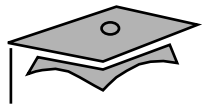


# Module 1

## Getting Started



## Objectives

- Describe the key features of Java technology
- Write, compile, and run a simple Java technology application
- Describe the function of the Java Virtual Machine (JVM™)
- Define garbage collection
- List the three tasks performed by the Java platform that handle code security

NOTE: The terms “Java Virtual Machine” and “JVM” mean a Virtual Machine for the Java™ platform.



## Relevance

- Is the Java programming language a complete language or is it useful only for writing programs for the Web?
- Why do you need another programming language?
- How does the Java technology platform improve on other language platforms?



# What Is the Java™ Technology?

- Java technology is:
  - A programming language
  - A development environment
  - An application environment
  - A deployment environment
- It is similar in syntax to C++.
- It is used for developing both *applets* and *applications*.



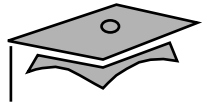
# Primary Goals of the Java Technology

- Provides an easy-to-use language by:
  - Avoiding many 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
- Loads classes dynamically; that is, at the time they are actually needed
- Supports changing programs dynamically during runtime by loading classes from disparate sources
- Furnishes better security



# Primary Goals of the Java Technology

The following features fulfill these goals:

- The Java Virtual Machine (JVM™)<sup>1</sup>
- Garbage collection
- The Java Runtime Environment (JRE)
- JVM tool interface

---

1. The terms "Java Virtual Machine" and "JVM" mean a Virtual Machine for the Java platform



# 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
- Fatal error reporting
- High-precision timing support



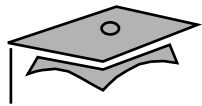
# The Java Virtual Machine

- The majority of type checking is done when the code is compiled.
- Implementation of the JVM approved by Sun Microsystems must be able to run any compliant class file.
- The JVM executes on multiple operating environments.



