OXFORD
UNIVERSITY PRESS

OXFORD HIGHER EDUCATION
FREE ONLINE RESOURCES For Teachers and Students
Programming in Java
Revised Second Edition
Java SE8 enhancements
220 programs, followed by line-by-line explanation
400 chapter-end exercises (MCQs, Review Questions, Programming Exercises)
Key notes to highlight important concepts
Q&A to help students prepare for interviews
Advanced Java topics such as J2EE, Servlets, Beans, and Jar Files
Prelude to Java SE9 (online resource)
Sachin Malhotra
Saurabh Choudhary

Programming in Java

Revised 2nd Edition

Sachin Malhotra & Saurabh Choudhary

Chapter 2

Getting started with Java

Objective

- Know the Java Features and its Runtime Environment
- Get familiar with new releases in Java
- Understand the basic structure of a Java program
- Get into the details about JDK Installation
- Know about the various constituents of JDK and its development environments

Introduction

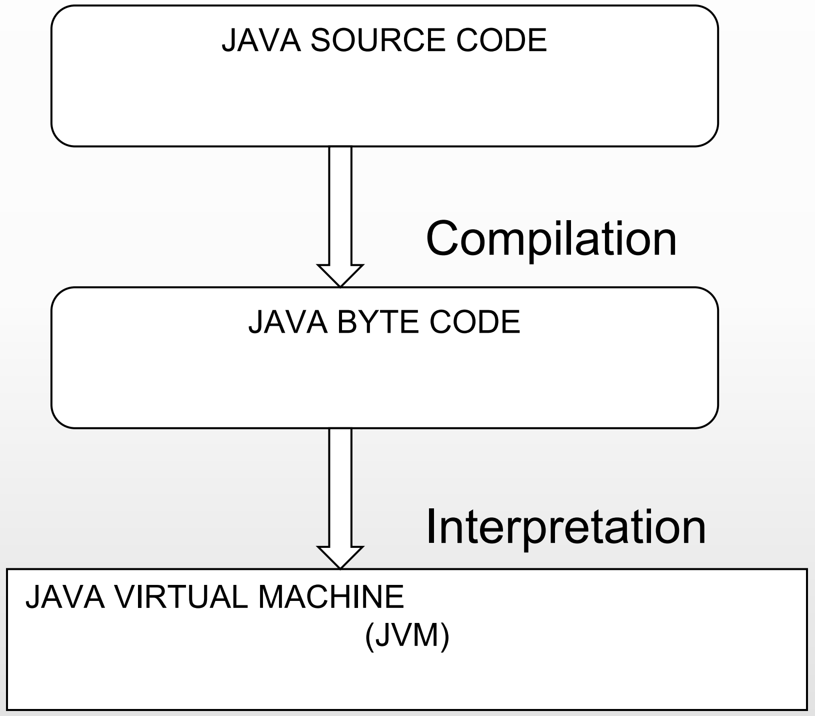
- Java is a programming language developed by James Gosling and others in 1994.
- Originally named Oak ,was developed as a part of the Green project at Sun Microsystems.
- Java 9 is latest stable release.

Java Essentials

- A high level language
- Java Bytecode – intermediate code
- Java Virtual Machine (JVM) – interpreter for bytecode

Java Runtime

Java Runtime Environment includes JVM, class libraries and other supporting files.



