Java Programming

Binnur Kurt
binnur.kurt@ieee.org

Istanbul Technical University
Computer Engineering Department

Version 0.0.4

About the Lecturer

- **BSc**

  İTÜ, Computer Engineering Department, 1995

- **MSc**

  İTÜ, Computer Engineering Department, 1997

- Areas of Interest
  - Digital Image and Video Analysis and Processing
  - Real-Time Computer Vision Systems
  - Multimedia: Indexing and Retrieval
  - Software Engineering
  - OO Analysis and Design

Welcome to the Course

- Introduction
  - Name
  - Company affiliation
  - Title, function, and job responsibility
  - Programming experience
  - Reasons for enrolling in this course
  - Expectations for this course
- Course Hours
  - Wednesday, Computer Lab.
  - 18:30–21:30

Course Overview

1. Java Technology
2. Object-Oriented Programming
   Encapsulation, Class, Method, Attribute,
   Accessing Object Members, Constructor
3. Identifiers, Keywords, and Types
   Java Keywords, Primitive Types, Variables,
   Declarations, Assignment, Reference Type
   Constructing and initializing Objects,
   Assigning Reference Types, Pass-by-Value
4. Expressions and Flow Control
   Variable and Scope, Initializing Variables, Operators,
   Logical Operators, Branching Statement,
   Looping Statement, Special Loop Flow Control

5. Arrays
   Declaring and Creating Arrays, Initialization of Arrays
   Multidimensional Arrays, Resizing and Copying Arrays
6. Class Design
   Inheritance, Access Control, Method Overriding, Super
   Polymorphism, Virtual Method Invocation, instanceof
   Casting Objects, Overloading Constructors,
   Object and Class Classes, Wrapper Classes
7. Advanced Class Features
8. When things go Wrong: Exceptions
9. Text-Based Applications
10. Building Java GUIs
11. GUI Event Handling
12. GUI-Based Applications
13. Threads
14. Advanced I/O Streams
15. Networking

1 Java Technology

Objectives

- Describe the key features of Java Technology
- Define the terms class and application
- Write, compile, and run a simple Java Technology application
- Describe the Java Virtual Machine's function
- Define Garbage collection
- List the three tasks performed by the Java platform that handle code security

What is Java Technology?

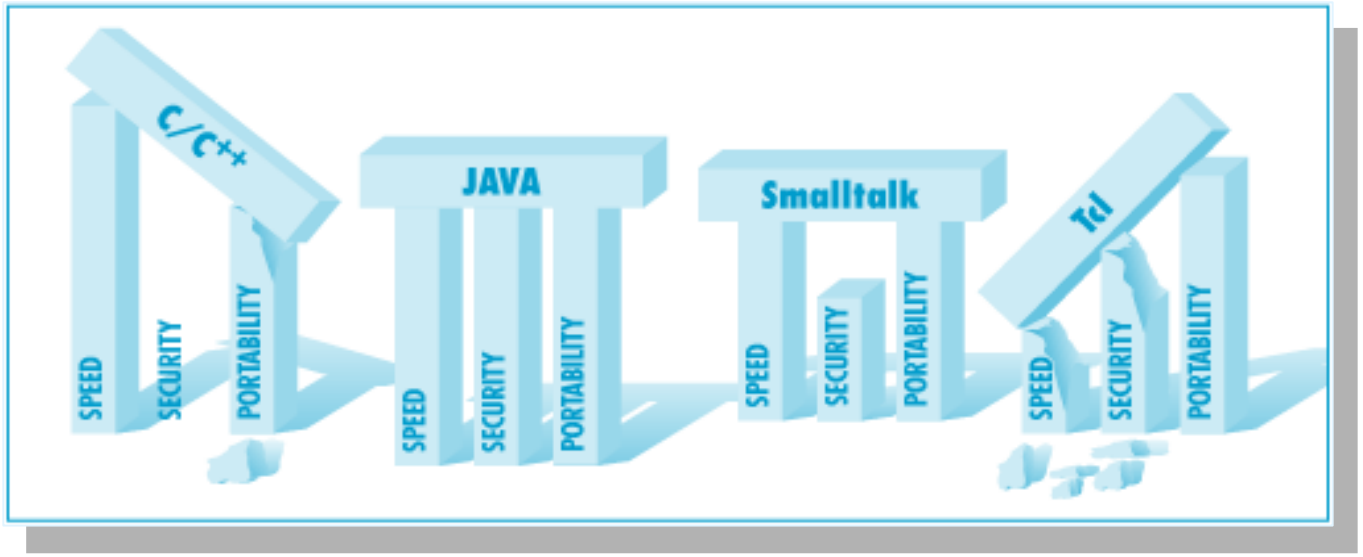
- A programming language
- A development environment
- An application environment
- A deployment environment

