Java Enterprise Application Development

Lecture 2

Variables, Operators, Control Flow Statements, and Arrays

Dr. Fan Hongfei

17 September 2025

Getting Started with Java Programming

- Step 1: download and install the latest JDK
  - https://www.oracle.com/java/technologies/downloads/
- Step 2: download and install an IDE
  - **IntelliJ IDEA (recommended)**
  - Visual Studio Code (considerable)
  - Eclipse
- Step 3: Go!
  - A *Hello World* example

Variables

- Four kinds of variables
  - Instance Variables (Non-Static Fields), Class Variables (Static Fields), Local Variables, Parameters
- Naming
  - Case-sensitive, beginning with a letter, $, or _
  - Subsequent characters may be letters, digits, $, _
  - Must not be a keyword or reserved word
  - If the name consists of only one word, spell that word in all lowercase letters; if it consists of more than one word, capitalize the first letter of each subsequent word
  - If your variable stores a constant value, capitalize every letter and separate subsequent words with the underscore character

Primitive Data Types

- Java is **statically-typed**: variables first declared, then used
- 8 primitive data types
  - byte, short, int, long
  - float, double
  - boolean
  - char
- Special support for character strings
  - String, **immutable**

Default Values

- Default will be zero or null

Data Type	Default Value (for fields)
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
String (or any object)	null
boolean	false

Literals

- A literal is the source code representation of a fixed value
- Integer Literals
  - An integer literal is of type long if it ends with the letter L or l; otherwise it is of type int
  - Decimal, Hexadecimal (0x~), Binary (0b~)
- Floating-Point Literals
  - A floating-point literal is of type float if it ends with the letter F or f; otherwise its type is double
  - Can also be expressed using E or e (for scientific notation)
- Character and String Literals

Operator Precedence

Operators	Precedence
postfix	expr++ expr--
unary	++expr --expr +expr -expr ~ !
multiplicative	* / %
additive	+ -
shift	<< >> >>>
relational	< > <= >= instanceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
logical AND	&&
logical OR	||
ternary	? :
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Assignment, Arithmetic, and Unary

- Simple Assignment Operator: =
- Arithmetic Operators: +, -, *, /, %
  - You may combine the arithmetic operators with the simple assignment operator to create compound assignments
  - The + operator can also be used for concatenating two strings together
- Unary Operators: +, -, ++, --, !

Equality, Relational, and Conditional

- Equality and Relational Operators: ==, !=, >, >=, <, <=
- Conditional Operators: &&, ||
  - Short-circuiting: the second operand is evaluated only if needed
- Ternary operator: a ? b : c
- **Type Comparison Operator: instanceof**
  - Test if an object is an instance of a class, an instance of a subclass, or an instance of a class that implements a particular interface

Bitwise and Bit Shift

- Unary bitwise complement operator "~" inverts a bit pattern
- Signed left shift operator "<<" shifts a bit pattern to the left, and the signed right shift operator ">>" shifts a bit pattern to the right
- Bitwise & operator performs a bitwise AND operation
- Bitwise ^ operator performs a bitwise exclusive OR operation
- The bitwise | operator performs a bitwise inclusive OR operation

The "Hello World!" Application

- Source code and bytecode
- General structure
- The main method
- Source code comments (3 kinds)
  - **Javadoc**: specific for Java

Expressions, Statements, and Blocks

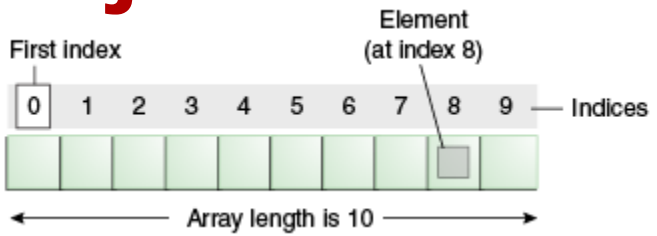
- Expression: made up of variables, operators, and method invocations, that evaluates to a single value
- Statement: a complete unit of execution
  - Expression statements
    - Assignment expressions
    - Any use of ++ or --
    - Method invocations
    - Object creation expressions
  - Declaration statements
  - Control flow statements
- Block: a group of zero or more statements between balanced braces

Control Flow Statements

- The if-then Statement
- The if-then-else Statement
- The switch Statement
  - Working with byte, short, char, int, enumerated types, String, and a few special classes that wrap certain primitive types
- The while and do-while Statements
- The for Statement
- **Branching Statements**: break, continue, return

Arrays

- Array: **a container object**



- Declaring a variable to refer to an array
  - `int[] anArray;`
- Creating, initializing, and accessing an array
  - `anArray = new int[10];`
- Length of array: `arrayRefVar.length`
- Shortcut syntax to create and initialize an array
  - `int[] anArray = {100, 200, 300, 400, 500, 600, 700, 800};`
- **The enhanced for statement: recommended**

Duplicating Arrays

- Whenever you need to duplicate an array
  - Using the assignment (=) statement?
- Correct way: using arraycopy utility!
  - `public static void arraycopy(Object src, int srcPos, Object dest, int destPos, int length)`
- Related issue: passing arrays to methods
  - Sample code and explanation
  - Java is "**passing by value**"

Array Manipulations

- Methods for performing array manipulations by the java.util.Arrays class
  - arraycopy, copyOfRange
  - equals
  - binarySearch
  - fill
  - sort, parallelSort (demonstration of performance difference)
  - Many more (for your private study)

Multidimensional Arrays

- Declaration and creation

```
int[][] myNumbers = new int[4][6];
int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7, 8} };
```

- In Java, a multidimensional array is an array **whose components are themselves arrays**
  - Rows are allowed to vary in length: **ragged array**

```
int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7}, {8, 9} };
```

  - Initialization issue

```
int[][] myNumbers = new int[3][];
```

ArrayList

- Declaration and initialization
- add, remove
- get, set
- size, clear, isEmpty
- contains, indexOf