Java Approach

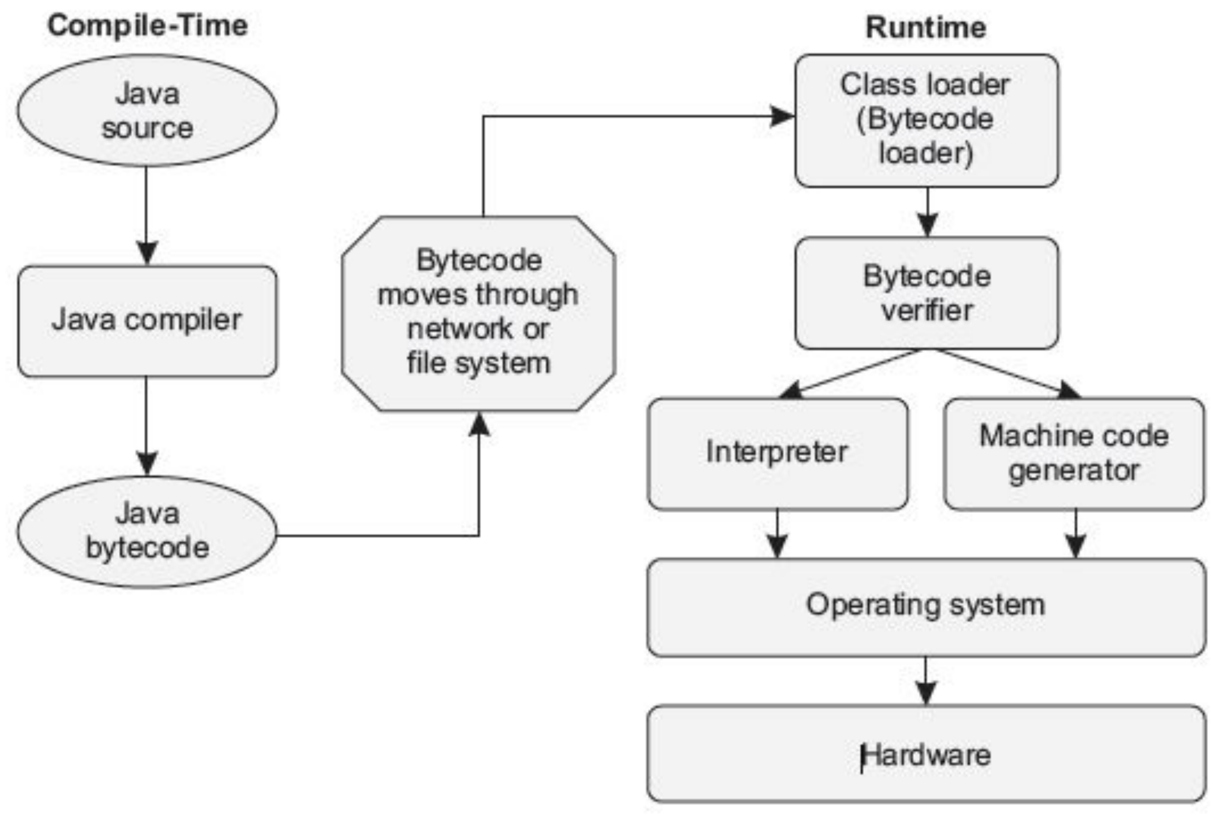


Fig. 2.6 Compilation and Interpretation in Java

Java Features

- Platform Independence
- Object oriented
- Compiled and interpreted
- Robust
- Security
    - Strictly typed language
    - Lack of pointers
    - Garbage collection
    - Strict compile time checking
    - Sandbox security

Java Features

- Multithreaded
- Dynamic binding
- Performance
- Networking
- No pointers
- No global variables
- Automatic Garbage collection

Java 5 New Features

- Autoboxing and unboxing
- Enhanced For loop
- Metadata
- Variable arguments
- Static import
- Graphics enhancemnets
- Generics
- Enum
- StringBuilder class

Java 6 New Features

- Enhancements in Collections API
- Console class
- Jar and Zip enhancements
- Enhancements to Network Interface
- Enhancements in Java web start and plug in

