

Core Java Course Basics of Java programming (Chapters 2 & 3)

Dr. Jeff Blessing
MSOE

Agenda

- ❖ Issues/Questions from the previous lecture
 - ❖ Java 1.1 Web browser compatibility
 - ❖ Development Environments
- ❖ Data Types
- ❖ Operators
- ❖ Programming constructs
- ❖ Code examples

2

Java System Requirements

- ❖ Windows 95/98/NT (not DOS or Win 3.1)
 - ❖ at least 32 MB memory
 - ❖ at least 50 MB to 200 MB of disk space
 - ❖ uses long filenames (up to 255 characters)
 - ❖ don't use spaces in the dir./file names
 - ❖ names are Case Sensitive in Java, including file names
- ❖ Also runs under Solaris, Linux, Macintosh

3

Finding the Proper Java Release

- ❖ <http://java.sun.com/products>
 - ❖ Java 2 contains:
 - ❖ The Java 1.3 SDK (Sys. Dev. Kit)
 - ❖ Also known as the JDK 1.3
 - ❖ Three different "bundles" are available:
 - ❖ J2SE (the Standard Edition)
 - ❖ J2EE (the Enterprise Edition)
 - ❖ J2ME (the Micro Edition)
 - ❖ Tons of other stuff...!

4

Downloading/Installing the JDK

- ❖ <http://java.sun.com/products/jdk/1.3>
 - ❖ download and extract
 - ❖ \jdk1.3\bin
 - ❖ add this directory to the path variable
 - ❖ \jdk1.3\src
 - ❖ add this directory to the classpath variable
 - ❖ \jdk1.3\lib
 - ❖ add this directory to the classpath variable

5

JDK Features

- ❖ Command line interface only!
 - ❖ javac is the compiler
 - ❖ java is the interpreter (JVM)
 - ❖ appletviewer is for testing applets
 - ❖ outside the environment of a web browser
 - ❖ jdb is a source code debugger (tedious)
 - ❖ javadoc is a documentation generator
 - ❖ other things are included

6

Shareware Products for Java

- ❖ TextPad (www.textpad.com)
 - ❖ comes on the book's CD
 - ❖ edits, compiles, and reviews errors
- ❖ CodeWright (www.premia.com)
 - ❖ edits, compiles, and reviews errors
- ❖ Kawa (www.tek-tools.com)
- ❖ Jpad Pro (www.modelworks.com)
 - ❖ both also have an integrated debugger

7

Full-fledged Java IDE's



- ❖ Java WorkShop
 - ❖ comes from Sun
 - ❖ loaded on workstations in S-307
- ❖ Symantec VisualCafé
 - ❖ leading seller since 1996
- ❖ Borland JBuilder
- ❖ Others, but not J++

8

Java Basics

- ❖ Java is based on C++
- ❖ C++ adds obj-oriented capabilities to C.
- ❖ Programs can be stand-alone
 - ❖ applications (usually run from local files)
- ❖ ... or run over a network
 - ❖ applets (usually run from a web page)
 - ❖ usually have viewable graphics
 - ❖ run from Browser's JVM implementation

9

Java Basics (cont.)

- ❖ Has same control statements as C/C++
 - ❖ { }, if, if-else, if-else-if, for, while, do-while, switch, break, continue, (no goto!)
- ❖ Data structures differ from C/C++
 - ❖ no structs, (array syntax differs slightly)
 - ❖ no pointers, only references (null ref. okay)
- ❖ Only member functions (methods)
 - ❖ no "global" functions (or "free" functions)
 - ❖ no "global" scope (everything is in packages)

10

Control Flow Statements

- ❖ Conditional statements


```
if (condition) statement;
if (condition) { block }
if (condition) statement1 else statement2;
if (condition) {block1} else {block2};
```
- ❖ Examples


```
if(day.equals("Saturday") isWeekend = true;
if(number/2*2 == number)
{ type = "even"; }
else
{ type = "odd"; }
```

11

Control Flow Statements

- ❖ Switch statement


```
switch (choice)
{
case 1:
. . .
break;
case 2:
. . .
break;
default:
// bad data
break;
}
```

12

Control Flow Statements (cont.)

