Introduction to the Java programming language

CS 4354
Fall 2012

Jill Seaman

A simple java program

Welcome.java

```
//This application program prints Welcome
//to Java!

public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```


Compilation

- To compile the program enter at the prompt (Unix or Dos):

```
javac Welcome.java
```

- ✦javac is the java compiler
- ✦If successful, this command creates the file Welcome.class in the same directory
- ✦Welcome.class contains platform-independent bytecode
- ✦bytecode is interpreted (executed) by a Java Virtual Machine (JVM), and will run on a JVM installed on any platform
- ✦The program does NOT need to be recompiled to run on another platform.

Execution

- To run the program enter at the prompt (Unix or Dos):

```
workspace jill$ java Welcome
Welcome to Java!
workspace jill$
```

- ✦This runs the java bytecode on a Java Virtual Machine.
- ✦The java tool launches a Java application. It does this by starting a Java runtime environment, loading a specified class, and invoking that class's main method.
- ✦The method must be declared public and static, it must not return any value, and it must accept a String array as a parameter.

Editions of Java

- Different editions of java target different application environments
  - ✦Java Card for smartcards.
  - ✦Java Platform, Micro Edition (Java ME) — targeting environments with limited resources.
  - ✦Java Platform, Standard Edition (Java SE) — targeting workstation environments.
  - ✦Java Platform, Enterprise Edition (Java EE) — targeting large distributed enterprise or Internet environments.
- Each edition offers slightly different libraries (APIs) suited for the given environment.
- API: Application Programming Interface: the specification of the interface.

Packages of Java

- Two distributions:
  - ✦Java Runtime Environment (JRE) contains part of the Java SE platform required to run Java programs (JVM)
  - ✦Java Development Kit (JDK) is for developers and includes development tools such as the Java compiler, Javadoc, Jar, and a debugger.

Releases of Java

- Different releases of Java
  - ✦JDK 1.0 (1996) Codename: Oak
  - ✦JDK 1.1 (1997)
  - ✦J2SE 1.2 (1998)
  - ✦J2SE 1.3 (2000)
  - ✦J2SE 1.4 (2002)
  - ✦J2SE 5.0 (2004) (1.5)
  - ✦Java SE 6 (2006) (1.6) (I have this one)
  - ✦Java SE 7 (2011)

Principles

- There were five primary goals in the creation of the Java language:
  - ✦It should be "simple, object-oriented and familiar"
  - ✦It should be "robust and secure"
  - ✦It should be "architecture-neutral and portable"
  - ✦It should execute with "high performance"
  - ✦It should be "interpreted, threaded, and dynamic"

Features

- Interesting features of Java
  - ✦Object-oriented: everything is an object
  - ✦Inheritance
  - ✦Polymorphism: can use a subclass object in place of the superclass
  - ✦Garbage collection (dynamic memory allocation)
  - ✦Exception handling: built-in error handling
  - ✦Concurrency: built-in multi-threading
  - ✦Persistence: support for saving objects' state between executions
  - ✦Platform independence: supports web programming

Free Java textbook available online

- "Thinking in Java" by Bruce Eckel, 4th edition, 2006, ISBN 0131872486, Pearson Education
- The third edition is a free electronic book:

  http://www.mindview.net/Books/TIJ/

Characteristics of Pure object-oriented programming

- Everything is an object.
  - ✦attributes + operations
- A program is a bunch of objects telling each other what to do by sending messages
  - ✦a message as a request to call a method that belongs to a particular object
- Each object has its own memory made up of other objects.
  - ✦this is how to represent complex systems
- Every object has a type.
  - ✦its type is a class, the class specifies the methods of the object
- All objects of a particular type can receive the same messages.
  - ✦Even the instances of the subclasses

All objects in Java are really references

- Everything is treated as an object, using a single consistent syntax.
- However, the identifier you manipulate is actually a "reference" to an object

```
String s;  //this is just a ref, a pointer
```

- Safer to initialize a reference when you create it:

```
String s = "asdf";
```

- Usually you use "new" to create new objects:

