



Chapters 1

OOP : Basic Concepts

Adapted from: Java™ How to Program, 9/e

Taif University
College of Computers and Information Technology



1.1 Introduction

- ▶ A programming language specifies the words and symbols that we can use to write a program
- ▶ A programming language employs a set of rules that dictate how the words and symbols can be put together to form valid program statements
- ▶ The Java programming language was created by Sun Microsystems, Inc.



2- Program Development

- ▶ The mechanics of developing a program include several activities
 - writing the program in a specific programming language (in our case Java)
 - translating the program into a form that the computer can execute
 - investigating and fixing various types of errors that can occur
- ▶ Software tools can be used to help with all parts of this process



3- Programming Languages

- ▶ Each type of CPU executes only a particular machine language
- ▶ A program must be translated into machine language before it can be executed
- ▶ A compiler is a software tool which translates source code into a specific target language
- ▶ Often, that target language is the machine language for a particular CPU type
- ▶ The Java approach is somewhat different



4- Java Translation

- ▶ The Java compiler translates Java source code into a special representation called *bytecode*
- ▶ Java bytecode is not the machine language for any traditional CPU
- ▶ Another software tool, called an *interpreter*, translates bytecode into machine language and executes it
- ▶ Therefore the Java compiler is not tied to any particular machine

5- Development Environments

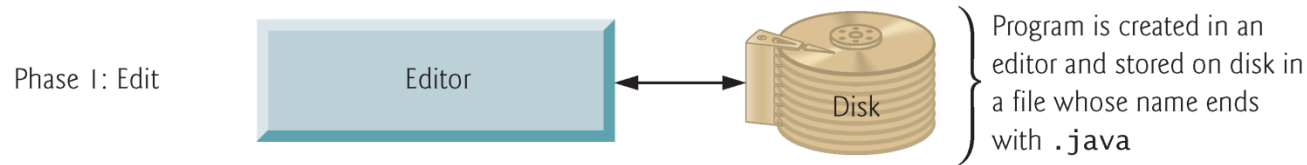


Fig. 1.6 | Typical Java development environment—editing phase.

5- Development Environments (cont.)

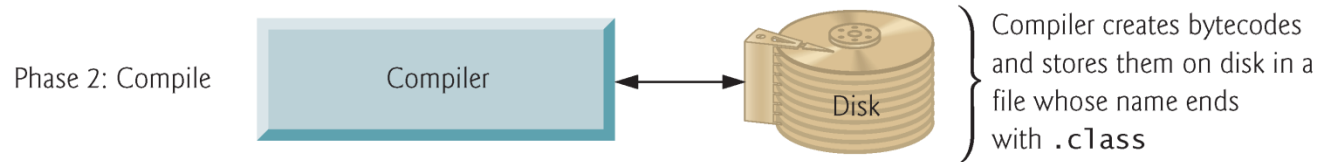


Fig. 1.7 | Typical Java development environment—compilation phase.

5- Development Environments (cont.)

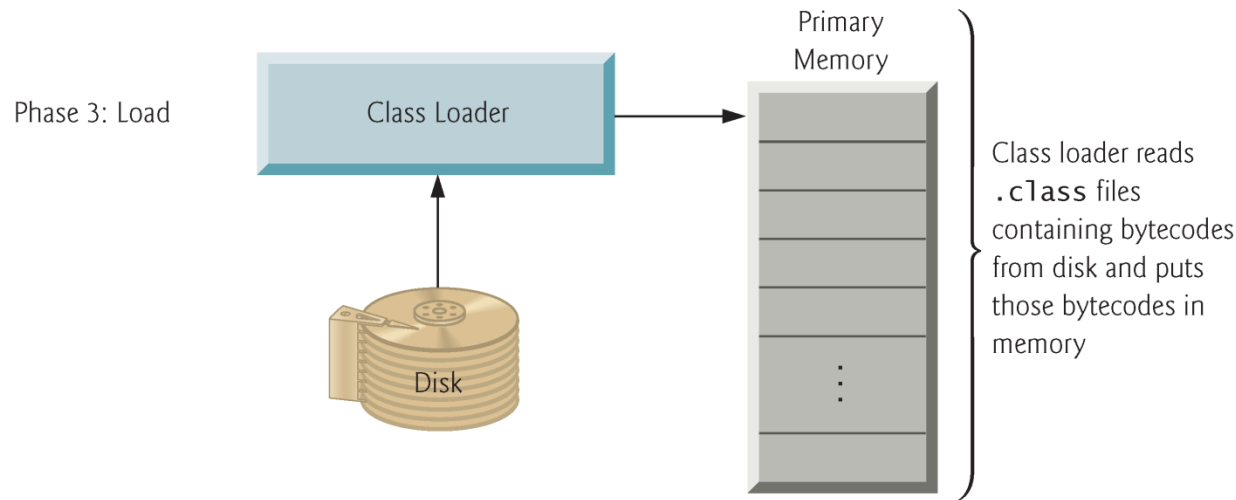


Fig. 1.8 | Typical Java development environment—loading phase.

