

Java Overview

Mark Volkmann
Object Computing, Inc.



What Is Java?

- A portable, object-oriented programming language
- Compiled to portable bytecode
- Executed by machine specific interpreters
 - Java Virtual Machine (JVM)
- Can develop on one platform and deploy on others
- Used for
 - applications
 - applets
 - servlets
 - Enterprise Java Beans (EJB)

more on these later



Java Development Kit (JDK)

- Free
 - from Sun, IBM, Microsoft and others
- Includes
 - compiler (javac)
 - application interpreter (java)
 - applet tester (appletviewer)
 - documentation generator (javadoc)
 - debugger (jdb)
 - large class library
 - and more (jar, javah, javap, rmic, ...)



Java Compared To C++

- C++
 - has
 - multiple inheritance
 - operator overloading
 - templates
 - somewhat more difficult syntax to learn (ex. pointers)
 - lacks
 - automated garbage collection
 - portability
 - must compile and link for a specific platform
 - some features not implemented in all compilers
 - large, standard, portable class library
 - GUIs, threads, sockets, reflection, object serialization, database access, security, and more



Object-Oriented (OO) Benefits

- Encourages better software design
 - can model real world by creating classes that represent real world concepts (people, places, things, operations, transactions, ...)
- Easier to maintain
 - problems related to a real world concept are often isolated to a single class
 - don't have to examine as much code to find the source of a problem
- Easier to extend
 - can add new functionality by reusing existing classes, either through inheritance or composition
- Can have less to write due to reuse
 - reuse custom classes created by your company
 - reuse classes from commercial or free class libraries such as the JDK



Object-Oriented Terminology

- Encapsulation
 - can define classes that encapsulate data and methods that operate on it
 - example Car class
 - attributes
 - make, model, year, mileage, color, license
 - methods
 - rent, checkIn, getMaintenanceHistory, updateMaintenanceHistory, sell
- Inheritance
 - can create new classes that take the data and methods of another class and add more or redefine
- Polymorphism
 - can operate on objects of any class from a group of related classes where each can have its own implementation of the operation
 - an advanced topic that can simplify code and make it more extensible



Object-Oriented Example

- Car class
 - attributes (fields)
 - make, model, color, mileage, purchase price, rental rate, sale price, maintenance history
 - methods
 - purchase, rent, repair, sell
 - superclass
 - could inherit from a Vehicle class
 - subclasses
 - could derive subclasses like CompactCar, LuxuryCar and SUV



Types of Java Programs

- Applications
 - bytecode is resident on client machine and is executed there
 - can invoke remote code using facilities such as RMI, EJB and CORBA
 - can be run from command line
- Applets
 - typically downloaded from web server (can be installed locally)
 - run in browser
 - can use JVM supplied with browser or Java Plug-in
- Servlets and Java Server Pages (JSP)
 - typically invoked from a browser (servlets can invoke other servlets)
 - run on a web server
 - typically generate HTML to be displayed in a browser



Java Native Interface (JNI)

- Can call functions written in C, C++ and assembly language
 - called native methods
- Can start a Java Virtual Machine (JVM) from those languages
 - using the Invocation API
- Allows Java to be used where it is most appropriate
 - other languages may be more efficient for specific tasks
- Allows legacy code to be used from Java



Collections API

- Classes for operating on collections of objects
- Includes
 - List ordered collection that may contain duplicates
 - Set unordered collection that may not contain duplicates
 - Map unordered collection of key/value pairs, both of which can be any object type
 - SortedSet ordered set
 - SortedMap ordered map
 - Iterator for iterating through the contents of a collection without knowledge of the collection type
 - Comparator for creating custom-ordered collections



Threads

- Multiple “threads” of execution
 - run simultaneously with multiple processors
 - take turns with a single processor
- Used to
 - achieve asynchronous operation (models real world activities)
 - provide high availability of services (servers don’t block)
 - control execution of a piece of code (sleep and wait/notify)
 - improve performance (parallel execution with multiple processors)
- Synchronization
 - allows objects to maintain a consistent state
- Priorities
 - indicate execution order preferences that the thread scheduler can use



