

MSIS 531
Foundations of E-Business
Russ Fish

Spring 2006

2

Today's Objectives

- Class logistics
 - Overview, syllabus, grading,...
- Tech stack
 - Pieces of the puzzle we'll be looking at during the quarter
 - Goal: no installs; everything copied on (and off)
 - Secondary goal: enable deliveries throughout the quarter
- Java overview
 - Focus will be at the "1000-foot" view
- J2EE, Servlet overview
 - Provide context for future class sessions

My Background

- Computer science undergrad, UW MBA
- 25 years IT experience; primarily in systems management, programming
- Interests in RDBMS management, application programming
- Lecturer at UW since Fall 1997
- Other job: Oracle DBA/Unix system administrator
 - Manage DBAs in Boston, MA, Burlington, VT
 - Technology at the “five foot level”
- Five years into a PeopleSoft (Oracle) ERP implementation
- Company (IDX) acquired by GE Healthcare 1/06

Class Learning Objectives

- Overview of Java programming language
- Java presentation technologies
 - Servlets, JSF
- Java database connectivity
 - JDBC, Hibernate
- Application frameworks
 - Spring
- Web services in Java

Course Structure

- Lectures
 - Class participation on papers, topics: 10%
- Assignments
 - Four assignments, worth 35%
- Exams
 - Short answer, programming problems, etc.
 - 1st Exam 25%
 - 2nd Exam 30%

Textbooks

- Just Java – van der Linden
 - Terrific Java resource, covers JDK 1.5
- Core JavaServer Faces – Geary/Horstmann
 - Good intro coverage & more detail if you want it later
- Hibernate: A Developer's Notebook – Elliott
 - ADN books offer quick way to “deep dive” into Hibernate
- Spring: A Developer's Notebook – Tate/Gehtland
 - Starting point only: Spring is a complex topic (to borrow from Churchill, it is a riddle, wrapped in a mystery, inside an enigma)

Use of Class Time

- Conflicting goals:
 - Lot of material and not much time
 - Building stuff teaches the most
- In each class session we'll do the following:
 - About an hour of lecture
 - 30 min. of implementation
- Goal is to treat the class as a seminar
 - Interactive, discussion-based

Why Java?

- Platform independence/portability
- Fully object oriented language
- Designed to make programming easier and safer
- APIs for everything
 - See e.g. Table 1-1, JJ p. 11
- Built-in networking, threading
- Variety of platform options (J2EE/SE/ME)
- Many open-source options available
 - Hibernate, Spring

The Java Virtual Machine

- Java source code -> binary (byte) code
- Byte code format is universal (system independent)
 - Same on Windows, Mac, Unix
 - Interpreted, not compiled
 - Set of instructions for a virtual machine
- Anyone can write a VM; VM and language specs are exhaustive
- JVM is lightweight, system dependent
 - Really does allow “write once, run anywhere”

Java Classes

- Fundamental unit of Java code and associated data
- Abstraction of a thing of interest
 - Employee, Sales Order, Inventory Item
 - Shared characteristics: actions, attributes
- A sort of “software blueprint”
- In use, treated as a “black box”

Java Execution

- Perception: slower than compiled languages
- Fast interpreted language
- Some tricks possible to speed execution:
 - Just-in-time compilation: compile byte code to native (machine-dependent) code
 - Profiling: find often-used portions of a program and compile them to native code
- Result—recent Java versions are [pretty fast](#)

Java vs. Other Languages

- Many similarities with C, C++, C#
- Good if cross-platform support required
 - Example: Oracle's java-based applications are delivered to about 60 platforms
- Business benefits:
 - C/C++ programmers find Java syntax easy to follow
 - Fully object-oriented
 - Compact, easy to extend language
- And to get it out in the open:
 - Java is not the same as JavaScript

Java Language Design

- Goal: safe, extensible development
- Addresses common design, programming problems
 - Safe(r) from simple mistakes
- Static typing: the data type of every data element is known at design time
- Late binding: method (code) lookup occurs at runtime

Java Language Design (cont.)

- Garbage collection: JVM cleans up data that is no longer in use
- Safe pointers: no C/C++ pointer tricks
 - Java calls these references
- Exception handling: robust error processing
- Multithreading: built-in support for parallel task execution

Java Implementation Safety

- Encapsulation: object implementation, data hidden from user
- Byte code verifier – enforces security
- Class loader – enforces class locations
- Security Manager – controls access to system resources (files, devices, etc.)
- Digital signatures – sign classes to ensure appropriate use

Initial Setup/Installation

- Everything copies from the CD
 - In subsequent classes, we'll work from new versions of the CD
- Windows batch files to set path entries
- Technologies (for now):
 - JDK 1.5
 - Netbeans 5.0
 - Ant 1.6
 - Tomcat 5.5
- Install doc available separately