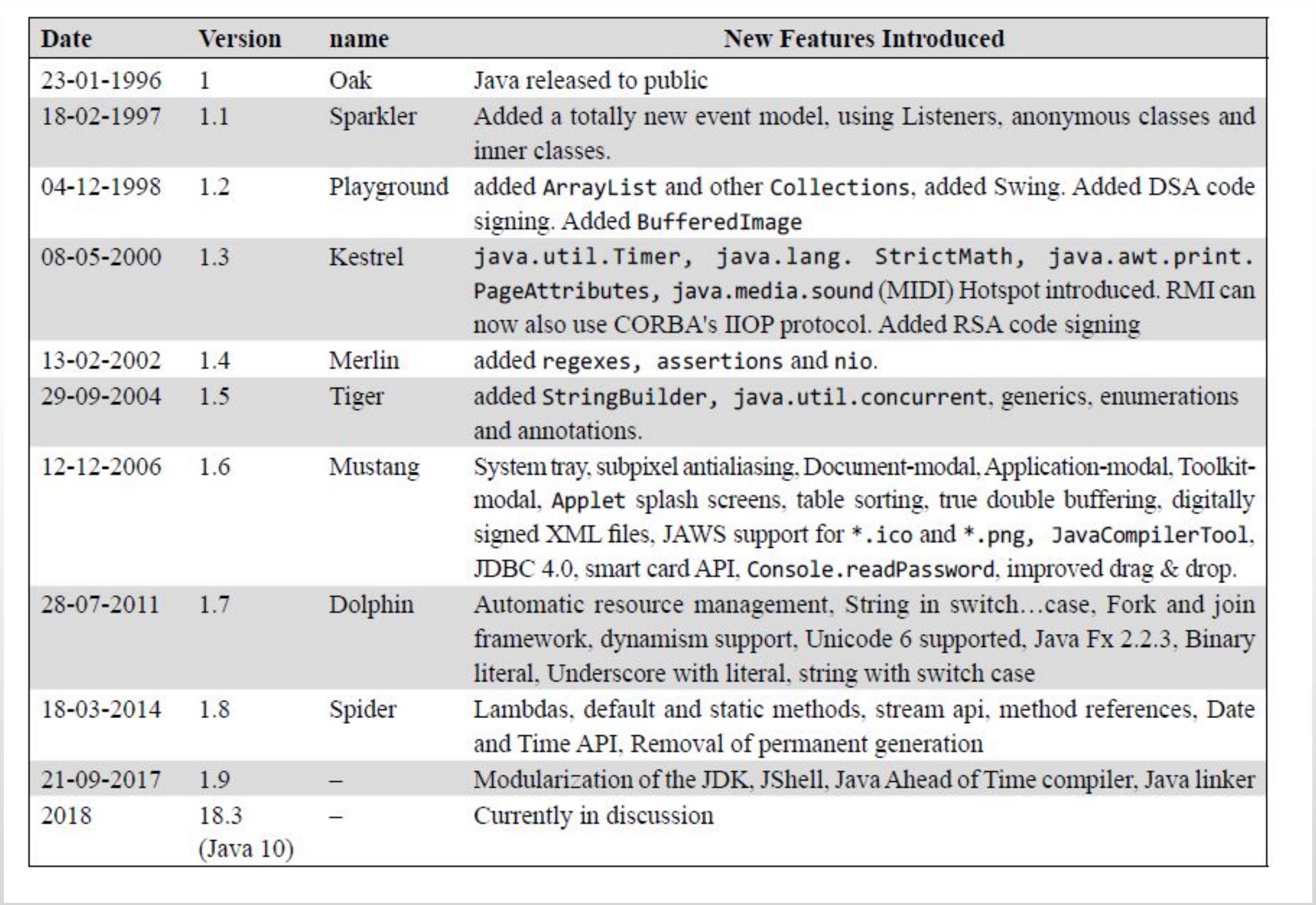
Java 7 New Features

- String in switch…case Statement
- Unicode 6.0.0 Support
- Binary Literals and Numeric Literals (with Underscores)
- Automatic Resource Management
- Improved Exception Handling
- nio 2.0 (Non-blocking I/O)—New File System API

Java 7 Features (contd.)

- Fork and Join for parallel processing
- Supporting Dynamism using invoke dynamic to let JVM resolve type info at runtime
- No need of Diamond Operator <> on right side of the expression
- Swing Enhancements
- Java FX 2.2.3 provides the new GUI toolkit for creating rich cross platform user interfaces

Comparison of Java Versions



Date	Version	name	New Features Introduced
23-01-1996	1	Oak	Java released to public
18-02-1997	1.1	Sparkler	Added a totally new event model, using Listeners, anonymous classes and inner classes.
04-12-1998	1.2	Playground	added `ArrayList` and other `Collections`, added Swing. Added DSA code signing. Added `BufferedImage`
08-05-2000	1.3	Kestrel	`java.util.Timer, java.lang. StrictMath, java.awt.print. PageAttributes, java.media.sound` (MIDI) Hotspot introduced. RMI can now also use CORBA's IIOP protocol. Added RSA code signing
13-02-2002	1.4	Merlin	added `regexes, assertions` and `nio`.
29-09-2004	1.5	Tiger	added `StringBuilder, java.util.concurrent`, generics, enumerations and annotations.
12-12-2006	1.6	Mustang	System tray, subpixel antialiasing, Document-modal, Application-modal, Toolkit-modal, `Applet` splash screens, table sorting, true double buffering, digitally signed XML files, JAWS support for `*.ico` and `*.png, JavaCompilerTool`, JDBC 4.0, smart card API, `Console.readPassword`, improved drag & drop.
28-07-2011	1.7	Dolphin	Automatic resource management, String in switch…case, Fork and join framework, dynamism support, Unicode 6 supported, Java Fx 2.2.3, Binary literal, Underscore with literal, string with switch case
18-03-2014	1.8	Spider	Lambdas, default and static methods, stream api, method references, Date and Time API, Removal of permanent generation
21-09-2017	1.9	–	Modularization of the JDK, JShell, Java Ahead of Time compiler, Java linker
2018	18.3 (Java 10)	–	Currently in discussion

Differences between C++ and Java

- **Multiple Inheritance not allowed**
    - Multi-level inheritance is enforced, which makes the design clearer. Multiple inheritance among classes is not supported in java. Interfaces are used for supporting multiple inheritance.
- **Common parent**
    - All classes are *single-rooted*. The class Object is the parent of all the classes in java.
- **Packages**
    - The concept of *packages* is used, i.e. a large, *hierarchical namespace* is provided. This prevents naming ambiguities in libraries.
- **In-source documentation**
    - *In-source code documentation* comments are provided. Documentation keywords are provided for example: @author, @version, etc.
- **All code inside class**
    - All code resides inside a class. Global data declaration outside the class is not allowed. However, static data within classes is supported.

Differences between C++ and Java

- **Operator overloading**
  - Operator overloading is not supported in java but there are few operators which are already overloaded by java, e.g. '+'. Programmers do not have the option of overloading operators.
- **Explicit boolean type**
  - boolean is an explicit type, different from int. Only two boolean literals are provided i.e. true and false. These cannot be compared with integers 0 and 1 as used in some other languages.
- **Array length accessible**
  - All array objects in java have a length variable associated with them to determine the length of the array.