- ❖ Loops


```
while (condition)
{
    block
}
for (statement; condition; statement)
{
    block
}
```

13

Control Flow Statements (cont.)

- ❖ Iteration


```
do
{
    block
}
while (condition);
```
- ❖ Break statement
 - ❖ break OR break label;
 - ❖ cancels the loop even if condition is true
 - ❖ can also mean jump to a labeled line of code

14

Java Basics (cont.)

- ❖ Strongly typed language
 - ❖ run-time type checking
- ❖ Two kinds of data types:
 - ❖ primitive
 - ❖ int, short, long, float, double, char, boolean, byte
 - ❖ reference
 - ❖ objects!
 - ❖ new operator required to create an object

15

Comparisons

	Java	C++	C
<i>Obj-oriented</i>	√	√	
<i>Platform-Ind.</i>	√	ANSI	ANSI
<i>Win/Graphics</i>	√	MFC/X	MFC/X
<i>Multi-threaded</i>	√	pthreads	pthreads
<i>Net/Internet</i>	√	Lib?	Lib?
<i>Performance</i>	3	2	1

16

Rules of the Game

- ❖ Everything is an object
 - ❖ except for primitive data types
- ❖ Everything is defined inside a class
 - ❖ all methods, including main()
- ❖ Every variable must have a declared type
- ❖ Code and file names are case-sensitive
- ❖ File name is the same as class + .java
- ❖ After compiling .java turns into .class

17

Code structure

```
/* multi-line comments */
/** multi-line comments to generate documentation */

public class Hello
{ // braces delimit code blocks
    public static void main( String[] args)
    {
        System.out.println("Hello World");
    }
}
```

18

Building & Executing Java Code

- ❖ Source file name must end in `.java`
 - ❖ and must match the public class's name
- ❖ Compiling to produce `.class` file
 - ❖ `javac Hello.java`
- ❖ Running in the JVM environment
 - ❖ `java Hello`
- ❖ Notice the lack of a `.class` extension

19

Primitive Data Types

- ❖ Numeric types
- ❖ Character type
- ❖ Boolean type

20

Numeric Data Types

- ❖ Integer number types
 - ❖ `int` 4 bytes (9-10 digits)
 - ❖ `short` 2 bytes (-32,768 - 32,767)
 - ❖ `long` 8 bytes (18-19 digits), L
 - ❖ `byte` 1 byte (-128 - 127)
- ❖ Floating point number types
 - ❖ `float` 4 bytes (10^{38} , 6-7 significant digits), F
 - ❖ `double` 8 bytes (10^{308} , 15 significant decimal digits)

21

Numeric Data Types (cont.)

- ❖ Unlike C/C++ `int`, `short`, `long`, `float`, and `double` are completely machine and OS independent
- ❖ In many cases using `long` and `double` is the most reasonable.
- ❖ Using `short` or `float` data types justifiable for large arrays or very memory/performance demanding applications

22

Numeric Data Types Conversion

- ❖ Data size in mixed expressions is upgraded to the largest size data size present
 - ❖ upgrade order byte, short, long, float, double
- ❖ Assignment of different types
 - ❖ conversion to larger size - no cast needed
 - ❖ conversion to smaller size - requires specific cast (since loss of precision is possible) or rounding: `Math.round(x)`
 - ❖ precision order double, float, long, short, byte

23

Character Data Type

- ❖ Character type
 - ❖ `char` 2 bytes (65,536 chars, Unicode)
- ❖ Unlike C/C++ characters are not numbers
 - ❖ conversion from a character to an integer requires specific cast
- ❖ Backslash is used to denote escape sequence:
 - ❖ `'\b'` or `'\u0008'`, `'\t'` or `'\u0009'`, `'\n'` or `'\u000a'`

24

Boolean Data Type

- ❖ Boolean type - used for logical testing
 - ❖ `boolean` 1 bit (true, false)
- ❖ Boolean type is not an integer number
- ❖ Unlike C/C++ an integer cannot even be converted into boolean type

25

Strings

- ❖ Not a primitive (built in data type)
- ❖ Predefined class `String`
 - ❖ `String name;`
 - ❖ `String dayOfWeek = "Monday";`
- ❖ Strings can be concatenated
 - ❖ `String student = "Smith";`
 - ❖ `String class = "Physics";`
 - ❖ `String assignment = student+ " takes "+class;`

26

Strings (cont.)

