COURSE: COMP 303

Java Enterprise Edition Programming

Developing for Enterprise Environments

Professor Info

- Professor: Yilmaz Cam
- e-Mail: ycam@my.centennialcollege.ca
- Link to his resume
  - https://1drv.ms/b/s!Ap0IyG6v0f9UiSAnwP-GDJwtvGQY

COMP303 is for programmers

COMP303 is a challenging course with advanced content

- Prerequisites:
  - COMP214 Advanced Database Concepts
  - COMP228 Java Programming
- Strongly recommended
  - COMP212 Programming 3
  - Recent use of Java
- You should feel confident about your ability to:
  - Write good Java code
    - Follow Java coding conventions and apply OOAD
    - Apply you experience and expand your knowledge of Java SE API
  - Use relational databases with SQL and from Java with JDBC
  - Create HTML documents with a flat text editor
  - Understanding how Web sites work
    - Role of the server, http protocol ...

Enterprise software environments

Middle to large-size companies tend to have:

- Large distributed systems
- Systems built from heterogeneous components often required to
  - Integrate software acquired through mergers and acquisitions
  - Web-enable legacy code
  - Wrap existing component in reusable services
- Different kinds of clients
  - Mix of end users and separate interacting systems
- Need for high availability, strong security, good performance …
- Code that must be maintainable, extendable, modifiable …
- Very separate development and production processes
  - Scattered development teams
  - Production servers and administrators in several locations
- Established processes required for stability
  - challenge of adapting development process to new technology
- Middleware ties the enterprise together

Scope of this course

At the end of this course you should be able to design and build small applications and deploy them to the runtime environment of a Java EE Application server.

- **COMP303 covers many technologies from Java EE but not all:**
  - Web apps where clients are end users at Web browsers are covered but not:
    - Web services or web service clients, including RESTful applications
    - Java API for JSON (JavaScript Object Notation used with REST)
  - Asynchronous processing
    - Web sockets (new in Java EE 7)
    - JavaScript and AJAX (not part of JEE but can be used with Web apps)
  - Message Driven Beans and the Java Message Service
    - MDB are EJBs that receive asynchronous requests from Message Oriented Middleware
  - Batch mode operations
    - Support for online transaction processing (OLTP) new in Java EE 7
  - Advanced features: new API for concurrency; management API …
  - Security beyond a mere mention
    - Security is more a concern of production team than of developers

Learning Objectives 1/3

- Upon completion of this course, you should be able to:
  - Explain how the runtime environments provided by Java Micro, Standard and Enterprise Editions are different
  - Use the Java SE API for:
    - Miscellaneous programming
    - (collections, TCP/IP sockets, simple multi-threading, …)
    - Apply design patterns such as the singleton
  - Name tiers in the Java EE n-tier architecture and explain how Java Application Servers support distributed applications
    - Use several component types supported by Web and EJB containers
    - Describe services provided by the Web and EJB container

Learning Objectives 2/3

- Upon completion of this course, you should be able to:
  - Design, code and test Java Web applications based on
    - Basic Web Apps: Servlets and JavaServer Pages (JSP)
    - Java Server Faces: Components library and HTML 5
  - Apply design patterns such as MVC
    - Design and implement layered applications
  - Package application components for deployment
    - Java archive (JAR)
    - Web archive (WAR)
    - Enterprise application archive (EAR) files.
  - Explain the relationship between JEE deployment descriptor files and Java annotations

Learning Objectives 3/3

- Upon completion of this course, you should be able to:
  - Use the Java Persistence Architecture (JPA) to
    - Provide a persistence layer that performs object-relational mapping
    - Access persistent data (relational databases) from application code
  - Describe the role of Enterprise Java Beans (EJBs) in Enterprise application Integration (EAI)
    - Use stateless session EJBs as facades for JPA entity beans
  - Discuss factors for success in enterprise applications
    - Reliability, Scalability, Performance, Adaptability, Interoperability…
  - Develop and package Java-based applications using
    - Eclipse for development
    - Wildfly (JBoss) application server for deployment
    - MySQL database server for persistence

Grades and Marking

Deliverable	%
Programming assignments (4)	40
Mid-Term Test	20
Project	20
Final Exam	20
Total	100

- Most assignments are done in pairs.
- Project is a team effort
- Mid Term and Final Exams are individual effort
- Mid Term includes closed book and open book sections.
- Attendance contributes your final mark positively or negatively depending on your presence in the sessions