Differences between C++ and Java

- **goto**
  - Instead of goto, break and continue are supported.
- **Pointers**
  - There are no pointers in java.
- **null pointers reasonably caught**
  - Null pointers are caught by a NullPointerException.
- **Memory management**
  - Explicit destructor is not needed. The use of *garbage collection* prevents memory leaks and referencing freed memory.
- **Automatic variable initialization**
  - Variables are automatically initialized except local variables.
- **Runtime container bounds checks**
  - The bounds of containers (arrays, strings, etc.) are checked at runtime and an IndexOutOfBoundsException is thrown if necessary.

Differences between C++ and Java

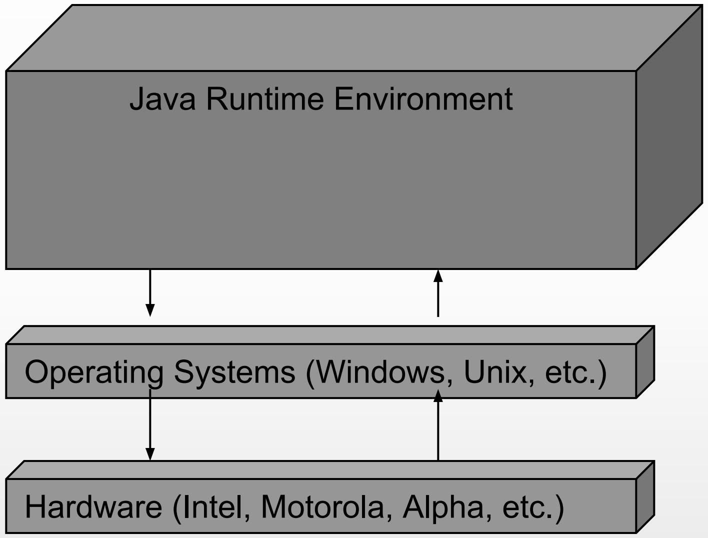
- **All definitions are well defined**
  - Methods and fields carry explicitly one of the access modifiers.
- **Sizes of the integer types defined**
  - The sizes of the integer type's byte, short, int and long are defined to be 1, 2, 4 and 8 bytes.
- **Unicode provided**
  - Unicode represents character in most of the languages, e.g. Japanese, Latin etc.
- **String class**
  - An explicit predefined String class is provided along with StringBuffer and new StringBuilder class.

Differences between C++ and Java

- **Extended utility class libraries: *package java.util***
  - Supported among others: Enumeration (an *Iterator* interface), Hashtable, Vector.
- **Multithreading support with synchronization**
  - Java supports multithreading with synchronization among them.
- **Default access specifier added**
  - By default, in java all variables, methods and classes have default privileges which are different from private access specifier. Private is the default access specifier in C++.