5- Development Environments (cont.)

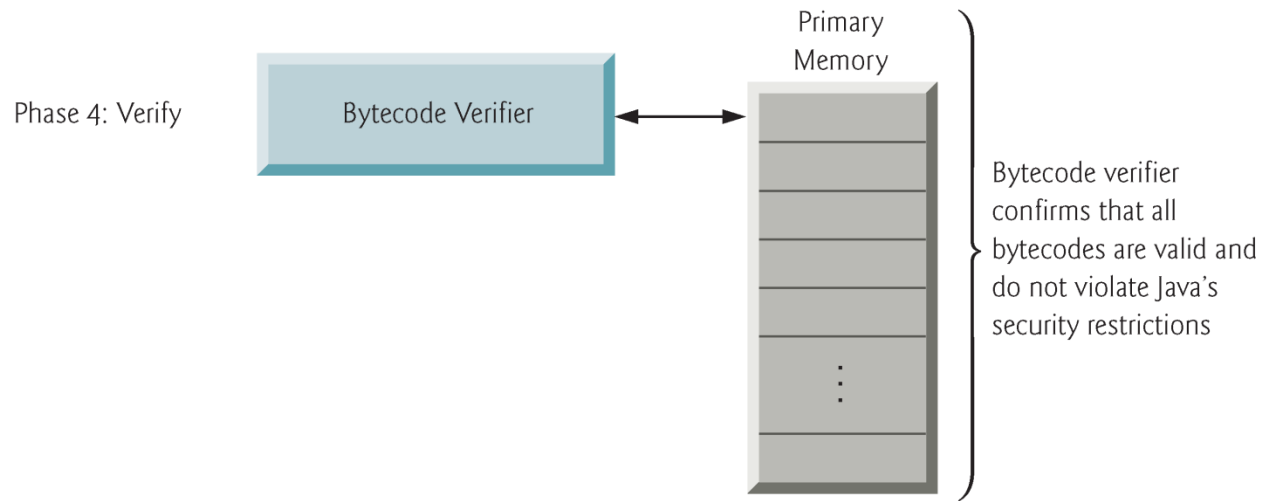


Fig. 1.9 | Typical Java development environment—verification phase.

5- Development Environments (cont.)

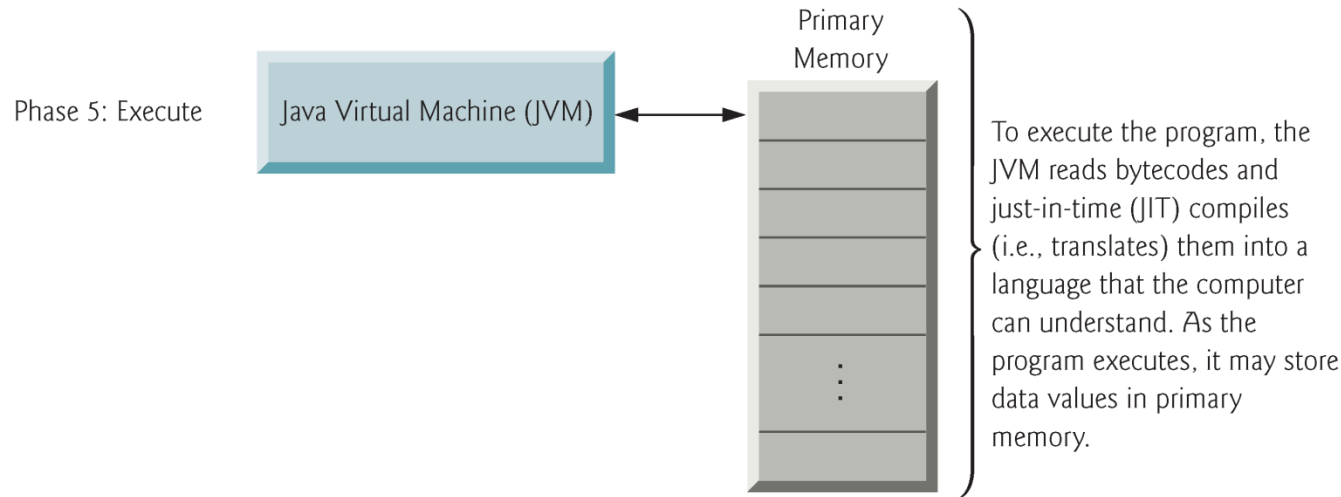


Fig. 1.10 | Typical Java development environment—execution phase.