Primary Goals of the Java Technology

- Provides an easy-to-use language by
  - Avoiding the pitfalls of other languages
  - Being object-oriented
  - Enabling users to create streamlined and clear code
- Provides an interpreted environment for
  - Improved speed of development
  - Code portability

Primary Goals of the Java Technology

- Enables users to run more than one thread of activity,
- Load classes dynamically; that is, at the time they are actually needed,
- Supports dynamically changing programs during runtime by loading classes from disparate sources,
- Furnishes better security



Primary Goals of the Java Technology

- The following features fulfill these goals:
  - JVM,
  - Garbage Collection,
  - Code security

The Java Virtual Machine

- Provides hardware platform specifications,
- Reads compiled byte codes that are platform independent,
- Is implemented as software or hardware,
- Is implemented in a Java technology development tool or a Web browser.

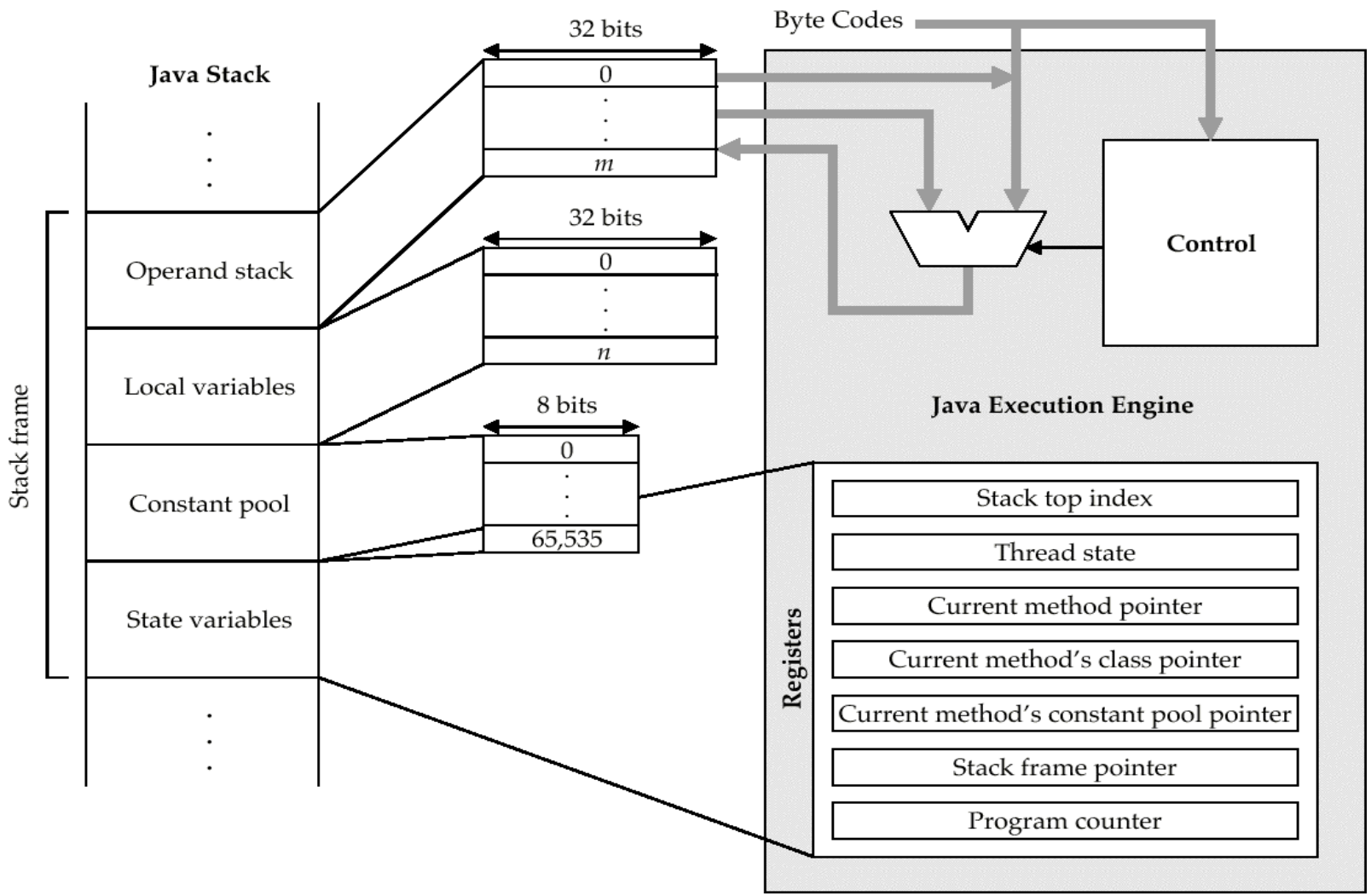
The Java Virtual Machine

- JVM provides definitions for the
  - Instruction set (central processing unit [CPU])
  - Register set
  - Class file format
  - Stack
  - Garbage-collected heap
  - Memory area

The Java Virtual Machine

- The Java Virtual Machine
  - **Bytecodes** that maintain proper type discipline form the code.
  - The majority of type checking is done when the code is compiled.
  - Every implementation of the **JVM** approved by Sun Microsystems must be able to run any compliant class file.