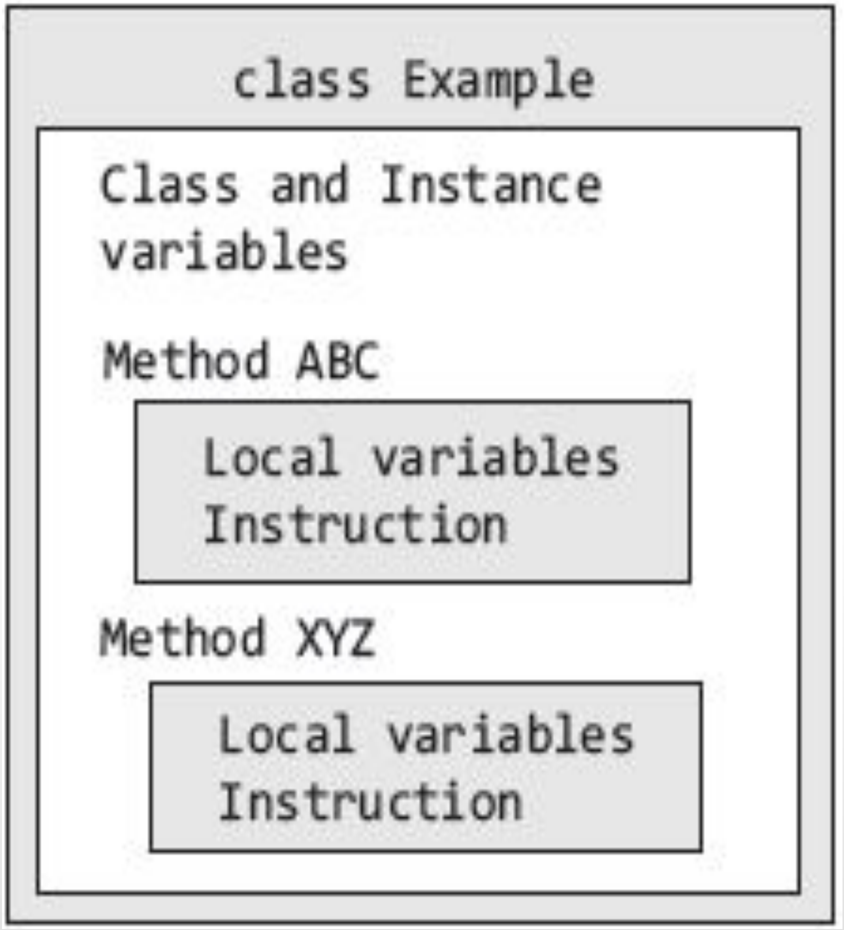
JVM and JRE

- JVM is a part of JRE


Java Runtime Environment
Operating Systems (Windows, Unix, etc.)
Hardware (Intel, Motorola, Alpha, etc.)

Program Structure

- A Java Application consists of a collection of classes.
- A class is a template containing methods and variables.



First Java Program

```
/* Call this file “Example.java”.*/

class Example {

//your program starts execution with a call to //main()

public static void main(String args[ ]){

System.out.println(“This is a simple Java program”);

}

}
```

Executing Java Program

- **Entering the source code:** text editor like notepad or any IDE
- **Saving the source code:**
  - Select File | Save As from the notepad menu.
  - In the ‘File name’ field, type “Example.java” within the double quotes.
  - In the ‘Save as type’ field select All Files (*.*).
  - Click enter to save the file.
- **Compiling & running the source**
  - type *cmd* at the run prompt.
  - move to the folder that contains the saved Example.java file.
  - compile the program using javac,
  - C:\javaeg\>javac Example.java

Executing Java Program

- Compilation creates a file called **Example.class**
- This class contains bytecode which is interpreted by JVM.
- To execute the program type the following command at the dos prompt:
  - C:\javaeg\>java Example
- The output of the program is shown below:
  - This is a simple Java program

Why save as Example.java?

- The name of the .class file will match exactly with the name of the source file.
- That is why it is a good idea to give the Java source files the same name as that of the class they contain.
- Java is case-sensitive.
- So example and Example are two different class names.