References

- No book purchase is mandatory
- For Java SE:
  - Java SE Java doc – API reference in javadoc format
    - View on line or download from http://docs.oracle.com/javase/8/docs/api/
    - Largely duplicated in code-assist built into Eclipse
  - Java SE Java tutorials
    - Excellent overview of API, with example, grouped by topic
    - View on line or download from http://docs.oracle.com/javase/tutorial/
- Also use your text from COMP228 or any Java Programming book you like

References

Java EE 7: The Big Picture

by Dr. Danny Coward
Publisher: Oracle Press
Date: October 2014
ISBN: 9780071837330

Java EE Tutorial, 2 volumes

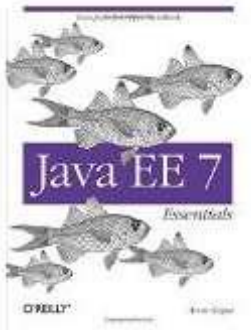
by Eric Jendrock
Publisher: Addison-Wesley
Date: May 2014
ISBN: 9780321994929

Java EE Tutorial online

https://docs.oracle.com/javaee/7/tutorial/

Java EE 7 Essentials

by Arun Gupta
Publisher: O'Reilly
Date: September 2013
ISBN: 978144930176



Also recommended

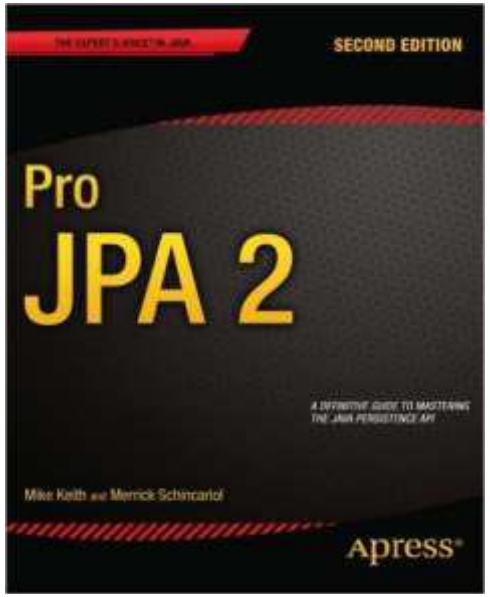
On the Java Persistence Architecture:

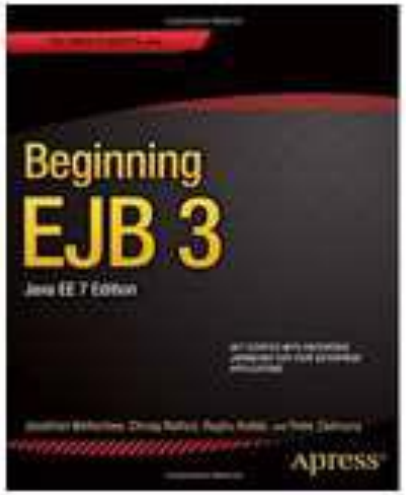
Pro JPA 2 Second Edition

By: Mike Keith and Merrick Schincariol

Date: Apress, September 2013

ISBN : 1430249269





Text listed in the official outline

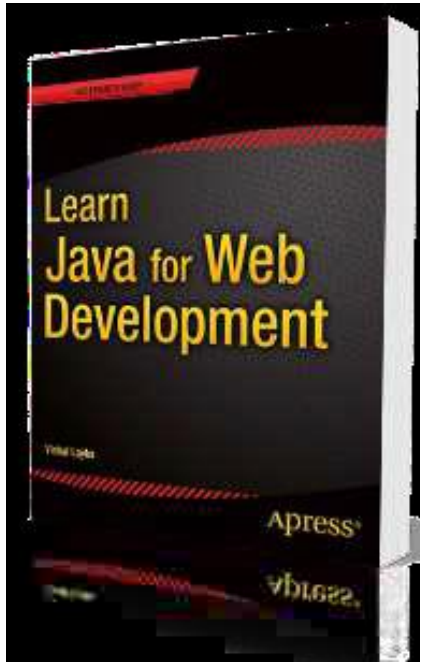
Learn Java for Web Development

by Vishal Layka

Publisher: Apress

Date: February 2014

ISBN: 987143025983-1



- Summary/comparison of different Java-based technologies for building Web Apps
  - Chapters 1, 2, 3, 6 chapters useful for this course
  - Alternative technology covered: Struts 2, Spring, JSF 2, Grails, Scala
    - Not Java EE standards but widely used, especially the Spring Framework
  - More an overview and summary than teaching text
- Covers only Web user interface to enterprise applications