Java Virtual Machine Architecture



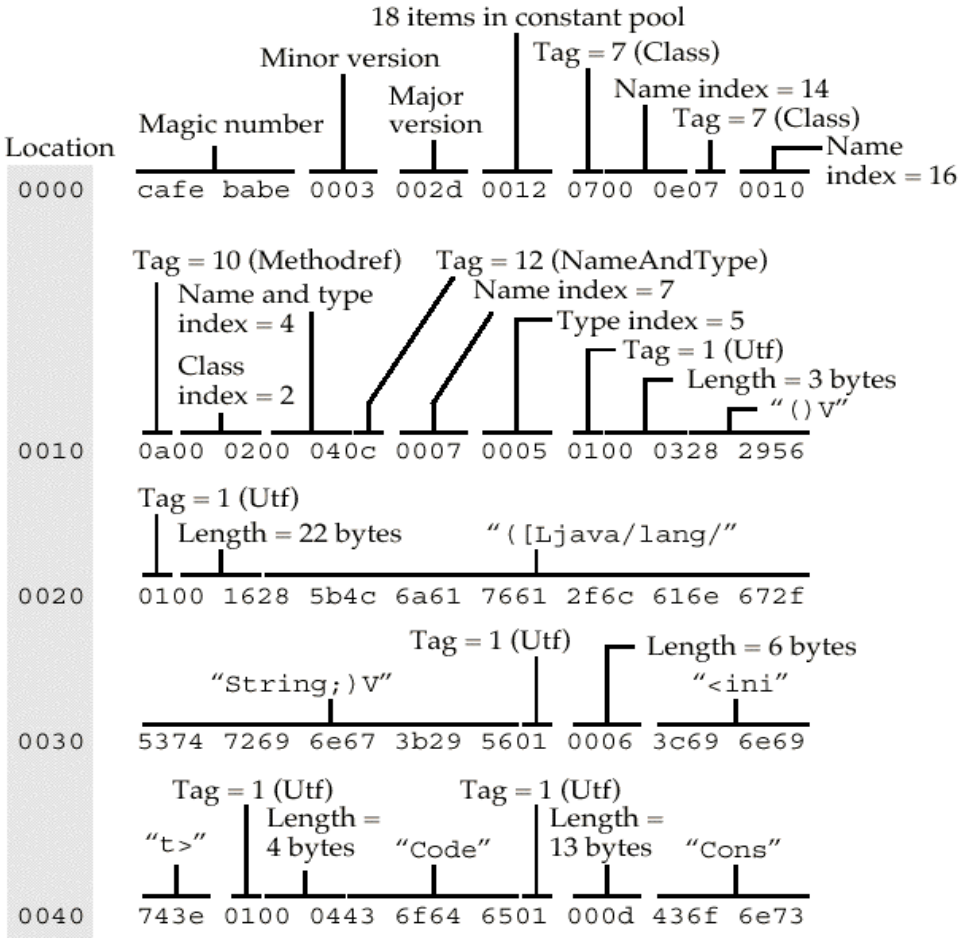
Java Program and Compiled Class File

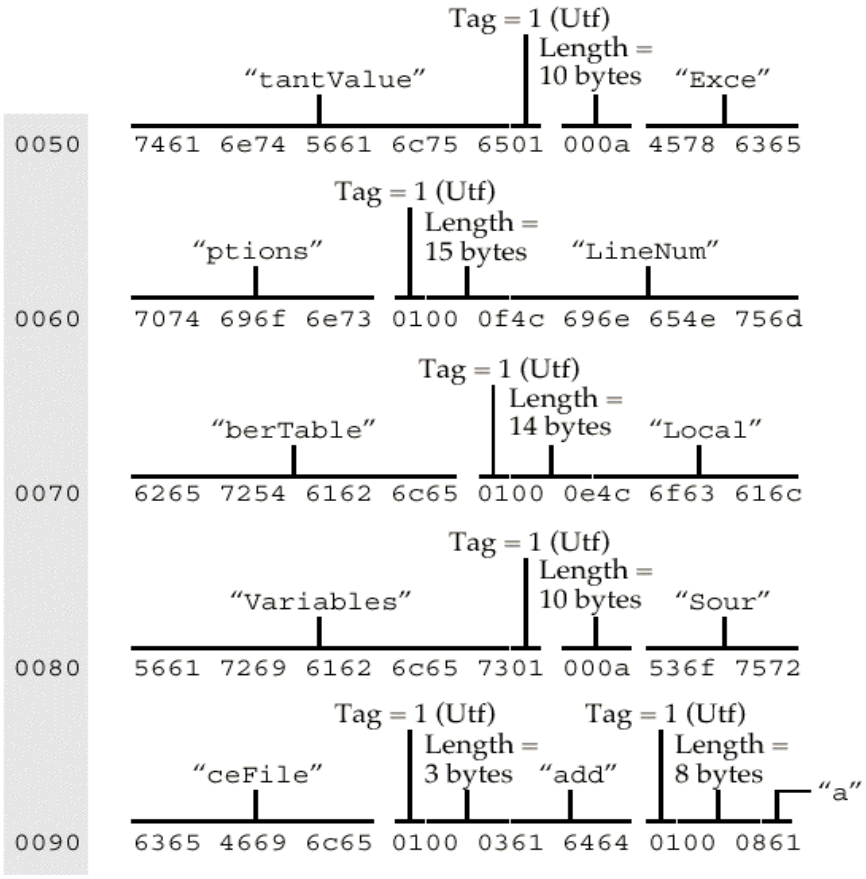

```
// This is file add.java

public class add {
    public static void main(String args[]) {
        int x=15, y=9, z=0;
        z = x + y;
        }
    }
```

