

Beginners Introduction to Java 2

James W. Cooper
<http://flahdo.watson.ibm.com>



Your handouts

- Java for Beginners CDROM
- Visual SlickEdit
- Draft Textbook



Your resources

- <http://flahdo.watson.ibm.com>
 - Click on Java for Beginners
 - Complete book and programs
 - Each day's lecture
 - Assignments/ answers
 - Other things we may think of.
- E-mail me
 - Jwcnmr@watson.ibm.com



What constitutes a computer?

- Memory
 - For data and programs
- Processor
 - To interpret program instructions
- Disk
 - Auxiliary memory for data and programs
- Keyboard, mouse
- Display
- Printer
- Network connection



Memory

- Is made up of 8-bit bytes
- PCs contain 32Mb to 512Mb or more
- Words consist of bytes
 - 32-bit words=4 bytes each



How can memory be both?

- Data
 - 1 byte for each character of text
 - 4 bytes for an integer
 - 8 bytes for a floating point number
 - 6.02×10^{23}
- Instructions
 - Usually each one is 32 bits
- Difference is only what you say is a program.
 - Point the computer to the numbers that make up a program and it will begin trying to execute them.



Consider a simple 4-bit computer

- Here are the instructions
 - 0000 stop
 - 0001 fetch
 - 0010 add
 - 0100 subtract
 - 1000 store
- Here is a program
 - FETCH 0001
 - ADD 0010
 - STORE 1000
 - STOP 0000



Maybe we need data memory, too

Fetch 1000	0001 1000	1000 1010
Add 1001	0010 1001	1001 0001
Store 1010	1000 1010	1010 1011
Stop	0000	



Computers also make decisions

- Not just glorified calculators
 - Jmp if 0
 - Jmp if non-zero
 - Jmp if < 0
 - Jmp if > 0
 - Jmp if add caused a carry
- These *branching instructions* allow programs to take alternate pathways depending computational results.



Writing programs at this level quickly becomes tedious

- | | |
|-----------------|-----------|
| ■ Fetch A | ■ A: xxxx |
| ■ Add B | ■ B: xxxx |
| ■ JmpIfNeg Z | ■ C: xxxx |
| ■ Stop | |
| ■ Z: Subtract C | |
| ■ Stop | |



Instead, we create high level languages
to carry out these operations

- $k = a + b$
- If $k < 0$ then
 - $k = k - 1$
- end



Compilers

- Programs that create machine level instructions from language statements.
- Compilers exist for many languages
 - Basic
 - Visual Basic
 - C, C++
 - Pascal



Why do we write computer programs?

- Perform tedious, repetitive tasks.
- Analyze a pile of experimental data
 - statistics, clustering
- Search for correlations in data
 - databases, tables
- Build friendly interfaces on top of complicated data systems.
 - Local programs or web sites



The Java language is a little different

- Java compilers produce code for a *virtual machine*.
- Hypothetical computer which has its own instruction set.
- Then for each computer an *interpreter program* is written to execute the instructions of the virtual machine.



Advantages and Disadvantages

- All computers can execute the same compiled Java binary code.
- They each have their own interpreter.
- This is of necessity somewhat slower.
- It is fairly easy to write nice windowing user interfaces in Java



Getting around speed limits

- Just-in-time (JIT) compilers
 - Compile repeated code segments to local machine's native code when they are first encountered.
- Hot spot compiler
 - Finds repeatedly executed code segments and specifically optimizes and compiles them.



Your Java Beginner's CDROM

- Complete copy of text of book as PDF file.
 - Need Acrobat Reader, www.adobe.com
- Complete program examples in zip file
- Java Development kit, jdk1.3
- Java Documentation (web pages)
- WinZip
 - unregistered vsn with nag screen



How you can help improve the book

- Send E-mail describing substantive improvements that would help you, and get your name in the acknowledgments.
- Print out pages and mark typos.
- Suggest additional exercises or examples.



Installing your programs

1. Install Jdk1.3
2. Unzip Jdk documents into \jdk1.3\docs
3. Install Visual SlickEdit



The Java Language

- Same basic syntax as C and C++
 - Language is case sensitive
- $Y = m * x + b;$
 $y = m * x + b;$
- These are different Y's
 - All statements end with a semicolon



Capitalization conventions

- Often constants are written in upper case.

```
final PI = 3.1416;
```

- Variables usually start in lower case
`roomTemperature = 20;`

- Embed caps to separate words



You must declare all variables

```
int x, y, z;
```

```
x = 5;
```

```
y = 4;
```

```
z = x + y;
```



In Java you often declare variables as you use them

```
int x = 5;  
int y = 6;  
int z = x + y;
```



Basic Java variable types

int	32-bit integer
long	64-bit integer
float	32-bit floating point
double	64-bit floating point
boolean	true or false
char	1 character 16-bit Unicode
String	a string of Unicode characters
short	16-bit integer
byte	8-bit number



Converting between types

- Can always convert to longer type
`int k = 234;`
`long lk = k;`
`float ftemp = lk;`
- If longer to shorter, must use “cast”
`int j = (int)ftemp;`