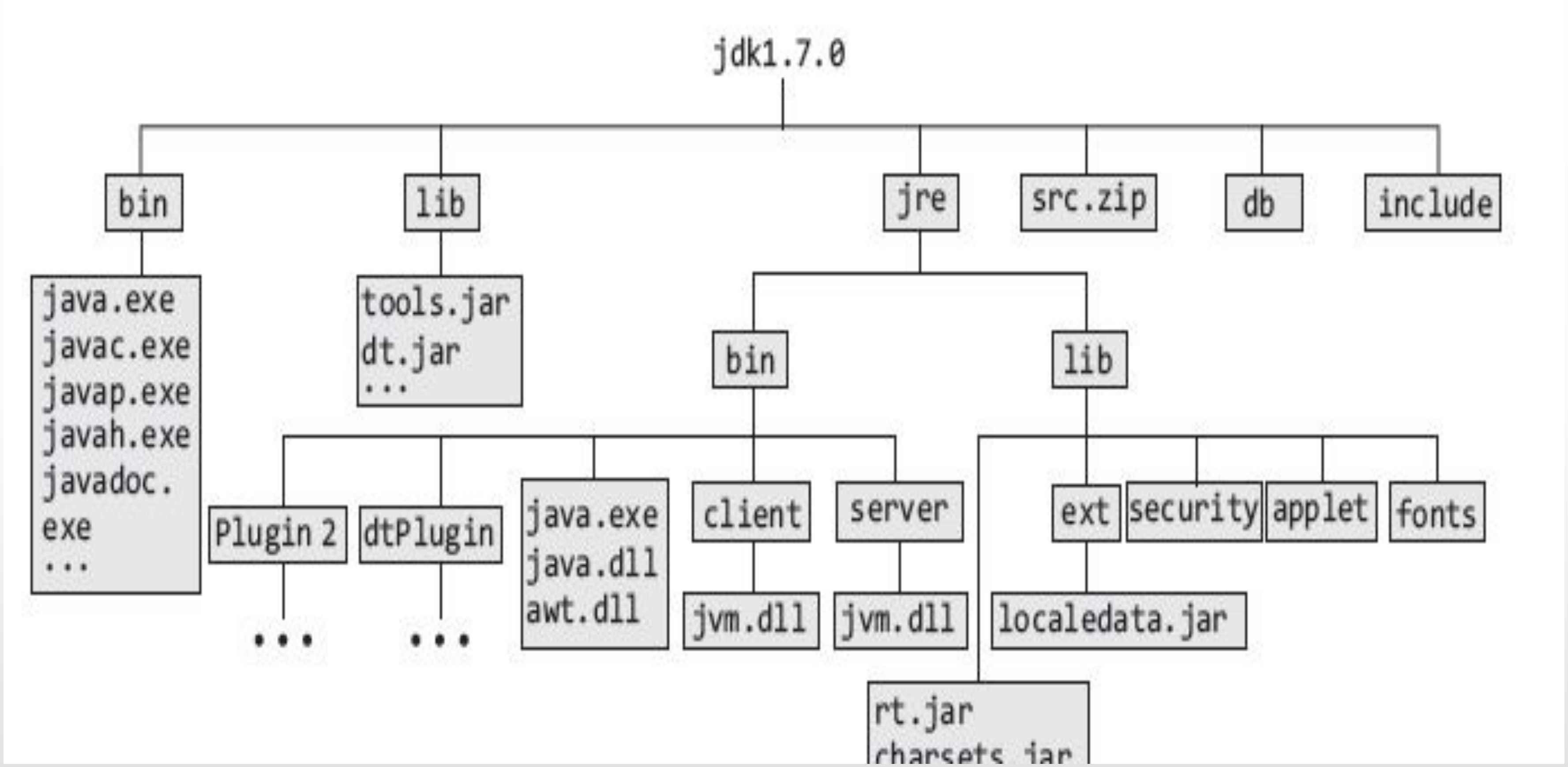
Your Turn

- Let us revise the concepts
  - What is platform independence?
  - What is the relation between JVM and JRE?
  - Differences between C++ and Java?
  - Why the source file is named after the class name in java?

Installation of Java

- Download the JDK installer.
- Run the JDK installer.
- Update PATH Environment variables.
- Test the installation – run javac and java on command prompt.

