

Programming using Java

Java Basics

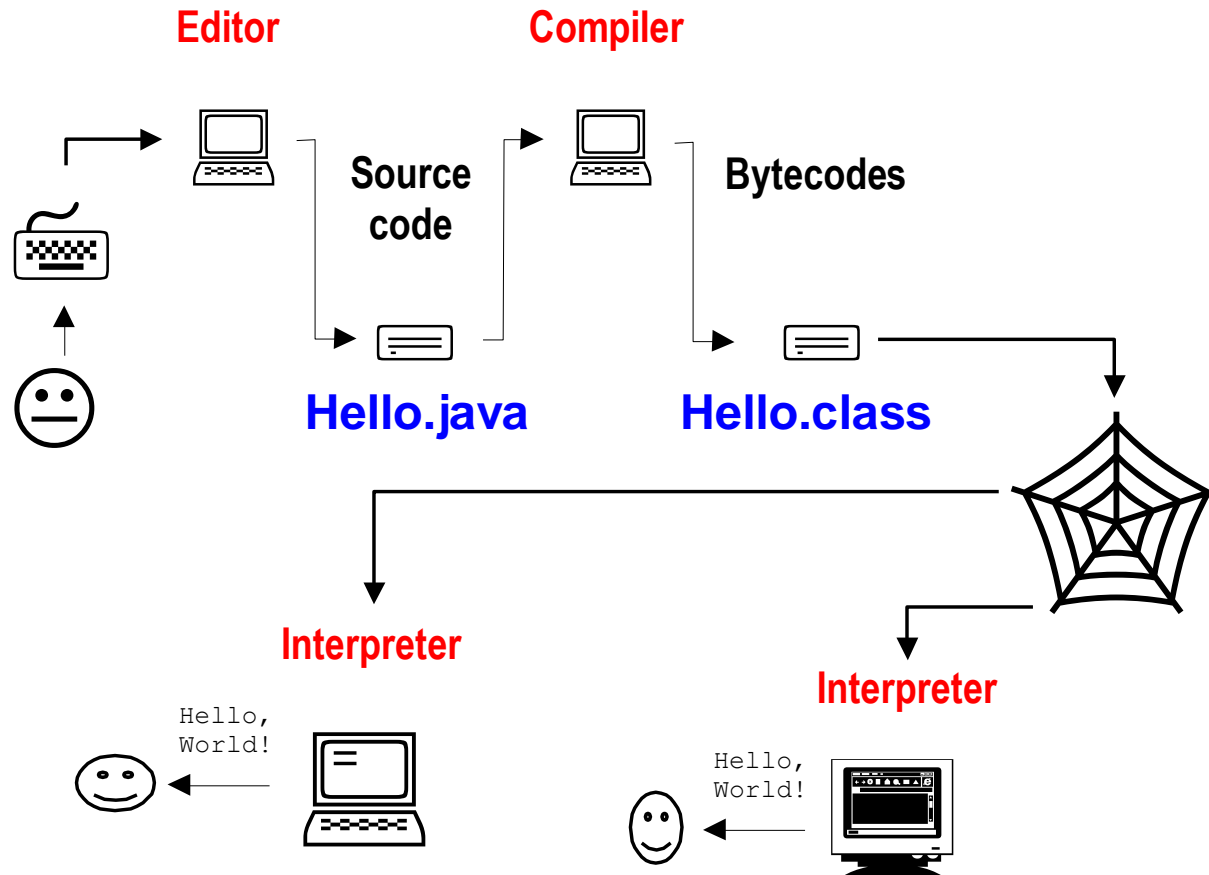
Java Technology

There is more to Java than the language.