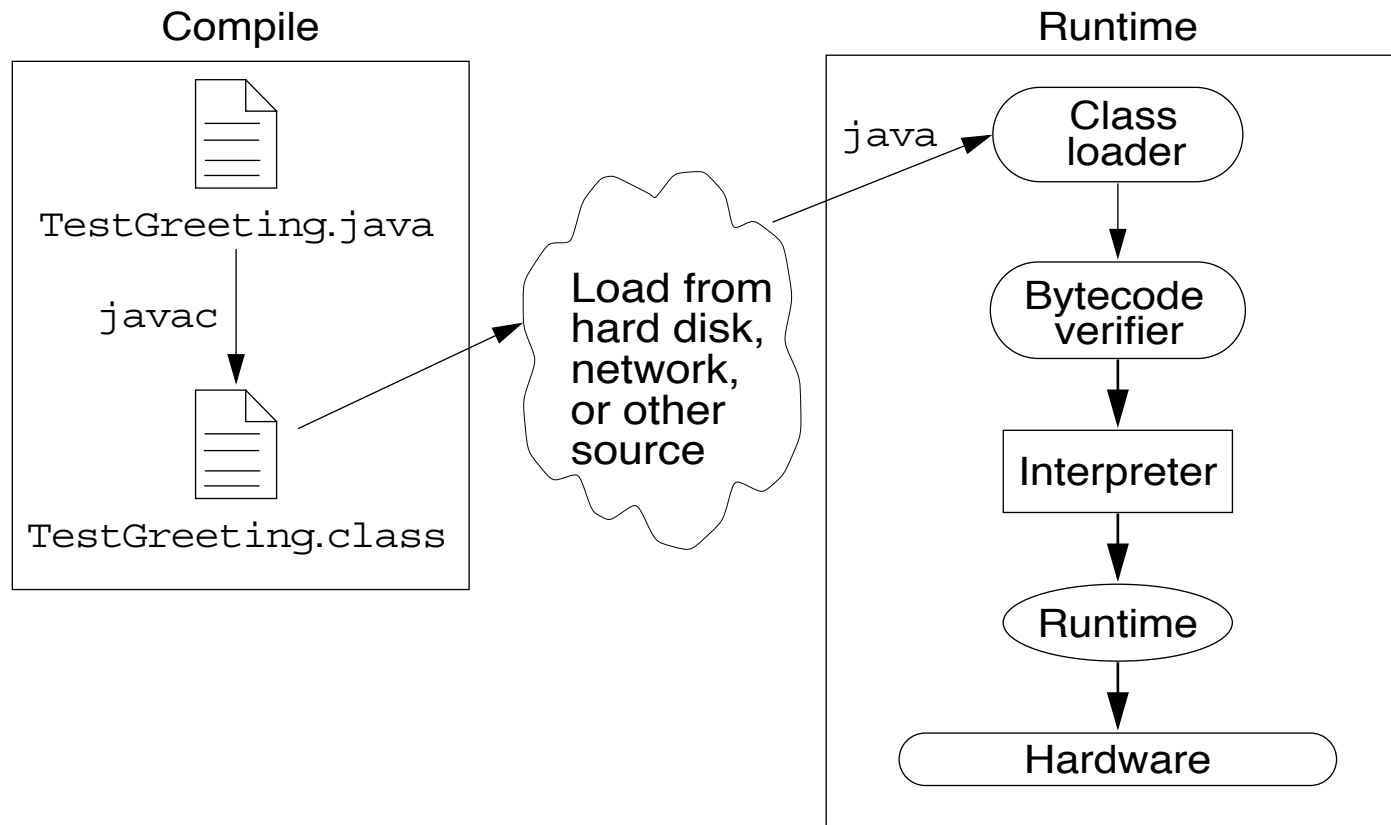
## Garbage Collection

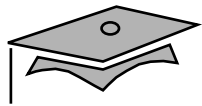
- Allocated memory that is no longer needed should be deallocated.
- In other languages, deallocation is the programmer's responsibility.
- The Java programming language provides a system-level thread to track memory allocation.
- Garbage collection has the following characteristics:
  - Checks for and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



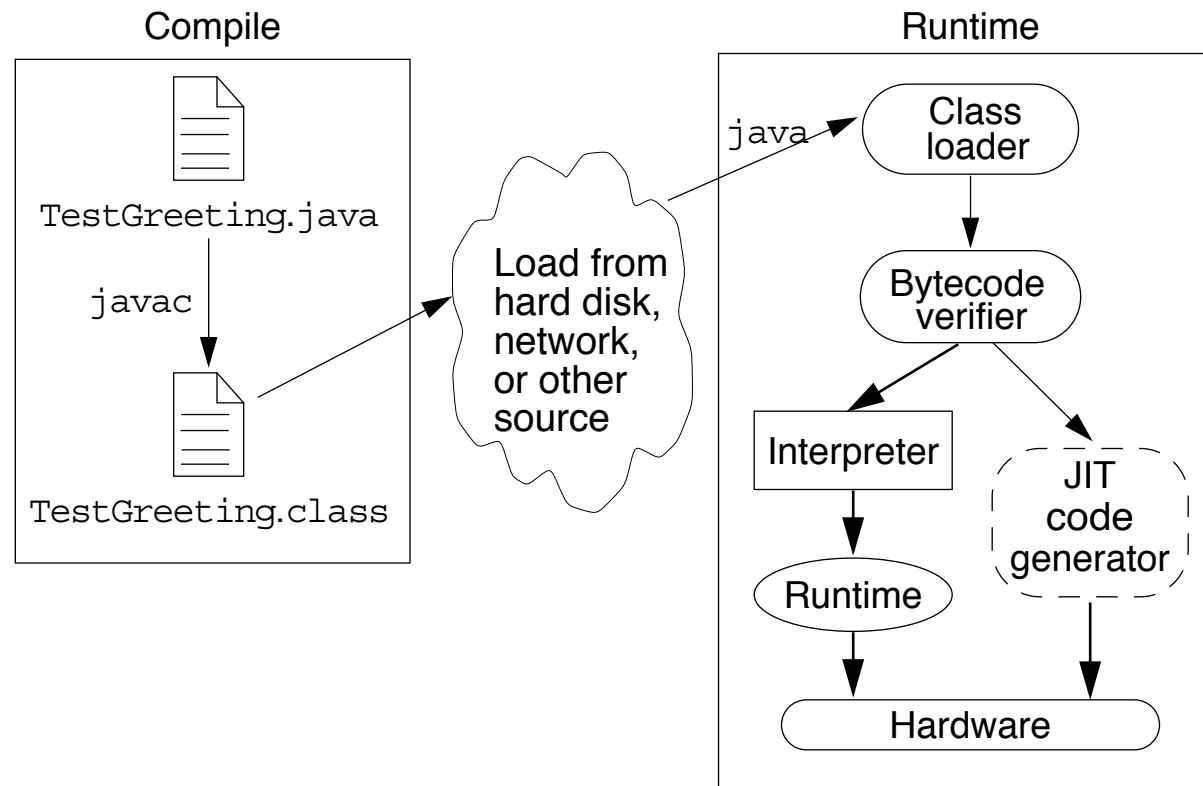
# The Java Runtime Environment

The Java application environment performs as follows:





# Operation of the JRE With a Just-In-Time (JIT) Compiler





## JVM™ Tasks

The JVM performs three main tasks:

- Loads code
- Verifies code
- Executes code



## 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

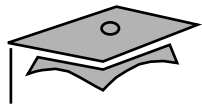


# The Bytecode Verifier

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 Simple Java Application

## The TestGreeting.java Application

```
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  }
```

## The Greeting.java Class

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



# The TestGreeting Application

- Comment lines
- Class declaration
- The `main` method
- Method body



# The Greeting Class

- Class declaration
- The greet method



# Compiling and Running the TestGreeting Program

- Compile TestGreeting.java:  
`javac TestGreeting.java`
- The Greeting.java is compiled automatically.
- Run the application by using the following command:  
`java TestGreeting`
- Locate common compile and runtime errors.



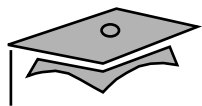
## Compile-Time Errors

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



## Runtime Errors

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



# Java Technology Runtime Environment

