

## The Java Language

- ### Java
- Interpreted language
  - Strongly typed
  - Similar syntax to C / C++
  - No pointers! Garbage collection
  - 100% portable
  - Integrates standard libraries for:
    - User interfaces
    - Distributed objects
    - Threads
    - XML processing
    - And much more...
  - Can be run within a web browser (Netscape, Mozilla, MS Explorer, etc.)
  - Origin: augment HTML for more dynamic pages
  - Versions: 1995 - 1.0, 1997 - 1.1, 1998 - 1.2, 2000 - 1.3, 2001 - 1.4...

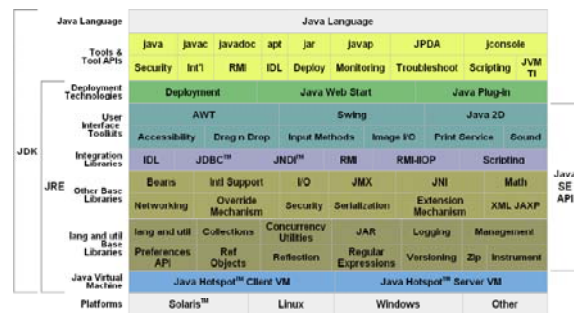
2

## Java Development Kit (JDK)

- Java Runtime Environment (JRE)
  - Java Virtual Machine (JVM)
  - Java API: basic language + standard class library
- Bytecode compiler
- Other compilers, tools and utilities: debuggers, applet viewer, javadoc, RMI compiler, etc.

3

## Java SE 6 Platform



4

## Java vs. C++

### Java

Interpreted  
 Totally portable  
 Dynamic memory  
 Garbage collection  
 No pointers  
 All methods are virtual  
 No multiple inheritance  
 Complete class information at runtime  
 Support for generic types (wrappers)

### C++

Compiled: more efficient (± 8:1)  
 Non-portable aspects  
 Automatic memory managed by RTS  
 Dynamic memory managed by programmer  
 Virtual and non-virtual methods  
 Supports multiple inheritance  
 Very limited (dynamic\_cast)  
 Pointers to void

5

### Java

Standard support for User Interfaces: AWT, Swing  
 Standard support for concurrent programming  
 Standard support for distributed objects: RMI  
 Can be run within web browsers (applets)  
 Language homogeneity  
 Well designed language

### C++

Not included in the language: Microsoft MFC, Borland OWL, etc.  
 Not included in the language  
 Not included in the language: CORBA, ActiveX, etc.  
 Language variations and exceptional cases  
 Compatible with C

6

## Language Elements

### Example

```
// package...
// import...

class Person {
    String name;
    int age;
    void showData () {
        System.out.println ("name: " + name + "\nAge: " + age);
    }
}

class MainClass {
    static void main (String args[]) {
        Person x = new Person ();
        x.name = "John Smith";
        x.age = 25;
        x.showData ();
    }
}
```

- Class definitions
- Naming conventions
- Main method
- Object declaration vs. creation
- Access to object members
- Strings
- String concatenation
- Packages
- Variable scope
- Instance vs. class variables and methods

```
class Person {
    String name;
    int age;
    void showData () {
        System.out.println ("name: " + name + "\nAge: " + age);
    }
    Person (String str, int n) {
        name = str;
        age = n;
    }
}

class MainClass {
    static void main (String args[]) {
        Person x = new Person ("John Smith", 25);
        x.showData ();
    }
}
```

9

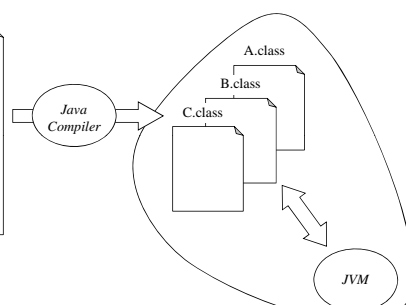
## Compilation

example.java

```
class A {
    ...
}

class B {
    ...
}

class C {
    ...
}
```



10

## Lexical Details

- White space
- Sentences end by ';' ;'
- Comments
  - // up to end of line
  - /\* \*/ more than one line
- Case sensitive
- Naming conventions