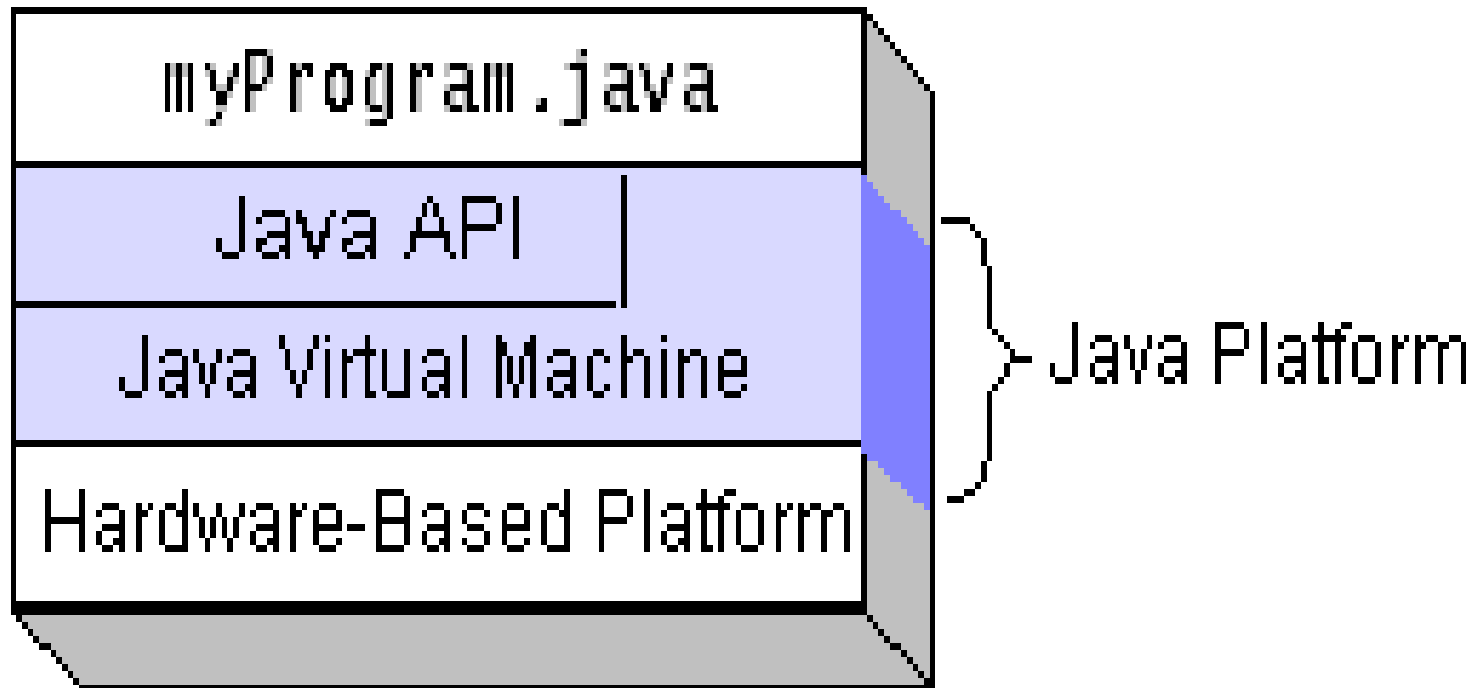
Java Technology consists of:

- 1) Java Programming Language
- 2) Java Virtual Machine (JVM)
- 3) Java Application Programming Interfaces (APIs)

Java's Compiler + Interpreter



Java Execution Platform



Start a Java Application

Java

C

“Hello.java”

Class Name

```
class Hello {
```

Main method

```
    public static void main(String args[]) {  
        System.out.println(“Hello World”);  
    }
```

“hello.c”

Main Function

```
void main() {  
    printf(“Hello  
World\n”);  
}
```

% javac Hello.java

Compile

% cc hello.c -o hello

% java Hello

Run

% hello

output

Hello World

Java Virtual Machine

- The .class files generated by the compiler are not executable binaries
 - so Java combines compilation and interpretation
- Instead, they contain “byte-codes” to be executed by the Java Virtual Machine
- This approach provides platform independence, and greater security

HelloWorld program

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

- Note that String is built in
- println is a member function for the System.out class

Comments are almost like C

```
/* This kind of comment can span  
multiple lines  
*/
```

```
//This kind is to the end of the line
```

```
/**  
 * This kind of comment is a special  
 * 'javadoc' style comment  
*/
```


Primitive data types are like C

- Main data types are int, double, boolean, char, byte, short, long, float
- **boolean** has values **true** and **false**
- Declarations look like C, for example,
 - `double x, y;`
 - `int count = 0;`

Variables and Assignments

■ Variables types

- char 16 bits Unicode character data
- **boolean Boolean Variable**
- byte 8 bits signed integer
- short 16 bits signed integer
- int 32 bits signed integer
- long 64 bits signed integer
- float 32 bits signed floating point number
- double 64 bits signed floating point number

