can be imported
- `java.lang.*` is imported automatically by the compiler
- is not like `#include` in C/C++

Java Keywords

abstract	boolean	break	byte	case
catch	char	class	continue	default
do	double	else	extends	false
final	finally	float	for	if
implements	import	instanceof	int	interface
long	native	new	null	package
private	protected	public	return	short
static	super	switch	synchronized	this
throw	throws	transient	true	try
void	volatile	while		

Keywords that are reserved but not used in Java
const goto

Java Language

- Language Basics
 - Identifiers
 - Classes
 - Primitives Types
 - String Class
 - Expressions
 - Operators
 - Basic Statements

Identifiers

- Variable, method, class, or label
- Keywords and reserved words not allowed
- Must begin with a letter, dollar(\$), or underscore(_)
- Subsequent letters, \$, _, or digits
- foobar // legal
- 3_node // illegal

Comments

- Java supports three comment formats:
 - Comments starting with “//” go to the end of the line.
 - Comments starting with “/*” continue to the next “*/”.
 - Comments starting with “/**” continue to the next “*/”. These comments are processed by javadoc, a tool in the JDK.
 - Javadoc is an automatic documentation generator that parses the Java source code and these special comments to generate HTML describing the API.

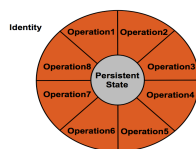
Reference and Primitive Types

- References
 - Constructed by operator 'new'
 - Based on Class definition
- Primitives – performance (1995)
 - Boolean
 - Integer
 - Floating-point
 - Character

Language Support for OO

- Objects
- Classes
- Run-time binding
- Automatic memory management

Objects



- Identity – All objects have an identity, the identity provides a means to reference, denote, name or label an object. The identity is unique within the run-time environment.
- Operations – An object provides one or more capabilities to the world outside itself. Each of these operations has a signature.
- Persistent State – Objects maintain state information.

Objects

Interact via:

- Message – Objects interact by sending requests to perform an operation to one another using messages. Method invocations are synonymous with messaging.
- Reply – The requested object may reply with a result.

Objects

Are Implemented with:

- Attributes / Instance variables – Individual piece of the persistent state of the object.
- Methods – The implementation of one or more operations. Similar to subroutines or functions within an object. May make requests of other objects.

Class

- An abstraction representing a set of objects that exhibit the same characteristics and that are subject to the same constraints. The class definition is the template or stamp from which objects are formed.

- Advantages & Disadvantages

- + Supports further abstraction by deferring the issue of how many objects will exist
- Adds complexity to the language, and especially the compiler/interpreter.

Wikipedia: http://en.wikipedia.org/wiki/Class_%28computer_science%29

Run-time Binding

- The act of associating program units to a name at run-time. The name may refer to objects, attributes, methods or other program units.
- Binding Stages in Run-time
 - Early (compile-time) – Program elements are statically associated to the name. The compiler/linker (or interpreter) does this to ensure that subsequent uses of the name are consistent.
 - Late (run-time) – Program elements are dynamically associated, and may be changed while the program is executing

Automatic Memory Management

- Management of the memory associated with creation and deletion of objects can be significant. Some languages perform the deletion task automatically, this is commonly referred to as “garbage collection”.

Garbage Collection

Advantages & Disadvantages

- + Frees the programmer from worrying about yet another thing that can go wrong.
- + Reduces the probability of unknowingly referencing garbage or non-existent memory.
- + Reduces the probability of memory leaks.
- Garbage collection takes significant time and occurs at unpredictable intervals.
 - A potential killer for real-time and interactive systems.
- Several memory management models to choose from, the one chosen by the language designer may not be the best suited for a particular application.

Assignment 1