```
String s = new String("asdf");
```

- Note: references are on stack, objects are in heap.

Special case: primitive types

- These are NOT references, not objects
- They are stored on the stack
- Size is not machine-dependent, always the same

Primitive type	Size	Minimum	Maximum	Wrapper type
boolean	—	—	—	**Boolean**
char	16-bit	Unicode 0	Unicode 2^{16}-1	**Character**
byte	8-bit	-128	+127	**Byte**
short	16-bit	-2^{15}	$+2^{15}-1$	**Short**
int	32-bit	-2^{31}	$+2^{31}-1$	**Integer**
long	64-bit	-2^{63}	$+2^{63}-1$	**Long**
float	32-bit	IEEE754	IEEE754	**Float**
double	64-bit	IEEE754	IEEE754	**Double**
void	—	—	—	**Void**

Wrapper:
object that
contains the primitive

Arrays in Java

- An array is ALWAYS initialized
  - ✦cannot access uninitialized elements by mistake
- Arrays have bounds checking
  - ✦unable to access memory outside its block (using the array): runtime error
- This is to enforce safety (though it requires overhead)
- Arrays are objects, contain primitives or references to objects
  - ✦member length returns size of array
  - ✦can access elements using [x]

```
Weeble[] c = new Weeble[4];
for(int i = 0; i < c.length; i++)
  if(c[i] == null) // Can test for null reference
    c[i] = new Weeble();
```


Classes in Java, fields

- A Class defines a type with fields (data) and methods (operations)
- Fields can be objects or primitives

```
class ClassA {
  int i;
  Weeble w;
}
```

- Can create an object of this class:

```
ClassA a = new ClassA();
```

- Fields are accessible using dot operator

```
a.i = 11;
a.w = new Weeble();
```


Default values for fields of primitive type

- All fields of primitive types are initialized to the following default values.

Primitive type	Default
boolean	**false**
char	**'\u0000' (null)**
byte	**(byte)0**
short	**(short)0**
int	**0**
long	**0L**
float	**0.0f**
double	**0.0d**

- These apply to fields, not to local variables.

Classes in Java, methods

- Methods in Java determine the messages an object can receive.
- They are functions that the object can execute on itself
- Syntax is very similar to C++

```
class ClassA {
  int i;
  Weeble w;
  int mult (int j) {
     return i*j;
  }
}
```

- Methods are accessible using dot operator

```
ClassA a = new ClassA();
a.i = 10;
int x = a.mult(4);
```


Accessing classes from libraries

- In Java libraries are called packages
- Packages have dotted path names (like internet domains)
- To use a class from a package, import the qualified class name:

```
import java.util.ArrayList;
```

- Or import the entire package:

```
import java.util.*;
```


static keyword

- When a field or method is declared static, it means that data or method is not tied to any particular object instance of that class
- Instances of the class share the same static fields
- Static methods may not access non-static fields

```
class StaticFun {
  static i = 11;
  static void incr () { i++; }
}
```

- Static fields and methods may be accessed without instantiating any objects, or from an existing object.

```
StaticFun.i = 100;
StaticFun sf = new StaticFun();
sf.incr();
```


A Java program

```
// HelloDate.java
import java.util.*;

public class HelloDate {
  public static void main(String[] args) {
    System.out.println("Hello, it's: ");
    System.out.println(new Date());
  }
}
```

- Standalone program: one class must have same name as file. that class must have a main method with signature as above.
- args are for command line arguments.
- public means method is available outside the file
- comments: `/* ... */` or `//...to end of line`

Java library documentation

- Online documentation for Java 1.6 API

  http://docs.oracle.com/javase/6/docs/api/

- java.lang is always implicitly loaded
  - ✦`System` class, contains `out` field (a static `PrintStream`)
  - ✦`PrintStream` has `println` methods
- Look for Date
  - ✦java.util.Date
  - ✦shows constructor and other methods in documentation

Operators in Java

- Mathematical operators, same as C++

```
+ - * / %
++  --
+= -= *= /= %=
```

  - ✦integer division truncates, like C++
- Relational operators yield boolean result (not int)

```
< > <= >= == !=
```

  - ✦== over objects tests the value of the reference (the pointers)
- Logical operators `&&  ||  !`
- String + is concatenation: `"abc" + "def"`
  this yields a new string object: `"abcdef"`

Assignment in Java

- Assignment in Java is like in C++
  - ✦For primitive types, values are copied

```
int a;
a = 10;
```

  - ✦For objects, the reference is copied so both variables refer to the same object.

```
Weeble b = new Weeble();
Weeble a;
a = b;    // a and b refer to same Weeble object
```

  - ✦changes to a will also affect b
- Objects are passed by reference by default

Control flow in Java (same as C++)

- if-else

```
if(Boolean-expression)
  statement
else
  statement
```

```
if(Boolean-expression)
  statement
```

- while, do-while, and for

```
while(Boolean-expression)
  statement
```

```
do
  statement
while(Boolean-expression);
```

```
for(initialization; Boolean-expression; step)
  statement
```

- break and continue (also with labels)
- switch statement like C++