Variables and Assignments

Variable Declaration

```
type varName;
```

```
float x, y, x;
```

Value assignments

```
varName = value;
```

```
int num = 100;
```

```
long m = 21234234L
```

```
double type : .5  0.8  9e-2  -9.3e-5
```

```
float type : 1.5e-3f;
```

Strings and Characters

- String : sequence of character

String s = “Enter an integer value: ” ;

Char c = ‘A’; ←————— Unicode

- Concatenation Operator ‘+’

String s = “Lincoln said: ” + “\” Four score ago\”” ;

Result :

Lincoln said: “Four score ago”

Expressions are like C

- Assignment statements mostly look like those in C
- you can use `=`, `+=`, `*=` etc.
- Arithmetic uses the familiar `+` `-` `*` `/` `%`
- Java also has `++` and `--`
- Java has boolean operators `&&` `||` `!`
- Java has comparisons `<` `<=` `==` `!=` `>=` `>`
- Java does *not* have pointers or pointer arithmetic

Arithmetic Operators

■ Operators

➤ + - * / %

➤ += -= *= /= %=

➤ ++ --

5 / 2 → 2 Why isn't it 2.5 ?

5 % 2 → 1


4 / 2 → 2

4 % 2 → 0

➤ count * num + 88 / val - 19 % count

➤ j += 6; ➡ j = j + 6;

Operator Precedence

Operator	Association	Precedence
<p> () [] . ! ~ ++ -- + - (Data Type) * / % + - << >> >>> < <= > >= instance == != & ^ && ?: = += -= *= /= %= &= ^= = <<= >>= >>>= </p>	<p> Left Assoc. Right Assoc. Left Assoc. Left Assoc. Left Assoc. Left Assoc. Left Assoc. Left Assoc. Left Assoc. Left Assoc. Left Assoc. Right Assoc. Right Assoc. </p>	<p> (High)  (Low) </p>

Type Conversions in Assignment

- Widening Conversion

```
byte b = 127;
```

```
int i;
```

```
i = b;
```

- Wrong Conversion

```
byte b;
```

```
int i = 127;
```

```
b = i;
```

- Right Conversion (But data may be lost)

```
byte b;
```

```
int i = 127;
```

```
b = (byte) i;
```

Type Casting

Increment & Decrement Operator

■ Operator

➤ ++, --

➤ Prefix operator

```
n = 1;  
x = ++n;    // x=2, n=2
```

➤ Postfix operator

```
n = 1;  
x = n++;    // x=1, n=2
```

➤ Cannot be use with expressions, only with variables

➤ Cannot be applied at the real type

```
(a + b)++  // error
```

Back-Slash Codes

`\b` back space

`\n` new line

`\r` carriage return

`\f` form feed

`\t` tab

`\"` double quotation

`'` single quotation

`\0` null

`\\` back slash

`\uxxxx` Unicode (x: hexa decimal)

```
class SpecialCharacters {  
    public static void  
    main(String args[]) {  
  
        System.out.print("\u00a0  
        \u00a1 \u00a2 \u00a3");  
    }  
}
```

Bitwise Operators

■ Operator

- $\&, |, \ll, \gg, \ggg, ^, \sim$
- Operand should be integer type
- Precedence

Operator	Precedence
\sim $\ll \gg \ggg$ $\&$ \wedge $ $	(H) \updownarrow (L)

Ternary Operator

■ Operator

➤ Expr1 ? Expr2 : Expr3 (3 Terms Operator)

```
max = x > y ? x : y ;
```

```
if (x > y) max = x;  
else max = y;
```

```
m = a > b ? (c > a ? c : a) : (c > b ? c : b) ;
```

Assignment Operators

Expr 1 = Expr 1 op Expr2



Expr1 op= Expr 2

■ Operator

➤ Arithmetic operator : + - * / %

➤ Bitwise operator : & | ^ << >> >>>

sum = sum + i ;



sum += i ;

x = x * y + 1;



x *= y + 1;

x = x * (y+1)

Cast Operators

■ Data Type Casting Operator

(Data Type) Expression

➤ Cast operator : ()

(int) 3.75	====>	3
(float) 3	====>	3.0
(float) (1 / 2)	====>	0.0
(float) 1/2	====>	0.5