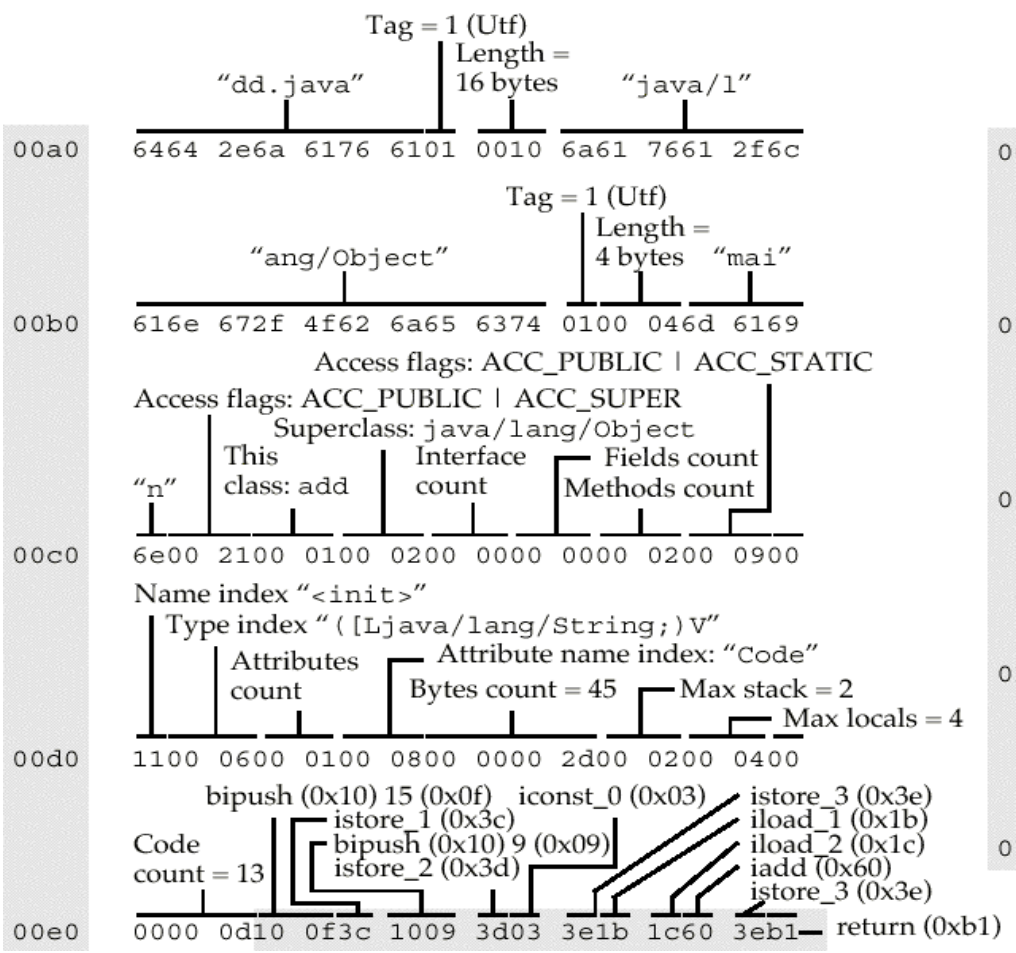
```
0000 cafe babe 0003 002d 0012 0700 0e07 0010   .............. 
0010 0a00 0200 040c 0007 0005 0100 0328 2956   .............()V
0020 0100 1628 5b4c 6a61 7661 2f6c 616e 672f   ...([Ljava/lang/
0030 5374 7269 6e67 3b29 5601 0006 3c69 6e69   String;)V...<ini
0040 743e 0100 0443 6f64 6501 000d 436f 6e73   t>...Code...Cons
0050 7461 6e74 5661 6c75 6501 000a 4578 6365   tantValue...Exce
0060 7074 696f 6e73 0100 0f4c 696e 654e 756d   ptions...LineNum
0070 6265 7254 6162 6c65 0100 0e4c 6f63 616c   berTable...Local
0080 5661 7269 6162 6c65 7301 000a 536f 7572   Variables...Sour
0090 6365 4669 6c65 0100 0361 6464 0100 0861   ceFile...add...a
00a0 6464 2e6a 6176 6101 0010 6a61 7661 2f6c   dd.java...java/l
00b0 616e 672f 4f62 6a65 6374 0100 046d 6169   ang/Object...mai
00c0 6e00 2100 0100 0200 0000 0000 0200 0900   n.............. 
00d0 1100 0600 0100 0800 0000 2d00 0200 0400   ..............  
00e0 0000 0d10 0f3c 1009 3d03 3e1b 1c60 3eb1   .............  
00f0 0000 0001 000b 0000 000e 0003 0000 0004   ..............  
0100 0008 0006 000c 0002 0001 0007 0005 0001   ..............  
0110 0008 0000 001d 0001 0001 0000 0005 2ab7   ..............  
0120 0003 b100 0000 0100 0b00 0000 0600 0100   ..............  
0130 0000 0100 0100 0d00 0000 0200 0f00        ..............  
```