11

## Values and Variables

### Basic Types

byte	1 byte
char	2 bytes (unsigned, Unicode characters, superset of ASCII)
short	2 bytes
int	4 bytes
long	8 bytes
float	4 bytes
double	8 bytes
boolean	1 bit (true or false, not compatible with numeric types)

### Variables and literals

Declaration, use, lexical scope, etc. similar to C

12

## Compatibility between Numeric Types

```
byte b = 42;
char c = 'a'; // also valid: c = 97;
short s = 1024;
int i = 50000;
long l = 120000;
float f = 5.67f;
double d = .1234;
double result = (f*b) + (i/c) - (d*s);
System.out.println ((f*b) + " + " + (i/c) + " - " + (d*s));
System.out.println ("result = " + result);
```

- |                           |                                |                              |
|---------------------------|--------------------------------|------------------------------|
| ▪ <b>Widen: automatic</b> | ▪ <b>Narrow: explicit cast</b> | ▪ <b>char: explicit cast</b> |
| f = c;                    | c = (char) i;                  | s = (short) c;               |
| d = s;                    | b = (byte) d;                  | c = (char) s;                |
| f = l;                    | f = (float) d;                 | c = (char) b;                |

13

## Operators

- 46 operators
- **Numeric**  
+, -, \*, /, %  
+=, -=, \*=, /=, %=, --, ++
- **Logic**  
&, |, ^, !, &&, ||
- **Bit operators**  
&, |, ^, -, >>, <<
- **Relationals**  
- Any type: ==, !=  
- Numeric types: >, <, >=, <=
- **Conditional**  
- If-then-else:(condition)? action1 : action2

14

## Operator Precedence

[ ]	.	(params)	expr++	expr--	
++expr	--expr	+expr	-expr	~	!
new	(type)expr				
*	/	%			
+	-				
<<	>>	>>>			
<	>	<=	>=		instanceof
==	!=				
&					
^					
&&					
? :					
=	*=	/=	%=	&=	^=
	=	<<=	>>=	>>>=	

15

## Arrays

- **Declaration**  
int a[];
- **Memory allocation**  
a = new int [3];  
int b[] = {34, 23, 9, 1, 18};
- **Value assignment**  
a[1] = 419;  
b = a; // Valid: arrays are not constants
- **Errors**  
a[6] = 7; // Out of range  
a = {51, 29, 7}; // Valid only at initialization  
int c[5]; // Dimension only when calling new  
char str[] = "hello"; // Strings are not arrays

16

## Multidimensional Arrays

```
float a[][] = new float [4][3];
float m[][] = new float [4]{};
m[0] = new float [2];
m[1] = new float [5];
m[2] = new float [m[1].length];
float x[] = {4.5, 8/3, m[2][1]};
m[3] = x; // Or any expression that returns a float[]
a[0] = m[1]; // Array variables are not constants
a[2, 1] = 7.3 // Syntax error: a[2][1]
```

17

## Conditionals

```
if (condition) action1 [else action2];

switch (expression) {
  case value1:
    ...
    break;
  ...
  case valuen:
    ...
    break;
  default:
    ...
}
```

Annotations in the original image:  
 - An arrow points from "byte, char, short or int" to the "value<sub>1</sub>" case label.  
 - An arrow points from "must be literals" to the "value<sub>n</sub>" case label.

18

## Iterations

```
while (condition) {  
    ...  
}  
  
do {  
    ...  
} while (condition)  
  
for (initialization; condition; iteration) {  
    ...  
}
```

19

## break

```
boolean t = true;  
a: {  
    b: {  
        c: {  
            System.out.println ("Before break");  
            if (t) break b;  
            System.out.println ("This is not executed");  
        }  
        System.out.println ("This is not executed");  
    }  
    System.out.println ("After b");  
}
```

20

## continue

```
for (int i = 0; i < 10; i++) {  
    System.out.print( i + " ");  
    if (i % 2 == 0) continue;  
    System.out.println ();  
}  
  
outer: for (int i = 0; i < 10; i++) {  
    for (int j = 0; j < 10; j++) {  
        if (j > i) {  
            System.out.println ();  
            continue outer;  
        }  
        System.out.print (" " + (i * j));  
    }  
}
```

21