WEEK 1
COURSE OVERVIEW

Editions of the Java platform

The JDK and JRE

Java coding conventions

JavaBeans

Editions of the Java platform

- Java Standard Edition (SE)
  - A runtime platform for developing and running Java applications.
  - It includes database access, CORBA interface technology, and security for local networks and the Internet.
  - SE is the core Java technology platform.
  - Used for stand-alone Java applications and applets.
- Java Enterprise Edition (EE)
  - A runtime platform used for developing, deploying, and managing multitier, server-centric applications on an enterprise-wide scale.
  - Builds on the APIs of Standard Edition.
  - It includes distributed communication, threading control, scalable architecture, and transaction management.
- Java Micro Edition (ME)
  - Designed for embedded devices and consumer products

Java SE and Java EE

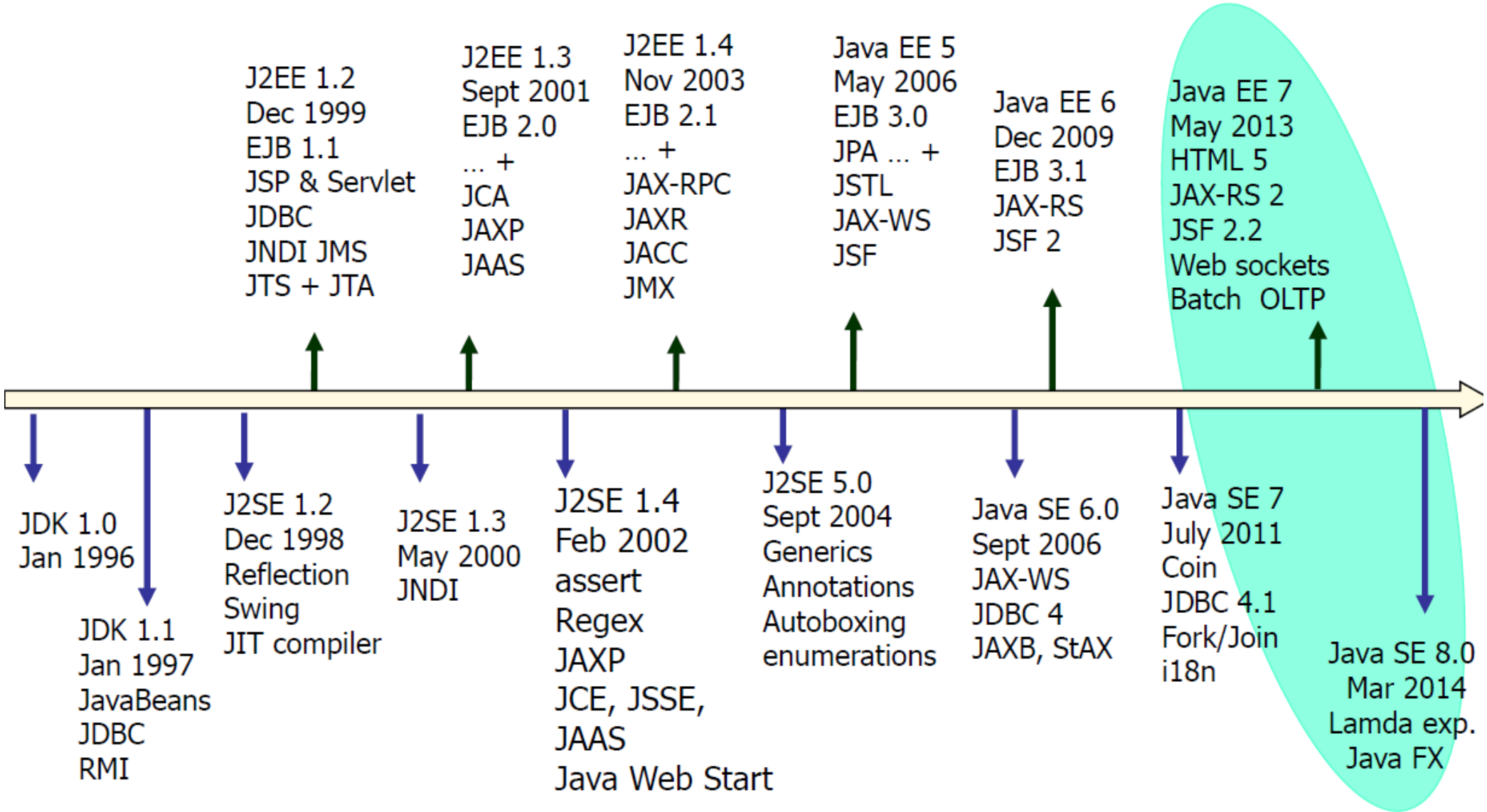
- Java SE usually downloaded:
  - Java Runtime Environment (JRE) to run programs
    - Java Virtual Machine (JVM) runs on native operating system
  - Java Development Kit (JDK) aka (SDK)
    - Command-line tools to compile and develop Java Programs
- Java EE usually purchased from vendor:
  - Application Server provides runtime environment
    - Application components must be packaged into archives that are then installed an started on the server
  - Includes a JVM all of Java SE + EE
  - Conforms to all Java EE standards
    - Performs all specified services in a common manner
    - Provides standard API for programmable access
  - Often has vendor-specific enhancements