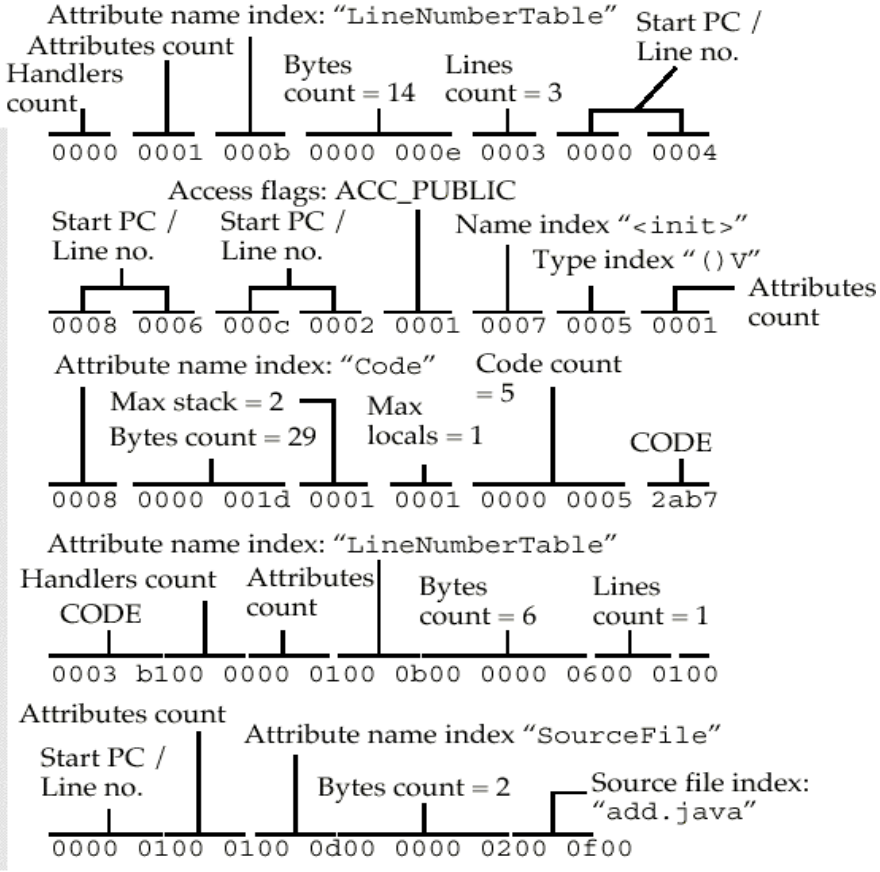
A Java Class File





A Java Class File (Cont'd)





A Java Class File (Cont'd)

```
ClassFile {
        u4 magic;
        u2 minor_version;
        u2 major_version;
        u2 constant_pool_count;
        cp_info constant_pool[constant_pool_count-1];
        u2 access_flags;
        u2 this_class;
        u2 super_class;
        u2 interfaces_count;
        u2 interfaces[interfaces_count];
        u2 fields_count;
        field_info fields[fields_count];
        u2 methods_count;
        method_info methods[methods_count];
        u2 attributes_count;
        attribute_info attributes[attributes_count];
    }
```