- ❖ Strings are immutable
 - ❖ you cannot replace character(s) in a string
 - ❖ you can only re-assign string variable and it is going to be a new string
- ❖ Strings are not C++ arrays of characters
 - ✗ `char language[] = "Java";` // not similar
 - `char* language = "Java";` // similar (C++)
 - ❖ memory is automatically allocated and freed up

27

(Some) String Methods

```
char charAt(int index);           // return specified char
int compareTo(String aString);   // similar to strcmp()
boolean equals(Object anObject); // checks equality
boolean equalsIgnoreCase(String anotherString);
int length();                     // returns the length
String substring(int beginIndex); // returns the substring
String substring(int beginIndex, int endIndex);
String toLowerCase();             // converts to lower case
String toUpperCase();            // converts to upper case
String trim();                    // deletes leading/trailing whitespace
```

28

Operators (cont.)

- ❖ Arithmetic
 - `++` - pre- or post increment (unary)
 - `--` - pre- or post decrement (unary)
 - `+`, `-` - unary plus and minus
 - `*`, `/`, `%` - multiplication, division, remainder
 - `+`, `-` - addition (concatenate), subtraction
- ❖ Relational
 - `<`, `<=` - less than, less then or equal
 - `>`, `>=` - greater than, greater then or equal

29

Operators (cont.)

- ❖ Cast
 - `(type)` - type conversion
- ❖ Logical
 - `==` - equal, refer to the same object
 - `!=` - not equal
 - `&&` - AND
 - `||` - OR
 - `instanceof` - refer to the same type

30

Operators (cont.)

- ❖ Bitwise
 - & - AND
 - | - OR
 - ^ - XOR
 - ~ - NOR
 - << - left shift
 - >>, >>> - right shift with sign, right shift with zero
- ❖ Conditional
 - ?: - conditional execution (ternary)

31

Operators (cont.)

- ❖ Assignment
 - = - assign
- ❖ Assignment with operation
 - *= - multiply and assign
 - /= - divide and assign
 - %= - divide and assign remainder
 - += - add and assign
 - = - subtract and assign
 - <<= - shift and assign
 - &=, ^=, |= - perform operation and assign

32

Variables

- ❖ Variables must be declared before using
 - ❖ int year;
 - ❖ char day, month;
- ❖ Variable name starts from a letter, '_', '\$'
- ❖ Variable name contains any letters or digits
- ❖ Size of a variable name is not limited and all characters are significant and case sensitive

33

Console I/O Application

Filename: Count.java

```

Public class Count
{
    public static void main(String args[])
    {
        int count = 0;

        while (System.in.read() != -1)
            count++;
        System.out.println(count + " chars read");
    }
}

```

34

Java Applications

- ❖ Have a method named main()
 - ❖ must be public & static
 - ❖ passed an args array
 - ❖ args.length is its size
- ❖ Compiled then run by the Java interpreter
 - ❖ javac Count.java (compiles to Count.class)
 - ❖ java Count (runs Count.class in JVM)

35

Web-based Applets

Filename: Hello.java

```

import java.applet.Applet;

public class Hello extends Applet
{
    public void paint(Graphics g)
    {
        g.drawString("Hello Web", 20, 30);
    }
}

```

36

Applets (cont.)

- ❖ Applets must be compiled:
 - ❖ `javac Hello.java`
- ❖ ... and run from a web page:

```
<html>
<head><title>Hello Web Demo</title></head>
<body>
<applet code="Hello.class"
        width=120
        height=80>
</applet>
</body>
</html>
```

`c:> appletviewer Hello.html`

37

CLASSPATH env. variable

In general, the CLASSPATH must contain the full pathnames to dirs that have `.class` files needed at compile OR run time. This includes the code file named in the applet tag! When using packages, the PARENT dirs to the package files (named on the `import` stmts) must be on the CLASSPATH as well! When defining (some say "declaring") packages, the pkg names are mapped to directory names on the server, with the "dots" acting as a subdir separator. The packages themselves can be located off of ANY parent dir on the CLASSPATH.

