Intro to Java

CSE 413, Autumn 2002
Programming Languages

http://www.cs.washington.edu/education/courses/413/02au/
Readings and References

• Reading
  » Chapters 1 and 2, (Intro to Java, Java Programming Environment), *Core Java Volume 1*, by Horstmann and Cornell

• Other References
  » "Object-Oriented Programming Concepts", Java tutorial
    » http://java.sun.com/docs/books/tutorial/java/concepts/index.html
What is Java?

• An object-oriented programming language
  » source code

• Application Programming Interfaces (APIs)
  » extensive class libraries

• A virtual machine
  » runs programs that were written in the source language and compiled to binary bytecodes
The Virtual Machine concept

- 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 explicitly support pointers or any other direct access to memory
- Java is automatically garbage-collected, so explicitly de-allocating memory is not necessary
- Java is interpreted. It is difficult (and in fact not part of the language) to compile to native machine code
- Java is dynamically linked, with run-time polymorphism
Java Environments

- Sun has developed subsets of the Java platform
  - Java Enterprise Edition
    » servers
  - Java Standard Edition
    » desktop
  - Java Micro Edition
    » mobile devices
Java Developers Kit (JDK)
Tools in the JDK

- **javac** - Java compiler
- **java** - Java interpreter
- **jdb** - Java debugger
- **appletviewer** - viewer for Java applets

- **javap** - Java bytecode disassembler
- **javadoc** - Java documentation generator
- Documentation for the JDK can be explored with your Web browser
Installing the JDK

• Instructions on the class software page
• JDK
  » tools
  » library sources
• Java API documentation
• Learning and reference materials
  » Java tutorial
    http://java.sun.com/docs/books/tutorial/
  » take the time to set up one-click shortcuts now
Our Environment

Development Environment
jEdit, jPad, command line

compiler
javac.exe

class definition
source file: Dog.java

java virtual machine (JVM)
java.exe

class definition
binary file: Dog.class

output

you and me
The compiler reads our source file and produces a binary class file.

Compile it

class definition
source file: Dog.java

Development Environment
jEdit, jPad, command line

compile it

compiler
javac.exe

java virtual machine (JVM)
java.exe

output

class definition
binary file: Dog.class

binary class definition

you and me

you and me
Run it

The virtual machine executes the instructions in the class definition to produce the output from our program.
Objects and Classes

• A class is a definition of a *type of thing*
  » The class definition is where we find a description of how things of this type behave.

• An object is a *particular thing*
  » There can be many objects of a given class. An object is an *instance* of a class.
  » All objects of a given class exhibit the same behavior.
What is a Java class?

- A class is a *template* or *blueprint* for building objects.
- A class is like a dictionary definition, while objects are like things in the real world that “are” whatever is defined.
- A class definition generally resides on disk long term:
  - the original class definition is written in Java (the .java file) then translated into a more compact form (the .class file) by the compiler.
  - the class definition can be used over and over to create more objects, just like a blueprint can be used over and over to build more houses.
- An object resides in memory and is generally discarded during or at the end of a program run.
Houses are instances of blueprints
Instantiate - create an object

• Once we create a class definition using an editor and the compiler, we can instantiate it with the "new" operator
  » to instantiate means to create objects based on the class definition
  » Oval moon = new Oval(100,100,20,20,Color.gray,true);

• We can then manipulate these objects to do the work that needs to be done

• Note that many classes have already been defined for us
  » There are 2723 classes defined in the standard Java libraries from Sun - see the JavaAPI documentation
Class Concepts

• Class definitions have two important components:
  » state
  » behavior or interface

• State is expressed using fields in the class definition

• Behavior is expressed using methods

• Together, fields and methods are called class members
Class Concepts: State

• State is a complete description of all the things that make a class a class.

• For example, part of the state of class Employee is the Employee’s UWNetID
  » All objects of class Employee will have a UWNetID specified.

• State is stored in data members
  » also known as fields, member variables, instance variables, properties
Class Concepts: Behavior

• Behavior of a class defines how other classes may interact with it. It indicates the capabilities of the class to “do” things.

• For example, a BaseballPlayer class might define such behavior as hit, pitch, stealBase, etc.

• Behavior is defined in methods
  » Methods look like functions in C, methods in C++, subroutines in Fortran, etc
Structure of Source File

- Source file must have same name as name of public class it contains
- Simple structure in order
  » package definition (Optional)
  » package and/or class import statements (Optional)
  » Class definition (multiple are allowed but can be messy)
Structure of Source File

Three components to a Java source file, in order

- Package identifier
- Import statements
- Class definition

```java
package package.name;
import java.io.*;
import java.util.ArrayList;
public class MyClass {
    // members go here
}
```
Packages

- A way to group related classes
- A key part of Java’s encapsulation mechanism
- 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
import - help the compiler find classes

• A class’ full name includes its package.
  » java.util.ArrayList or java.io.FileReader
• Usually it is more convenient simply to use the class name without the package
• 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++
Example class

```java
public class Dog {
    public Dog(double rate) {
        consumptionRate = rate;
        weight = 20;
    }
    public void bark() {
    }
    public double getRate() {
    }
    public void eat(double pounds) {
    }

    private double consumptionRate;
    private double weight;
}
```
Basic Libraries Sample Members

- `java.lang` - Object class, numbers, strings, System, Exceptions, Threads and more
- `java.io` - streams, readers, writer, files
- `java.util` - Dates, Locales, data structures, zip
- `java.net` - Sockets, URLs, datagrams, InetAddresses, connections
- `java.awt`, `javax.swing` - Graphics, Layout, Event, User Interaction
Documenting Source Code

- // - single line comment
- /* multiple line comment */
- /** javadoc style comment */
- javadoc utility provides automatic generation of documentation from code comments
Javadoc Tags

• The javadoc utility supports several “tag” fields in javadoc comments
  » @param -- passed parameter description
  » @return -- returned value description
  » @throws -- error indicators

• javadoc formats these and includes them in the generated documentation