Control statements are like C

- `if (x < y) smaller = x;`
- `if (x < y){`
 `smaller = x; sum += x;}`
 `else { smaller = y; sum += y; }`
- `while (x < y) {`
 `y = y - x; }`
- `do { y = y - x; }`
 `while (x < y)`
- `for (int i = 0; i < max; i++)sum += i;`
- **BUT: conditions must be boolean !**

Control statements II

```
switch (n + 1) {  
    case 0: m = n - 1; break;  
    case 1: m = n + 1;  
    case 3: m = m * n; break;  
    default: m = -n; break;  
}
```

- Java also introduces the **try** statement, about which more later

Arrays

Definition of One Dimensional Array

type VarName[]

int ia[];

Assign Range to One Dimensional Array

varName = new type[size]

ia = new int[10];

Declaration of One Dimensional Array with Range

type varName[] = new type[size]

int ia[] = new int[10];

Java Keywords

■ 50 Java Keywords

abstract	double	int	super
boolean	else	interface	switch
break	extends	long	synchronized
byte	final	native	this
case	finally	new	throw
catch	float	package	throws
char	for	private	transient*
class	goto*	protected	try
const*	if	public	void
continue	implements	return	volatile
default	import	short	while
do	instanceof	static	strictfp
assert (New in 1.5)		enum (New in 1.5)	

Simple Java Program

A class to display a simple message:

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

Running the Program

Type the program, save as **Hello.java**.

In the command line, type:

```
> ls
```

```
Hello.java
```

```
> javac Hello.java
```

```
> ls
```

```
Hello.java, Hello.class
```

```
> java Hello
```

```
Hello World!
```

Explaining the Process

- 1) creating a source file - a source file contains text written in the Java programming language, created using any text editor on any system.
- 2) compiling the source file Java compiler (javac) reads the source file and translates its text into instructions that the Java interpreter can understand. These instructions are called bytecode.
- 3) running the compiled program Java interpreter (java) installed takes as input the bytecode file and carries out its instructions by translating them on the fly into instructions that your computer can understand.

Java Program Explained

Hello is the name of the class:

```
class Hello{
```

main is the method with one parameter args and no results:

```
public static void main(String[] args) {
```

println is a method in the standard System class:

```
System.out.println("First Java program.");
```

```
}
```

```
}
```

Main Method

The main method must be present in every Java application:

1) *public static void main(String[] args)* where:

a) public means that the method can be called by any object

b) static means that the method is shared by all instances

c) void means that the method does not return any value

2) When the interpreter executes an application, it starts by calling its main method which in turn invokes other methods in this or other classes.

3) The main method accepts a single argument – a string array, which holds all command-line parameters.

Getting Started:

(1) Create the source file:

➤ open a text editor, type in the code which defines a class (*HelloWorldApp*) and then save it in a file (*HelloWorldApp.java*)
file and class name are case sensitive and must be matched exactly (except the .java part)

Example Code: HelloWorldApp.java

```
/**
 * The HelloWorldApp class implements an application
 * that displays "Hello World!" to the standard output
 */
public class HelloWorldApp {
    public static void main(String[] args) {
        // Display "Hello World!"
        System.out.println("Hello World!");
    }
}
```



Java is CASE SENSITIVE!

Getting Started:

(2) Compile the program:

➤ compile HelloWorldApp.java by using the following command:

```
javac HelloWorldApp.java
```

it generates a file named HelloWorldApp.class

Getting Started:

(3) Run the program:

- run the code through:

```
java HelloWorldApp
```

- Note that the command is `java`, not `javac`,

- and you refer to

`HelloWorldApp`, not `HelloWorldApp.java` or
`HelloWorldApp.class`

Modify:

(4) Modify the program:

- Declare 2 variables a and b of type int.

- Multiply them

- Print the output as:

`a * b = Value obtained`

- You can copy paste any c program you have made and it will work (by taking care of issues discussed in the previous slides)

Java isn't C!

- In C, almost everything is in functions
- In Java, almost everything is in classes
- There is often only one class per file
- There *must* be only one **public** class per file
- The file name *must* be the same as the name of that public class, but with a **.java** extension

THE JAVA BUZZWORDS

■ Object oriented	5
■ Network Savvy	7
■ Robust	3
■ Secure	0
■ Architecture Neutral	2
■ Portable	9
■ Interpreted	8
■ High Performance	1
■ Multithreaded	4
■ Dynamic	6