38

Applets

- ❖ Subclass class `Applet` and over-ride methods
 - ❖ `init()` opens files, sockets, threads
 - ❖ `start()` applet is visible on web page
 - ❖ `stop()` applet is not visible on web page
 - ❖ `destroy()` close files, sockets, threads
 - ❖ `paint()` draws the applet's viewable image
 - ❖ `run()` if a separate thread runs applet

39

Reading Input

- ❖ Use `CoreJava Console` class
- ❖ Methods:
 - ❖ `String readString(String prompt);`
 - ❖ `String readWord();`
 - ❖ `int readInt(String prompt);`
 - ❖ `double readDouble(String prompt);`

40

Formatting Output

- ❖ `NumberFormat` class in `java.text` package
 - ❖ `numbers` (28,456.3)
 - ❖ `currency` (\$57.22)
 - ❖ `percentage` (88.2%)
- ❖ Methods (localized):
 - ❖ `NumberFormat.getNumberInstance();`
 - ❖ `NumberFormat.getCurrencyInstance();`
 - ❖ `NumberFormat.getPercentInstance();`

41

Formatting Output (cont.)

- ❖ Examples


```
NumberFormat nf =
    NumberFormat.getNumberInstance();
string number = nf.format(amount);

NumberFormat nfg = NumberFormat.
    getCurrencyInstance(Locale.GERMAN);
string dm = nfg.format(amount);
```

42

Create Your Own Format

- ❖ DecimalFormat class
- ❖ Example


```
DecimalFormat df =
  new DecimalFormat("0.###");
```
- ❖ Symbols:
 - ❖ 0 - a digit
 - ❖ # - a digit (not show if trailing/leading)
 - ❖ . - decimal separator
 - ❖ , - group separator
 - ❖ - - negative sign
 - ❖ % - divide by 100 and show percentage
 - ❖ any other - symbol gets included into output

43

Formatting: If you need more...

- ❖ For people who are used to C/C++ formatting flexibility...
- ❖ For those who need to utilize octal and hexadecimal format...
- ❖ ... take advantage of CoreJava Format class

44

Code Examples

- ❖ Square Root Calculator
 - ❖ $y_{n+1} = (y_n + x/y_n)/2$
 - ❖ Add greeting message
 - ❖ Enable multiple calculations without restarting
 - ❖ Enable exit
 - ❖ Add iteration info

45

Code Examples (cont.)

- ❖ Mortgage Calculator
 - ❖ $mPayment = principal * mInterest / (1 - (1 / (1 + mInterest)^{nYears * 12}))$
 - ❖ Total cost of all the payments
 - ❖ Accelerated payment option
 - ❖ How much can you save?
 - ❖ getData method / multiple mortgages

46

Arrays

- ❖ Arrays are objects in Java
 - ❖ Not a machine dependent type as in C/C++
- ❖ Every array has a length attribute
 - ❖ Its statically allocated size
- ❖ Access is via [] (subscript operator)
- ❖ Arrays of primitive types hold values
- ❖ Arrays of objects hold references (pointers)
- ❖ Arrays of arrays hold pointers to arrays

47

Array Example (Shell Sort)

```
public static void main(String[] args)
{ // make an array of ten integers
  int[] a = new int[10];
  int i;
  // fill the array with random values
  for (i = 0; i < a.length; i++)
    a[i] = (int)(Math.random() * 100);
  print(a);
  sort(a);
  print(a);
}
```


48



Shell Sort (cont.)

```
public static void print(int[] a)
{ for (int i = 0; i < a.length; i++)
  System.out.print(a[i] + " ");
  System.out.println();
}
```

49



```
public static void sort(int[] a)
{ int n = a.length;
  int incr = n / 2;
  while (incr >= 1)
  { for (int i = incr; i < n; i++)
    { int temp = a[i];
      int j = i;
      while (j >= incr && temp < a[j - incr])
      { a[j] = a[j - incr];
        j -= incr;
      }
      a[j] = temp;
    }
    incr /= 2;
  }
}
```

50



51