Byte Code for Java Program

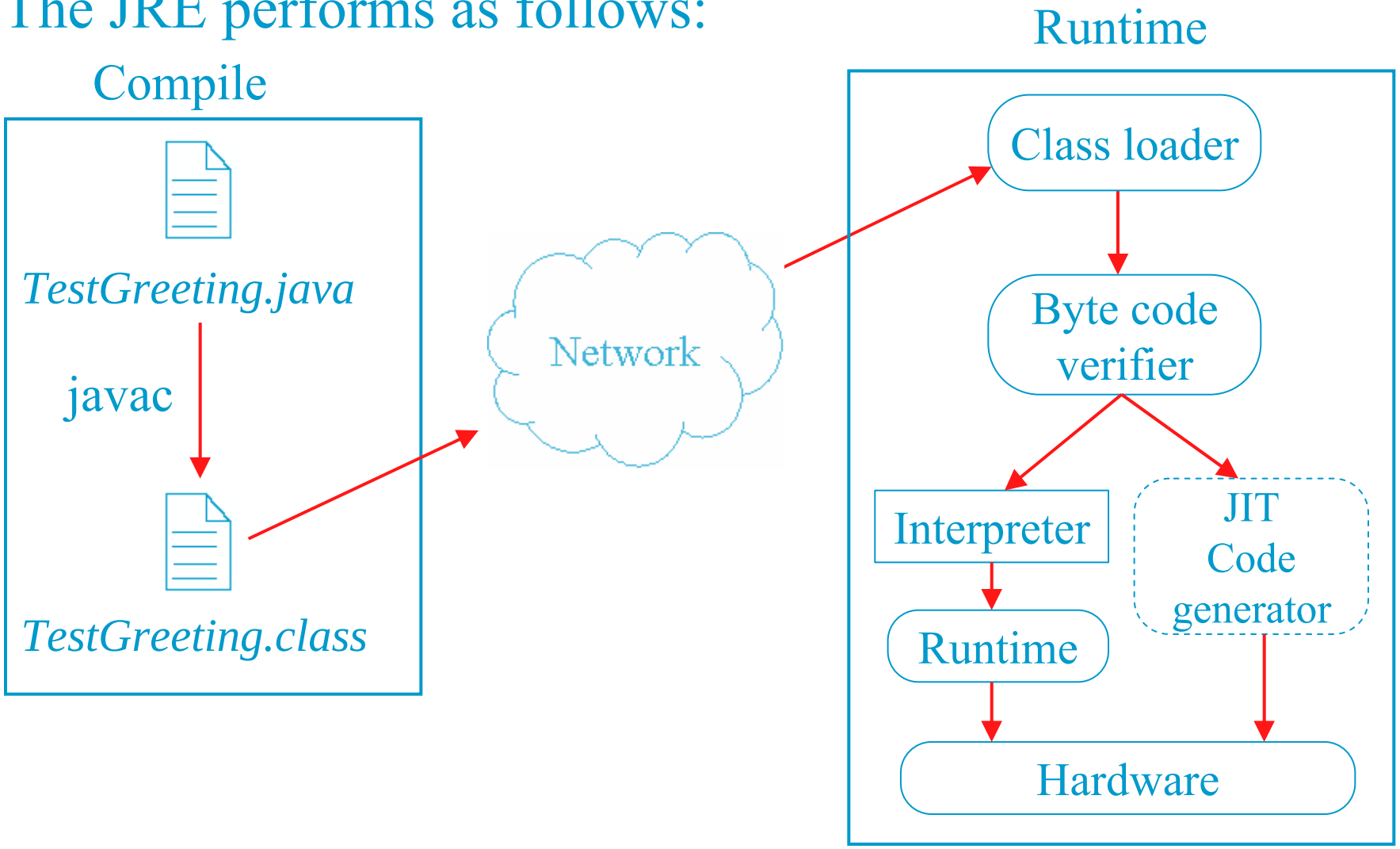
Location	Code	Mnemonic	Meaning
0x00e3	0x10	bipush	Push next byte onto stack
0x00e4	0x0f	15	Argument to bipush
0x00e5	0x3c	istore_1	Pop stack to local variable 1
0x00e6	0x10	bipush	Push next byte onto stack
0x00e7	0x09	9	Argument to bipush
0x00e8	0x3d	istore_2	Pop stack to local variable 2
0x00e9	0x03	iconst_0	Push 0 onto stack
0x00ea	0x3e	istore_3	Pop stack to local variable 3
0x00eb	0x1b	iload_1	Push local variable 1 onto stack
0x00ec	0x1c	iload_2	Push local variable 2 onto stack
0x00ed	0x60	iadd	Add top two stack elements
0x00ee	0x3e	istore_3	Pop stack to local variable 3
0x00ef	0xb1	return	Return

Garbage Collection

- Allocated memory that is no longer needed should be deallocated
- In other languages, deallocation is the programmer's responsibility
- The Java programming language provides a system-level thread to track memory allocation
- Garbage collection
  - Checks for and frees memory no longer needed
  - Is done automatically
  - Can vary dramatically across JVM implementations

Java Runtime Environment

The JRE performs as follows:



The Java Runtime Environment

- Performs three main tasks:
  - Loads code – Performed by the class loader
  - Verifies code – Performed by the bytecode verifier
  - Executes code – Performed by the runtime interpreter

The Class Loader

- Loads all classes necessary for the execution of a program
- Maintains classes of the local file system in separate "namespaces"
- Prevents spoofing

The Bytecode Verifier

- Ensures that
  - The code adheres to the JVM specification
  - The code does not violate system integrity
  - The code causes no operand stack overflows or underflows
  - The parameter types for all operational code are correct
  - No illegal data conversions (the conversion of integers to pointers) have occurred

A Basic Java Application: TestGreeting.java

```
1 //
2 // Sample "Hello World" application
3 //
4 public class TestGreeting{
5     public static void main (String[] args) {
6             Greeting hello = new Greeting();
7             hello.greet();
8     }
9 }
```

Greeting.java

```
1 // The Greeting class declaration.
2 public class Greeting {
3     public void greet() {
4         System.out.println("hi");
5     }
6 }
```

Running and Compiling

- Compiling TestGreeting.java
  - javac TestGreeting.java
- Greeting.java is compiled automatically
- Running an application
  - java TestGreeting
- Locating common compile and runtime errors