Installing Sample Projects

- **Code from all tutorials is available online**
  - http://courses.coreservlets.com/
    - Click on Java tutorial on top left of page
- **Import project into Eclipse**
  - Click on appropriate tutorial section
  - Download ZIP file
    - The one for this section is called “intro”
  - Start Eclipse and go to Workbench
  - File → Import → General → Existing Projects into Workspace → Select archive file (not “Select root directory”).
    - Then browse to ZIP file you downloaded, OK, Finish

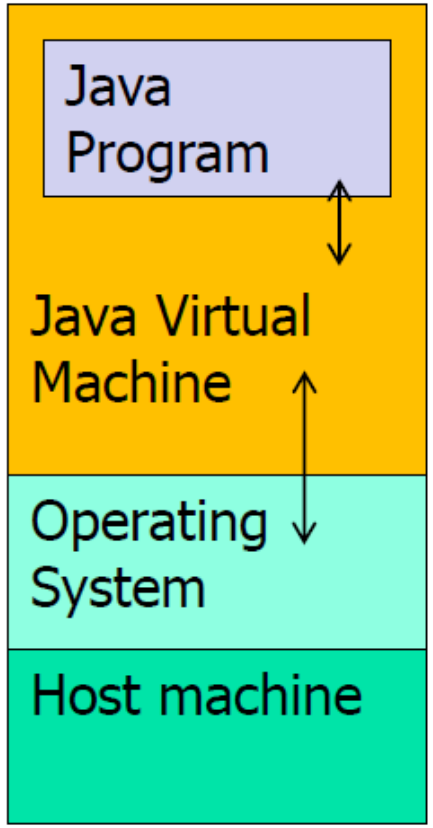
A timeline: The evolution of the Java platform