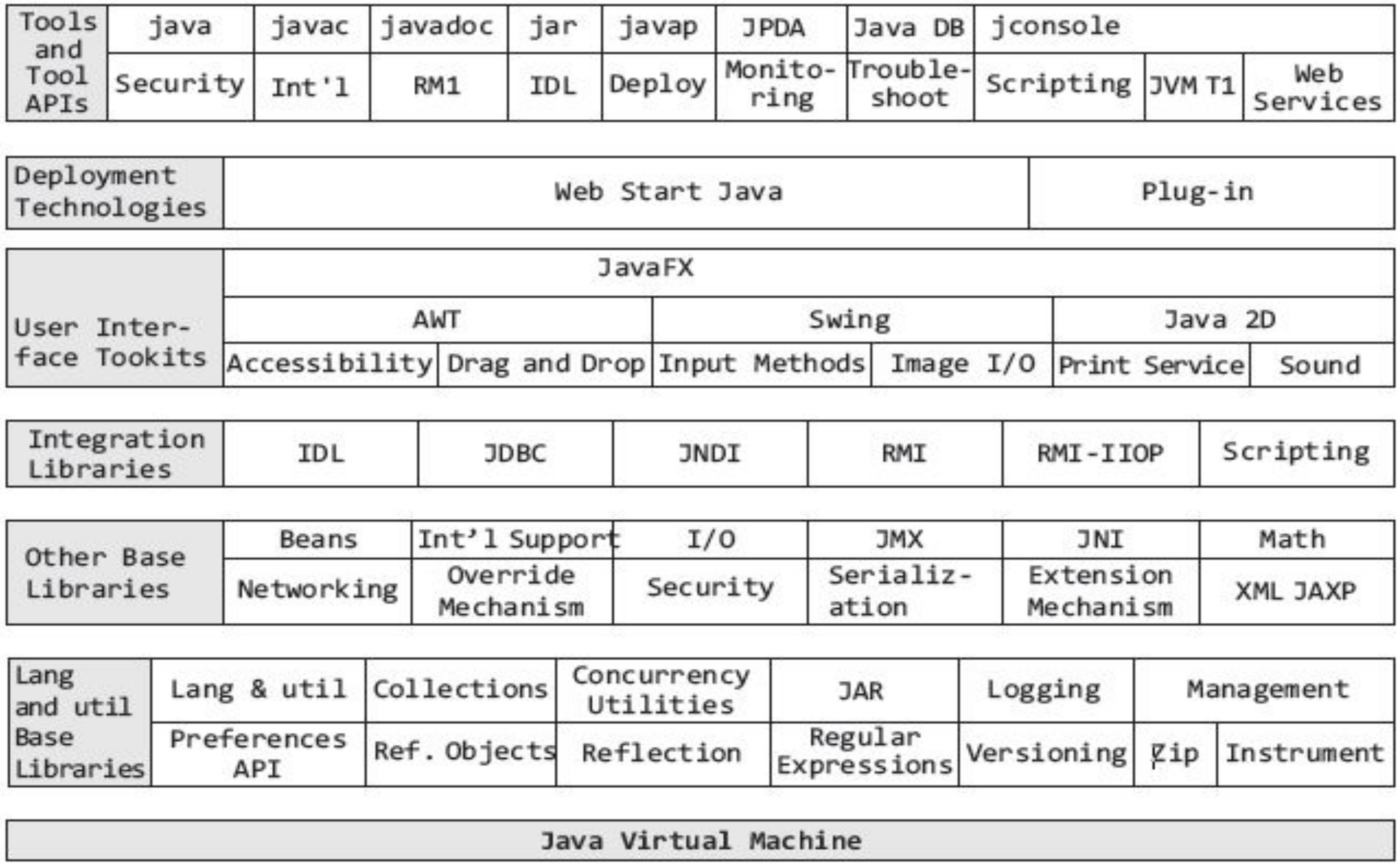
Installed Directory Structure



Installed Directory Structure

- **src.zip file** contains all the core class binaries, and is used by JDK in this form.
- **include\** directory contains a set of C and C++ header files for interacting with C and C++.
- **lib\** directory contains non-core classes like **dt.jar** and **tools.jar** used by tools and utilities in JDK.
- **bin\** The bin directory contains the binary executables for Java. For example, Java Compiler (Java), Java Interpreter (Java) , rmicompiler, (rmic) etc.
- **jre\** is the root directory for the Java runtime environment.
- **db\** contains java database.

Exploring the JDK



Tools and Tool APIs	java	javac	javadoc	jar	javap	JPDA	Java DB	jconsole		
	Security	Int'l	RM1	IDL	Deploy	Monito-ring	Trouble-shoot	Scripting	JVM T1	Web Services

Deployment Technologies	Web Start Java	Plug-in

User Inter-face Tookits	JavaFX					
	AWT		Swing		Java 2D	
	Accessibility	Drag and Drop	Input Methods	Image I/O	Print Service	Sound

Integration Libraries	IDL	JDBC	JNDI	RMI	RMI-IIOP	Scripting

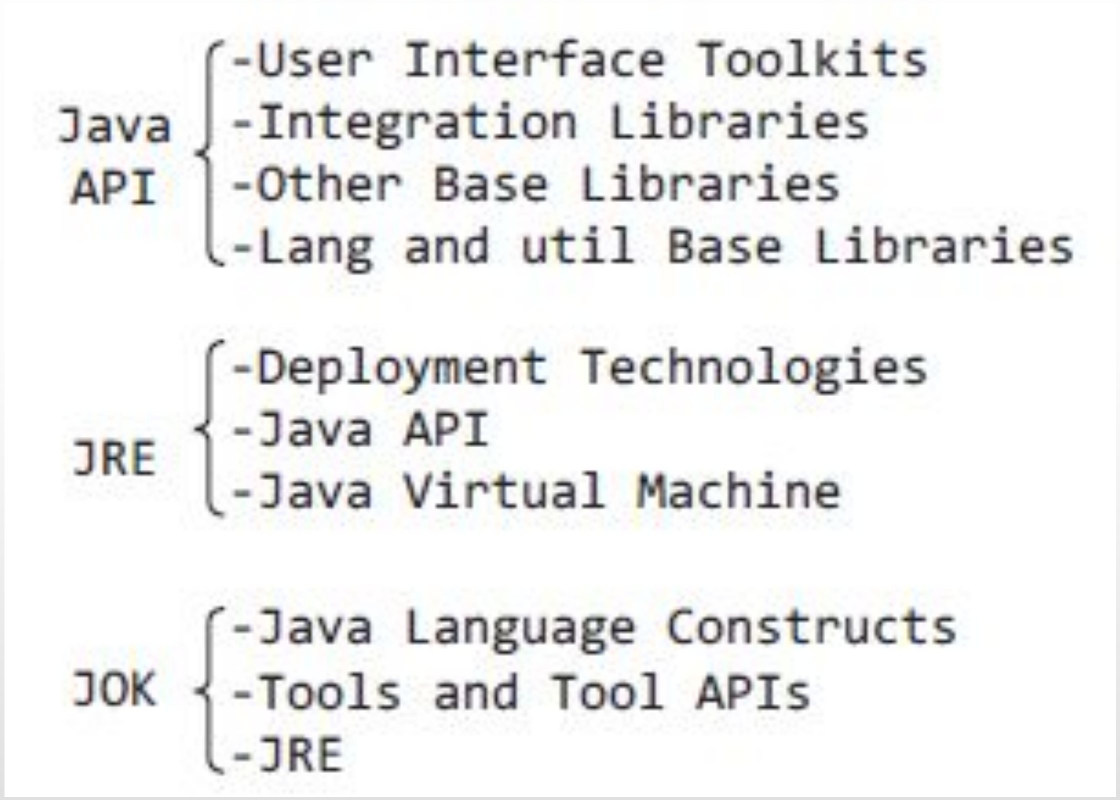
Other Base Libraries	Beans	Int'l Support	I/O	JMX	JNI	Math
	Networking	Override Mechanism	Security	Serializ-ation	Extension Mechanism	XML JAXP