Java Beans

- Naming convention for accessor methods
 - get, set, is
 - for example
 - `getRentalPrice()`, `setRentalPrice(int price)`, `isRented()`
- Listening for property changes
 - `PropertyChangeListener`
 - can register to listen for changes in the “properties” of other objects and be notified when they change
 - for example, listen for `rented` property to be set to `false`, meaning a car has been returned, so that a maintenance check can be scheduled
 - `VetoableChangeListener`
 - can register to be given the opportunity to “veto” proposed changes to the properties of other objects
 - for example, listen for `rented` property to be set to `true`, meaning a car is being rented, and the rental can be vetoed if the maintenance history contains an unresolved problem



Ways of Creating Graphical User Interfaces (GUIs)

- browser-based
 - servlets, JSP, XML/XSL
- AWT
- Swing
- 2D API
- 3D API



AWT & Swing

- AWT (Abstract Windowing Toolkit)
 - original package for creating graphical user interfaces (GUIs)
 - limited set of GUI components (basics like buttons and text fields)
 - supported by Netscape and IE for writing applets
- Swing
 - newer package for creating GUIs
 - many more GUI components
 - combo boxes, tabbed panes, split panes, tables, trees and more
 - more containers
 - can create multiple document interfaces (MDI)
 - supports model-view-controller development
 - supplies model interfaces, abstract classes and default classes
 - supports pluggable look-and-feel



Graphics

- AWT
 - draw lines, rectangles, polygons, ellipses and images
 - create animations
 - create custom GUI components
- 2D API
 - better support 2D graphics, image processing and fonts
 - Shape interface and many implementing classes
 - customizable line widths, ends, joins and dash patterns (Stroke)
 - gradient and pattern fills (GradientPaint & TexturePaint)
 - blend colors of shapes, text and images (called “compositing”)
 - transform shapes, text and images (AffineTransform)
 - user-defined coordinate spaces (AffineTransform)
- 3D API
 - support for 3D modeling and virtual worlds (similar to VRML)



Ways of Providing Persistence

- Object serialization
- Java DataBase Connectivity (JDBC)
- eXtensible Markup Language (XML)
- Java Naming and Directory Interface (JNDI)



Object Serialization

- Allows objects to be saved to and restored from streams
 - includes files and sockets
 - saves all objects reachable from a serialized object
- Features
 - usable for all objects with little customization
 - just have to implement Serializable
 - extensible
 - can customize what is written and read and how (ex. compression and encryption)
 - supports persistence
 - saving the state of a collection of objects and restoring that state in a future session
 - provides security
 - through ability to customize what is written (ex. can suppress credit card numbers)
 - supports marshaling and unmarshaling for RMI (covered later)



Java DataBase Connectivity (JDBC)

- Standard interface for accessing data sources
 - typically used to access relational databases
 - utilizes SQL
 - can change database vendors with little impact on code
 - patterned after Microsoft's ODBC
 - supports transactions and stored procedures
- Implemented by
 - data source specific JDBC drivers
 - accept JDBC calls and perform operations using the API of the specific data source
 - JDBC-ODBC bridge
 - special JDBC driver included with JDK that can access ODBC compliant data sources



eXtensible Markup Language (XML)

- Primary benefits
 - data portability
 - passing data between processes/platforms
 - separating data from presentation
 - can “transform” into HTML and other formats
- Many XML tools are written in Java
 - examples are
 - Xerces - an XML SAX and DOM parser from Apache
 - Xalan - an XSL processor from Apache
- Browser support
 - Internet Explorer 5
 - Netscape 6

can do all XML
processing on server



eXtensible Markup Language (Cont'd)

- Key aspects
 - XML documents
 - portable way of representing data
 - composed of custom tags (elements) that look similar to HTML tags
 - DTDs and XML Schemas
 - used to enforce rules on what is allowed in specific XML documents
 - CSS and XSL style sheets
 - used to format XML data for browser display and other forms of output
 - SAX and DOM parsers
 - used to process data in XML documents and create new XML documents
 - browser and server-side processing
 - XML processing can be performed in browsers that support it
 - can perform processing on server-side to avoid browser compatibility issues