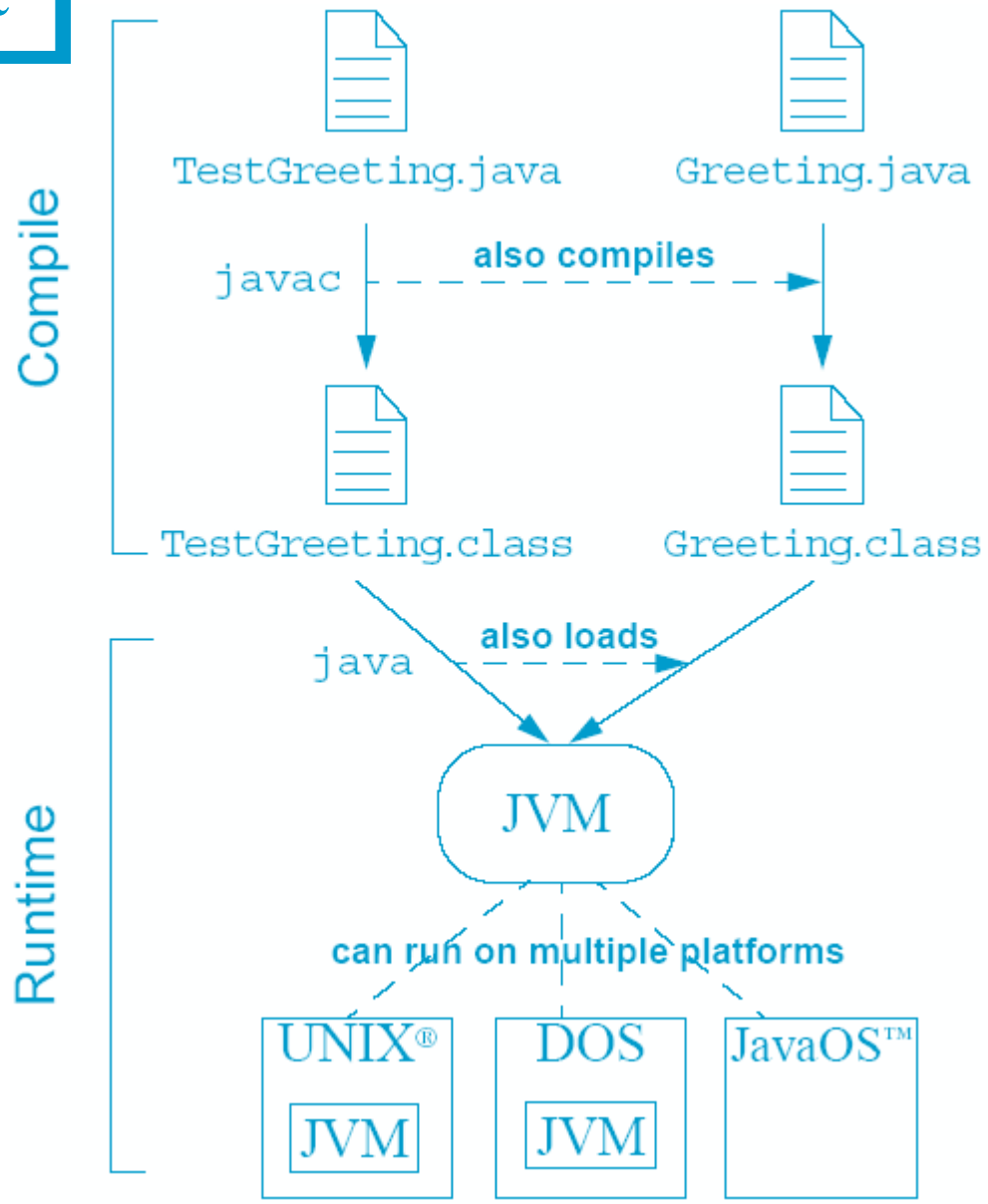
Compile-Time Errors

- `javac: Command not found`
- `Greeting.java:4: cannot resolve symbol`
  `symbol : method printl (java.lang.String)`
  `location: class java.io.PrintStream`
  `System.out.printl("hi");`
- `TestGreet.java:4: Public class TestGreeting`
  `must be defined in a file called`
  `"TestGreeting.java".`

Run-Time Errors

- Can’t find class TestGreeting
- Exception in thread "main"
  java.lang.NoSuchMethodError: main

Java Runtime Environment



1# JAVA TECHNOLOGY

1. Compile Test1.java
2. Compile Test2.java
3. Compile Test3.java
4. Compile Test4.java

Sun microsystems

HANDS-ON LAB