Lang and util Base Libraries	Lang & util	Collections	Concurrency Utilities	JAR	Logging	Management	
	Preferences API	Ref. Objects	Reflection	Regular Expressions	Versioning	Zip	Instrument

Java Virtual Machine

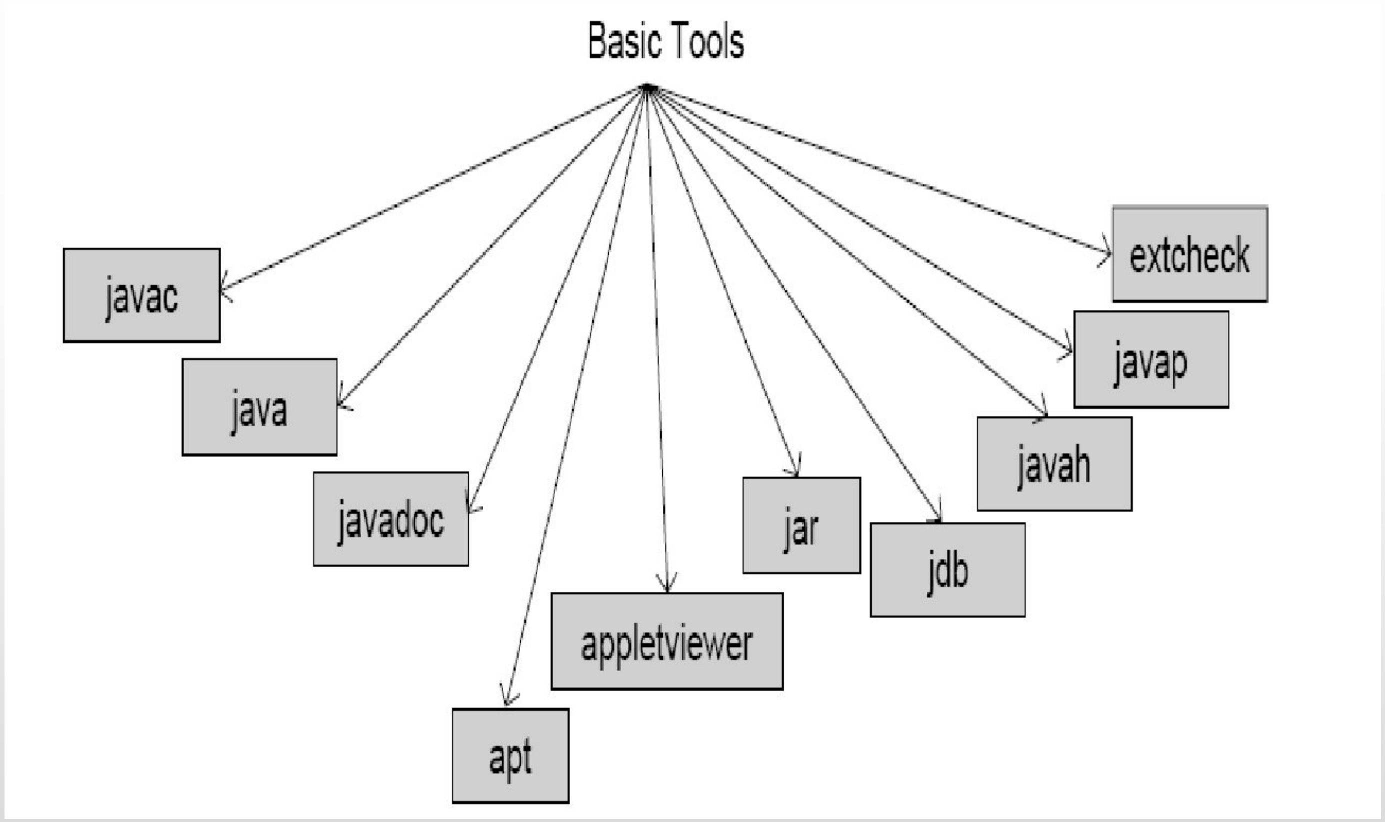
Exploring the JDK

JDK=JRE + JAVA API



Tools in JDK

- Basic Tools in Java



IDE

- Tools specifically designed for writing Java code.
- Tools offer a GUI environment to compile and debug your Java program easily from the editor environment, as well as browse through your classes etc.
- Popular IDEs
    - Eclipse
    - Netbeans
    - Kawa
    - JCreator

Summary

- Java is an object-oriented language.
- Java is designed to be platform independent, so it can run on multiple platforms.
- Every Java program consists of one or more classes.
- A class is nothing but a template for creating objects.
- In Java, code resides inside a class.
- Java bytecode executes on a special type of microprocessor.
- As there was not a hardware implementation of this microprocessor available when Java was first released, the complete processor architecture was emulated by a software known as *virtual machine*. (popularly known as JVM).