Default Java assumptions

- A number with a decimal point is a double.
- A number without a decimal is an integer.
- Otherwise, you must specify the type.
`double trouble = 3.45;`
`float loan = 1.23f;`
`long pig = 5L;`



Character constants

- Characters enclosed in single quotes
`char c = 'k';`
- White space characters preceded with backslash.

```
char carriageReturn = '\r';
```

```
char lineFeed = '\n';
```

```
char backSlash = '\\';
```



Strings enclosed in quotes

- Strings can be of any length
`String theory = "Physics class";`

- Can include special characters

```
String title =
```

```
    "Are we \"happy\" yet?";
```



Writing our first Java program

```
public class add2 {  
    public static void main(String arg[]) {  
        double a, b, c;  
        a = 1.75;  
        b = 3.46;  
        c = a + b;  
        //print out sum  
        System.out.println("sum = " + c);  
    }  
}
```



Points to observe in program

- Program starts in main.
 - Exactly that form reqd
- Each file is one class
- Name of file must match class name
- Comments start with //
- Class and each function within it are surrounded by braces { }

```
public class add2 {  
    public static void main(String arg[]) {  
        double a, b, c;  
        a = 1.75;  
        b = 3.46;  
        c = a + b;  
        //print out sum  
        System.out.println("sum = " + c);  
    }  
}
```



Many possible brace styles

- Compact

```
public class add2{  
}
```

- Spread out

```
public class add2  
{  
}
```



Output to console

```
System.out.println(string);
```

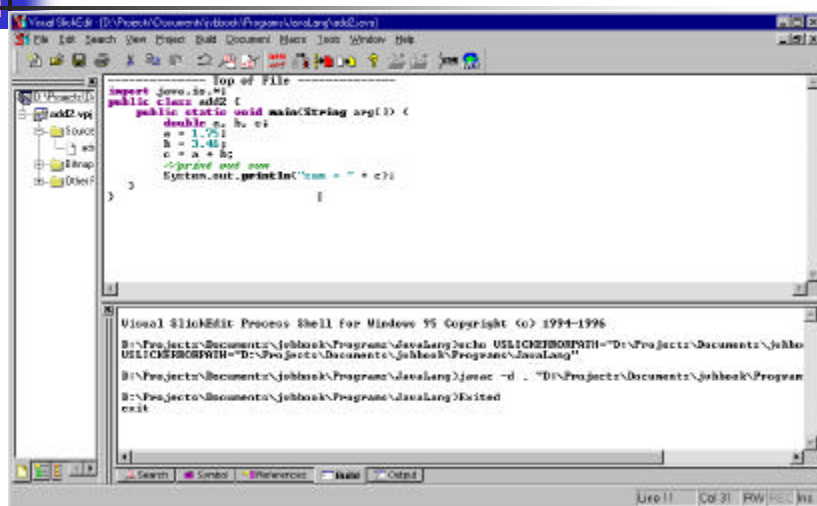
- Combine strings with +-sign

```
System.out.println("sum=" + sum);
```

Compile and run

- Compile with
`javac add2.java`
 - Forms add2.class
- Run with
`java add2`
 - “.class” implied

Compile Using VSlick



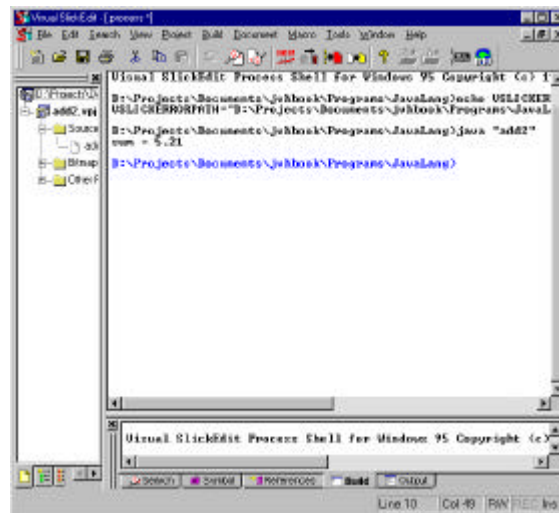
The screenshot shows the VSlick IDE interface. The main editor window displays the following Java code:

```
import java.io.*;
public class add2 {
    public static void main(String arg[]) {
        double a, b, c;
        a = 1.751;
        b = 3.452;
        c = a + b;
        //print and sum
        System.out.println("sum = " + c);
    }
}
```

The left sidebar shows a project tree with folders for 'add2', 'src', 'lib', and 'Other'. Below the editor, a terminal window titled 'Visual SlickEdit Process Shell for Windows 95 Copyright (c) 1994-1996' shows the following commands and output:

```
D:\Projects\Documents\johhask\Programs\java\lang>echo %USERPROFILE%
D:\Projects\Documents\johhask\Programs\java\lang>
D:\Projects\Documents\johhask\Programs\java\lang>javac -d . "D:\Projects\Documents\johhask\Programs\java\lang\add2.java"
D:\Projects\Documents\johhask\Programs\java\lang>exit
```

Execute from within VSlick



Homework assignment

1. Install Java and Visual SlickEdit
2. Write a program to print out your name.