Java Naming and Directory Interface (JNDI)

- Portable way to access naming and directory services such as LDAP
 - naming and directory services are ideal for data that
 - is hierarchical in nature
 - for example, HR data and company printers
 - does not change often
 - very efficient for reads, not so efficient for writes
 - can change vendors with little impact to code
 - similar to JDBC in this respect
 - example products are Netscape Directory Server and Novell's NDS
- Also used to locate EJBs (covered later)



Ways of Creating Distributed Applications

- Sockets
- Common Object Request Broker Architecture (CORBA)
- Remote Method Invocation (RMI)
- Enterprise JavaBeans (EJB)



Sockets

- Low-level method of sending and receiving data between processes running on same or different hosts
 - can be used to communicate with non-Java applications
- Server processes “listen” on a particular port
- Client processes establish a connection to that port
- Two protocols supported
 - TCP (Transmission Control Protocol)
 - reliable, slower, connection based (verifies receipt and won't lose data if multiple clients send data at the same time)
 - UDP (User Datagram Protocol)
 - unreliable, faster, connectionless based (doesn't verify receipt and can lose data if multiple clients send data at the same time)



Common Object Request Broker Architecture (CORBA)

- Language independent, distributed object model
 - can locate remote objects and invoke methods on them
- Many ORBs support Java bindings that
 - allow Java applications to invoke CORBA services
 - allow CORBA services to be implemented in Java
- Somewhat more complex than RMI and EJB
- Specification has been maturing for many years
 - Object Management Group (OMG)
 - started in 1989
 - over 600 member companies
 - many capabilities have been defined
 - naming service, automated launching of servers, transactions, ...



Remote Method Invocation (RMI)

- Java-only, distributed object model
- Relatively simple to use
- Shares many concepts with CORBA
 - registry lookup vs. naming service
 - stubs
 - client-side proxy for a remote object
 - marshals arguments, forwards method calls, unmarshals return values and exceptions
 - created when a reference to a remote object is passed to a client VM
 - lookup or method return
 - generated by rmic compiler
 - skeletons
 - server-side object that dispatches calls to corresponding remote object methods
 - unmarshals arguments, calls remote object methods, marshals return values and exceptions
 - generated by rmic compiler



Enterprise JavaBeans (EJB)

- Framework that provides
 - persistence - typically to relational databases
 - transactions - across multiple data stores on multiple hosts (distributed)
 - security - specifying who can create/delete entities and call methods
 - scalability
 - EJBs can be distributed
 - servers can provide redundancy and failover
 - database connection pooling
 - reduces number of concurrent open connections required
 - EJB object lifecycle management
 - instance pooling (passivation & activation), timeout (passivation)
- Allows developers to focus on writing business methods
 - hides framework complexity
 - removes need to write code that provides the features above

EJB interfaces are defined in terms of RMI but EJB implementations can be built on RMI's Java Remote Method Protocol (JRMP) or CORBA's Internet Inter-ORB Protocol (IIOP)



Enterprise JavaBeans (EJB)

- Types of EJBs
 - entity beans
 - associated with a data store - typically a row in a relational database table
 - two types: bean managed persistence (BMP) and container managed persistence (CMP)
 - survive server crashes
 - can be used by multiple clients
 - session beans
 - often used to interact with multiple entity beans on server-side; reduces network traffic
 - two types: stateful and stateless
 - don't survive server crashes
 - associated with a single client session
- Home interfaces
 - provide methods to create and find EJBs
 - especially important for entity beans since they are associated with a data store



Summary

- Java is one of the easiest object-oriented languages to learn
- Java is portable to a large number of platforms
- Java boasts big developer productivity gains
- Java comes with a larger class library that can drastically reduce the amount of code that developers must write
 - distributed applications (sockets, RMI, CORBA, EJB)
 - GUI applications (browser-based, AWT, Swing, 2D API, 3D API)
 - multi-threading
 - data persistence (serialization, JDBC, XML, JNDI)
 - more
- The Java talent pool is growing as new college grads come out already knowing how to use it

