JAVA

Features

simple
object-oriented (relatively pure oo, not procedural + oo extensions)
distributed
both interpreted and compiled instruction sets
robust
secure
architecture neutral
portable
high-performance
multi-threaded
dynamic

Object Orientation

class = abstraction
    class variables
    class functions
instance
    fields are instance variables
    methods are functions
hierarchy
subclasses = design by difference
inheritance
overloading
constructors
accessors
encapsulation (public, package, protected, private)

Implementation Features

virtual machine
    byte-code = machine instructions for a virtual machine (VM)
    VM maps closely to most native hardware machine
call-by-value parameter passing (compare to call-by-name, call-by-need)
    the value of an object is its reference
    copies binding into parameter field of method
automatic garbage collection
streams
type-safe references (strong typing)
exception handling
multiple threads (multitasking, lightweight)
simultaneous processes and shared objects
    locks; user provided deadlock avoidance
    automatic switching, scheduling, synchronization

Language Features

base data-types are not objects
first-class strings, read-only
international Unicode character set
first-class exceptions, checked by compiler
HTML inline interface
first-class network interface (URL, TCP, sockets)
protection and security model
class Object is root
interface concept for limited multiple inheritance

no pointers (use references instead)
no global variables (use root classes)
no goto (use catch/throw and labels)
no operator overloading (static basic operators)
no delete

Language Keyword Features

final:	constants, unforgable classes, non-overridden methods
this:	reference to self object
new:	constructs a new object or class
. :	accessor function
[ ] :	arrays
{ } :	sequential block
super:	references things from the superclass(es)
try-catch-finally:	exception handling
labeled break:	for skipping sequences and exiting loops

Packages

class libraries
functionality groups
user interface code provided
user provide application specific abstract data types

Provided Java API Packages

java.lang	the language
java.net	networking
java.io	streams and files
java.util	utilities, higher-order data-structures (enumeration, vector, stack, dictionary, hashtable)
java.awt	Abstract Window Toolkit
java.awt.image	image processing
java.awt.peer	interface with native interfaces
java.applet	basic applets

Interfaces

unique in Java
separate design inheritance from implementation inheritance
can inherit a contract without inheriting an implementation
tie together dissimilar classes for object reference
subclasses provide code for all interface methods
multiple inheritance (classes can implement multiple interfaces)
no root, does not default to Object root-class
constrained to:
- abstract class (no instances, only subclasses)
- no code, only abstract method declarations
- static and final variables
- public methods

Exceptions

catch and throw handlers
programmer declared compile-time errors
cleanly checks for errors without cluttering code
try/catch/throw environment
finally clean-up

Protection

runtime system does not permit memory access

public	full access by all classes
package	access by classes in common library
protected	access by subclasses only
private	no access by other classes

Streams

usually paired as InputStream, OutputStream
Piped, Filter, Buffered
StreamTokenizer

System Programming Classes

Runtime	(state of Java at runtime)
Process	(running java process)
System	(state of environment)
Math	(standard computations)
Native	(foreign function interface)

Abstract Window Toolkit (AWT)

- embedding within the local browser
- standard component set
  - button, checkbox, choice, label, list
  - scrollbar, textarea, textfield,
  - windows, menus, dialog boxes
- containers
  - graphical collections of components
- layout management
- event handling
  - mouse clicks and movements
  - keyboard
- graphics
  - drawing, color, fonts, clipping, image handling

Sample HTML Applet Call

```
<HTML>
<HEAD>
<TITLE>Applet Page</TITLE>
</HEAD>
<BODY>
<H4>This is an example of a Java applet:</H4>
<HR> <APPLET CODE="MyApplet.class" WIDTH=100 HEIGHT=50> </APPLET> <HR>
</BODY>
</HTML>
```

Sample Applet

```
import java.applet.Applet;
import java.awt.Graphics;
public class MyApplet extends Applet
      {public void paint(Graphics g)
            { g.drawString("Hello world.", 5, 10); } }
```

Web Resources

http://java.sun.com/	...from the Source
http://www.rpi.edu/~decemj/works/java.html/	a Java book author
http://www.gamelan.com/	registry of programs
http://sunsite.unc.edu/javafaq/javafaq.html	FAQs
http://www.well.com/user/yimmit/	links to resources
http://www.natural.com/	major developer
http://www.io.org/~mentor/J__Notes.html	more resources
http://www.acm.org/~ops/java.html	ACM resources
http://www.yahoo.com/Computers/Languages/Java/	search engine resources
http://rendezvous.com/Java/hierarchy	class diagrams

DEFINITIONS

Abstract (oo)
a class which is intended to have no instances

Accessor (prog)
special function to retrieve hierarchical data

Applet (java)
a dynamic, interactive program that runs inside a Web page

Attributes (prog)
the instance variables of an object

Bytecode (java)
machine instructions for a virtual machine

Casting (java)
changing the type of data. Coercing.

Class (oo)
template which abstracts objects with similar features

Clipping (graphics)
redrawing within a container

Constructor (prog)
special method for creating and initializing new instances

Contract (java)
semantics of the set of methods, no class implementation

Encapsulation (oo)
limited access to class methods and fields (public, package, protected, private)

Errors (java)
Runtime violation of system constraints; usually not recoverable.

Exceptions (java)
Compiler checked violations of typing, ranges, assignments; usually catchable.

Finalizer (java)
special method for closing and reclaiming old instances. Inverse of constructor.

Garbage Collection (java)
automated management of memory

Inheritance (oo)
hierarchy of included functionality; design and implementation by difference.
Single inheritance: inherit from only one superclass (tree)
Multiple inheritance: inherit from several superclasses (DAG)

Instance (oo)
concrete digital objects with bound properties. Same as Object.

Interface (java)
limited type of class which provides multiple inheritance
abstract class, no method implementation, static and final variables

Method (oo)
the functions within an object or a class

Overriding (oo)
subclass methods which redefine superclass methods

Package (oo, java)
set of classes, usually with functional similarities. Same as Class Library

Polymorphism (oo)
objects belong to all classes in their class hierarchy

Signature (prog)
the abstract form of a method (name, type of object returned, parameter list)

Statement (prog)
A program component. Expressions return a value; declarations define a scope.

Streams (prog, java)
A communication path between data source and destination

Subclass (oo)
the class(es) below a class in the class hierarchy

Superclass (oo)
the class(es) above a class in the class hierarchy

Threads (prog)
basic unit for multitasking, used for long processes

Variable (prog)
the data within an object or a class

Virtual Machine (java)
software which emulates a physical machine