

Notes on the Java Programming Language

Section 1: Java Fundamentals

1) Java

- a) native object-oriented language
- b) C++ like syntax
- c) created at SUN corporation in 1990s
- d) compiles to binary code (called byte code) that generally must be interpreted
- e) uses Unicode for character set
- f) free Java compilers for most operating systems are available at <http://java.sun.com>

2) Java Compiled to Binary in two Forms

- a) Applet
 - i) binary code file ending in .class started via the Applet tag in an HTML file
 - ii) run via Appletviewer or browser
 - iii) Applet tag:

```
<applet code="code-file" width=#pixels height=#pixels></applet>
```

example:

```
<applet code="Greeting.class" width=300 height=200></applet>
```

- b) Free Standing Programs
 - i) Binary code file ending in .class run via the "java" interpreter

3) Naming and Compiling Java Files

- a) files of Java code must be named with extension ".java"
- b) to compile Java code, use javac compiler: `javac filename.java`
- c) compiling Java code in a file creates a file called "fileName.class"

4) Running Java Programs

- a) To run Applet from command line: `appletviewer fileWithAppletTag.html`
- b) To run Applet from browser: Use URL or File and Open
- c) To run free standing program: `java fileName.class`

5) Basic Java Program Makeup

```
imports  
class definition
```

6) Importing Libraries and Packages

- like C++, Java is a small language augmented by pre-defined and user created code
- libraries are generally predefined classes precompiled classes that may contain other classes, etc.
- packages are programmer created code are stored in a directory with the packages name
- the subclasses or subparts of libraries or packages are accessed (brought into a Java program) via the dot syntax of the import statement:

```
import library-or-package.subunit(s);  
import library-or-package.subunit.subunit(s);  
import library-or-package.subunit.subunit.subunit(s);  
etc.
```

examples (all of common Java libraries):

```
import java.awt.*;  
import java.applet.Applet;  
import java.awt.event.*;
```

Note: The asterisk means get all subunits of the preceding class or subunit.

7) Basic Java Class Structure

```
class header  
class body
```

8) Java Class Header

```
access class class-name [extends class[, class]]  
                        [implements interface[, interface]]
```

9) Access

- public** - allows outside of class access as well as access inside of class
- private** - allows only inside of class access, private may not be inherited
- protected** - acts like private in class, but when class inherited (via extends), becomes private in the inheriting class; i.e. it is a way of providing inheritance to otherwise private items

10) Class Name

A class within a ".java" file must have the same name as the file.

11) Extends

Classes may be inherited by other classes via *extends*. Inherited items of a class become part of the inheriting class so they do not have to be rewritten.

12) Implements

Abstract classes (classes that are shells without working code) can be inherited into a program via *implements*. The class that receives this shell must implement this shell. This is a way of imposing certain logic cycles on a program.

13) Class Body

consists of a compound statement ({ }) with variable identifier declarations, methods and other classes inside.

```
{
    declaration
    declaration
    :
    declaration

    method
    method
    :
    method

    internal-class
    internal-class
    :
    internal-class
}
```

14) Compound Statement

- consists of code surrounded by curly braces ({ })
- all code in a compound statement is within the scope of the compound statement

15) Identifiers

Identifiers are names for objects in Java. They must begin with an underscore ("_") or a letter and must consist of underscores, letters or digits.

16) Declarations

See Section 2: Basic Java Statements

17) Fundamental Data Types

See Section 2: Basic Java Statements

18) Methods

Methods are the basic working functions of a Java class. They consist of

```
method-header  
compound-statement
```

19) Method Header

```
access return-type method-name( parameter-list )
```

20) Parameter Lists

a) Syntax:

```
data-type-or-class identifier[,data-type-or-class identifier  
[, data-type-or-class identifier[,...]]]
```