- This course is based on Java EE 7 and Java SE 8

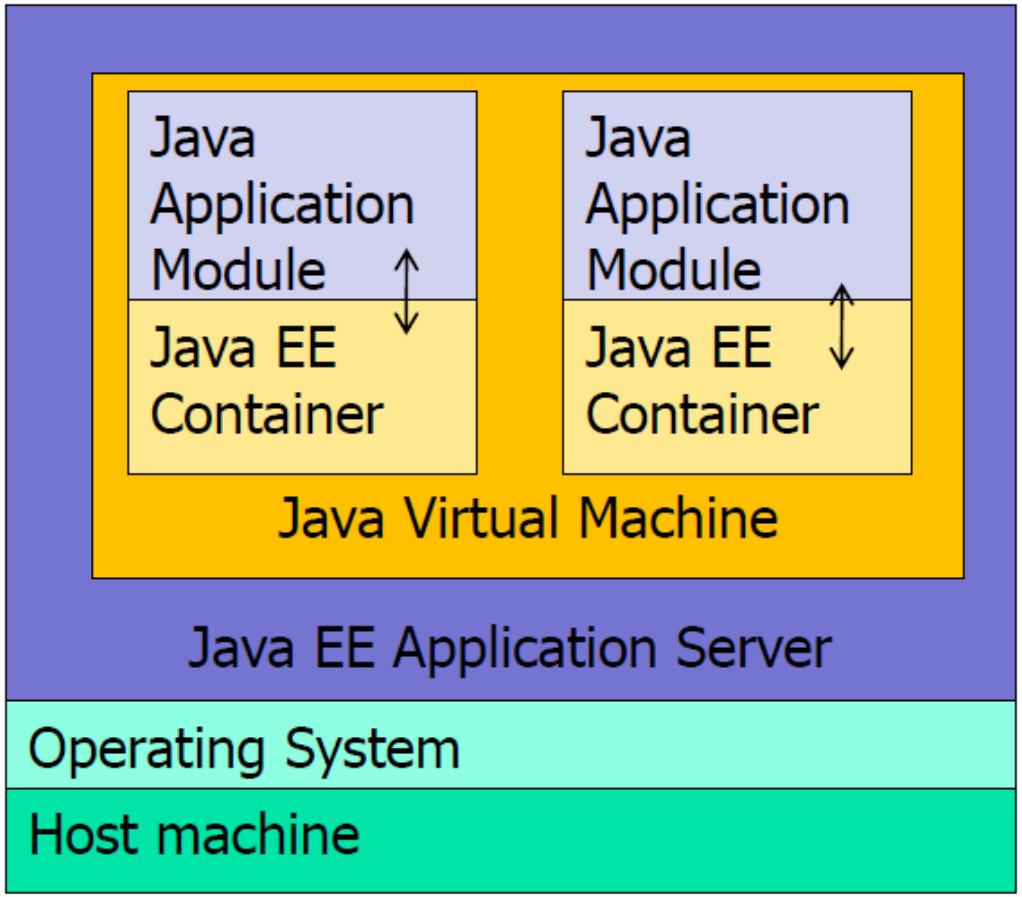


Java Runtime Environments

- Each edition of Java is designed to deploy to a different runtime environment.



Java SE



Java EE

Included in the SDK (SE 7)

- Base Libraries
  - Java language implemented in java.lang
- Java I/O, Swing and AWT for GUIs
- Common Object request Broker Architecture
  - Used for remote method invocation (RMI)
- Java Virtual Machine: runtime environment
- Java Database Connectivity (JDBC)
- Java Networking API
- Java Security API
- API building and parsing XML documents
- Many more tools and API

Running Java SE programs

- To run Java SE programs from and IDE
  - Create classes in a Java Project
  - All types defined by the JRE are automatically available
  - Add jars containing additional classes required to the build path (Java classpath)
  - Use the Run feature for the IDE
- To build on using the SDK
  - Compile each class with javac command
  - Combine bytecode into a JAR using the jar command
    - Identify the main class in file MANIFEST.MF in JAR
- To run using the JRE
  - Ensure all required types are on the classpath
  - Launch the main class with the java command
    - >java MyJar.jar

Deploying Java EE programs

- Typically the developers have access to a development server, often installed on the local machine
  - May not have full power/functionality of production environment
  - Eclipse can communicate with a wide variety of servers so you can build and deploy through the development environment
- For deployment to a production server:
  - Developers export completed components into archives
  - Build team (not always same people as developers) combine archives into enterprise applications
  - Server administrators manage the enterprise application. They:
    - Install, Start, stop, update, uninstall … as required
    - Configure resources, such as databases, messaging queues … as specified in deployment descriptors
    - Establish security
    - Tweak server settings for performance

Developers can and sometimes should have responsibilities for security and performance

Classroom Platform

- Integrated Development Environment
  - Eclipse IDE for Java EE Developers
- Testing runtime environment
  - Wildfly Application Server
    - *IDE and server bridge through* JBoss tools for Eclipse
- Production runtime environment
  - Not provided: any Java EE Application Server
- Database
  - MySQL database community edition

Windows 7 on 64-bit architecture

- Software install on student laptops/desktops recommended

Review of the basics

- Java Coding Conventions
- Java Beans
- Singletons

Java coding conventions 1 of 3

A few of many commonly accepted conventions

- Package names
  - All lower case, separate words with .
  - Consider domain-based prefix to avoid name conflicts
    - Example: com.cencol.sws311.asgn5
- Declarations
  - One per line
  - Place at the start of blocks except loop counter
  - Initialize variables in declaration
    - statics may also be initialized in a static initializer block
- Statements
  - One per line
  - Always use { and } after if
- Omitted from this list:
  - Code layout : line length, indenting, blank lines, white space …
  - Comments : // one line /* block comments */ /** javadoc **/

```
static private byte[] vowels
        = {'a','e','i','o','u'};

public String justVowels(String input) {
       StringBuffer output = new StringBuffer();
       for ( int i=0; i< input.length(); i++) {
         char c = input.charAt(i);
         // compare Strings with .equals()
         if (c == vowels[i] ) {
                 output.append(c);
         }
        }
        return new String(output);
}
```

