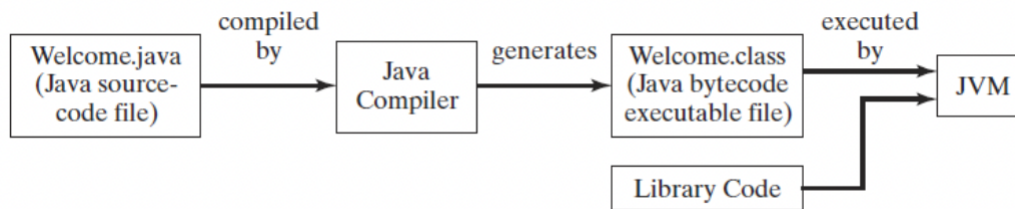


# 1 Introduction to Java

- What is Java?
  - An object-oriented language invented by James Gosling in 1994 at Sun Microsystems
  - Write once, run anywhere (WORA)
  - Widely-used in industry
  - Used to develop software running on:
    - \* Desktop Computers
    - \* Servers
    - \* Mobile devices
- Java Programs
  1. Writing the source code using a text editor
  2. Translating the source code into Java bytecode using a compiler
    - Bytecode is similar to machine instructions but is architecture neutral and can run on any platform that has a Java Virtual Machine (JVM)
  3. Executing the bytecode
    - The JVM is an interpreter: it translates bytecode into the target machine language code one at a time rather than the whole program as a single unit
    - Each step is executed immediately after it is translated



- Integrated Development Environment
  - A system comprising several tools that facilitate software development and testing
  - Popular IDEs:
    - \* Eclipse
    - \* NetBeans
    - \* IntelliJ
- Data Types
  - Eight primitive types
    - \* byte, char, short, int, long, float, double, boolean
  - Objects
    - \* Defined using **classes**
    - \* Java provides wrapper classes to use primitive types as objects (e.g. Integer, Double, etc)
- Numeric Primitive Types

<i>Name</i>	<i>Range</i>	<i>Storage Size</i>
<b>byte</b>	$-2^7$ to $2^7 - 1$ (−128 to 127)	8-bit signed
<b>short</b>	$-2^{15}$ to $2^{15} - 1$ (−32768 to 32767)	16-bit signed
<b>int</b>	$-2^{31}$ to $2^{31} - 1$ (−2147483648 to 2147483647)	32-bit signed
<b>long</b>	$-2^{63}$ to $2^{63} - 1$ (i.e., −9223372036854775808 to 9223372036854775807)	64-bit signed
<b>float</b>	Negative range: $-3.4028235E + 38$ to $-1.4E - 45$ Positive range: $1.4E - 45$ to $3.4028235E + 38$	32-bit IEEE 754
<b>double</b>	Negative range: $-1.7976931348623137E + 38$ to $-4.9E - 324$ Positive range: $4.9E - 324$ to $1.7976931348623137E + 38$	64-bit IEEE 754

- Classes
  - A typical Java class includes the following:
    - \* Data fields to represent the state of an object
    - \* Methods to represent the behavior of an object. Each method has:
      - A return type (**void** if nothing is returned)
      - Zero or more arguments
    - \* Special type of methods, known as constructors, that perform initialization actions. A constructor:
      - Has no return type (not even **void**)
      - Has zero or more arguments
      - Should have the same name as the class
      - Is invoked using the **new** operator
  - Instantiation is creating an object (or an instance of a class)
- The *main* method
  - The main method is the entry point where the program begins execution
  - Should have the following form:
 

```
public static void main(String [] args) {
    //write your code here
}
```
- Default values
  - The default value of a data field is:
    - \* **null** for a reference type    \* **0** for a numeric type
    - \* **false** for a boolean type    \* **'\u0000'** for a char type
  - Java assigns no default value to a local variable inside a method
- Scope
  - The scope of fields and methods is the entire class
  - The scope of a local variable starts from its declaration until the end of the block that contains it
- Differences between Variables of Primitive Types and Reference Types
  - Every variable represents a memory location that holds a value
  - For a variable of a primitive type, the value is of the primitive type
  - For a variable of a reference type, the value is a reference to where an object is located (i.e. a pointer)
  - When you assign one variable to another:
    - \* For a variable of a primitive type, the real value of one variable is assigned to the other variable
    - \* For a variable of a reference type, the reference of one variable is assigned to the other variable.
- The *this* reference
  - The **this** *keyword* is the name of a reference that an object can use to refer to itself
  - It can be used to reference the object's instance members
- The *static* modifier
  - Static fields/methods can be accessed from a reference variable or from their class name
  - Non-static (or instance) fields/methods can only be accessed from a reference variable

- Arrays

- An array is a data structure that represents a collection of the same types of data
- Once an array is created, its size is fixed
  - \* e.g. `int [] A = new int[10];`
- The size of an array A can be found using `A.length`
- When an array is created, its elements are assigned the default value
- Array elements could be initialized individually
  - \* e.g. `A[0] = 5;`
- Array initializer (combines declaration, creation, and initialization)
  - \* e.g. `double[] myList = 1.9, 2.9, 3.4, 3.5;`

- Two-dimensional Arrays

- The syntax for declaring a two-dimensional array is:
  - \* `elementType [][] arrayRefVar;` (e.g. `int[][] matrix;`)
- For a two-dimensional array A, `A.length` returns the number of rows
- Two-dimensional array examples:

	[0]	[1]	[2]	[3]	[4]
[0]	0	0	0	0	0
[1]	0	0	0	0	0
[2]	0	0	0	0	0
[3]	0	0	0	0	0
[4]	0	0	0	0	0

`matrix = new int[5][5];`

	[0]	[1]	[2]	[3]	[4]
[0]	0	0	0	0	0
[1]	0	0	0	0	0
[2]	0	7	0	0	0
[3]	0	0	0	0	0
[4]	0	0	0	0	0

`matrix[2][1] = 7;`

	[0]	[1]	[2]
[0]	1	2	3
[1]	4	5	6
[2]	7	8	9
[3]	10	11	12

```
int[][] array = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9},
    {10, 11, 12}
};
```