Java coding conventions 2 of 2

- Naming conventions

Identifier type	Guidelines	Examples

Class	Noun
	Start with capital, then lower case except 1st letter of words
		`MarketManager`
		`Ticket`
		`RasterImpl`

Interface	Start with capital, then lower case except 1st letter of words	`Marketable`
		`ImageMaker`
		`Raster`

Method	Verb
	Lower case except 1st letter of internal words
		`fly()`
		`getHistory()`
		`colourMePurple()`

Variables	Lower case except 1st letter of internal words	`price`
		`firstName`
		`ticket`

Constants	All upper case with _ to separate words As of Java SE 5 use Enum type	`MAX_MARK` `public enum Colour { RED, YELLOW, GREEN }`

Java coding conventions 3 of 3

- Miscellaneous
  - Be consistent with placement of { } and other layout options
  - Add parenthesis liberally to improve readability
    - Examples:

```
larger = (a < b ) ? b : a;
if ( (answer != null) && (answer.length() > 0))
```

  - Avoid hiding variables by reusing identifiers

```
String answer = "done";
if ( more ) {
    String answer =""; // hides answer with value “done”
    … // build answer
}
```

  - Avoid * imports

```
import java.util.* // be more precise
```

- Order of class members:
  - Class variables (static fields) - use sparingly
  - Instance variables: protected, private - should not be public
  - Constructors
  - Methods: public, protected, private

Classes and Interfaces

Consider the main role/responsibility of each type

- Classes
  - Some classes are primarily for data objects
    - Encapsulate fields
    - Provide accessor / mutator methods (get/set)
    - Perform minimal business operations
      - Data validation in mutators is acceptable even desirable
  - Some classes manage aggregates of data objects
    - Add/update/delete/retrieve data objects
  - Some classes implement behaviour to provide functionality
    - Fields used to support behaviour
  - Some classes support application infrastructure
    - Helper classes
- Interfaces define behavior
- Enums define constants and can add some behavior

Java Beans

- Try to write classes that are good Java Beans
  - Java Beans is a coding standard
    - Ubiquitously used for data objects
      - Fields are private or protected
      - Fields have accessor / mutator methods as required

```
public class MyClass {
    protected String description = null;;
    protected boolean current = false;

    public String getDescription() {
         return description;
    }
    public void setDescription( string desc ) {
         if ( desc == null || desc.length() < 1 ){
            throw new InvalidSettingException();
         }
         description = desc;
    }
    public void setCurrent( boolean current) {
         this.current = current;
    }
    public boolean isCurrent() {
         return current;
    }
}
```

```
Prototypes for set and get
Methods for field xyz
with type <T>

public void setXyz( <T> xyz );

public <T> getXyz();

public boolean isXyz();
```

Java Beans in specialized contexts

- Some tools generate/consume/use beans
  - Beans often used in Web applications
    - To transfer date between server-side logic (model layer) and browser-displayable pages (view layer)
    - Java Server Pages have special tags for manipulating beans
  - Java Server Faces introduce concept of managed beans
    - Follow additional JavaBeans conventions for event handling such as property change notification
- Bean class must have default constructor
  - No arguments so tools or framework can generate object
- Bean class should be Serializable
  - So objects can sent over communications line / stored and retrieved later

The full Java Bean story

- Java Beans and Enterprise Java Beans (EJB) are not related
  - Java Beans is a Java SE convention for coding classes
  - EJB are a Java EE component type that require environment provided by Java EE Application Server

- See the Java Beans tutorial:
  - http://docs.oracle.com/javase/tutorial/javabeans/
  - Beans have properties, methods and events
  - BeanInfo class is a companion class for tools that build Java UI from AWT/Swing components

First Lab: Review Java SE

- Setup Eclipse
- Import Eclipse projects into Eclipse IDE
- Review Java SE code
- Change and run code using Eclipse

Summary

- **Downloading Java**
  - http://www.oracle.com/technetwork/java/javase/downloads/
- **Bookmarking the Java API**
  - http://docs.oracle.com/javase/8/docs/api/ (or …/7/…)
- **Downloading Eclipse**
  - http://eclipse.org/downloads/
- **Downloading sample projects**
  - http://www.coreservlets.com/
    - Click on Java Programming tutorial on top left
  - Import with File → Import → Existing Projects …
- **Executing a class that has "main"**
  - R-click in code, Run As → Java Application