5- Development Environments (cont.)



- ▶ There are many **Integrated development environments (IDEs)** that support the software development process, including editors for writing and editing programs and debuggers for locating **logic errors**—errors that cause programs to execute incorrectly.
- Sun NetBeans
- **Eclipse (www.eclipse.org)**
- NetBeans (www.netbeans.org).
- Borland JBuilder

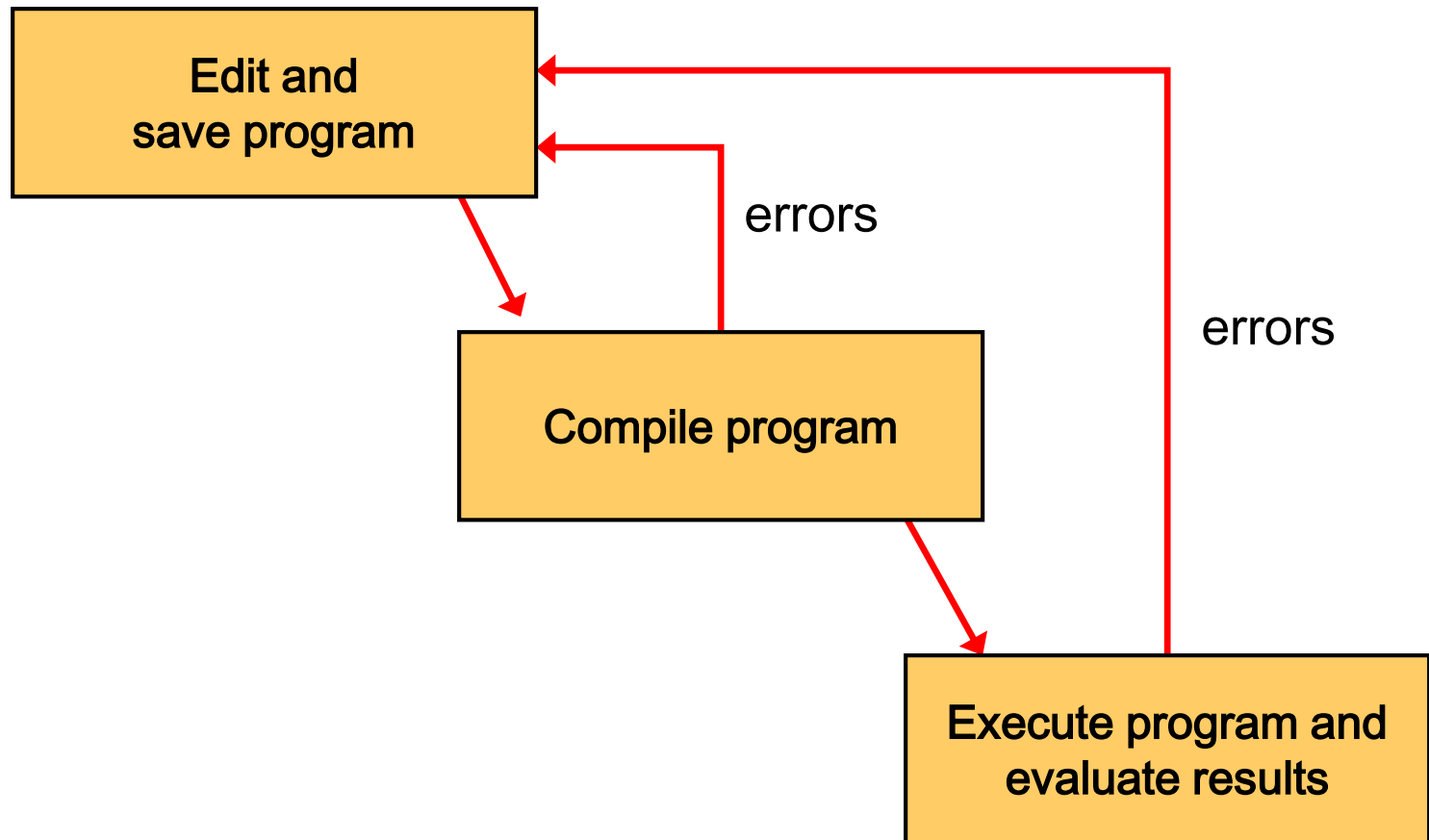
6-Errors



- ▶ A program can have three types of errors
- ▶ The compiler will find syntax errors and other basic problems (compile-time errors)
 - If compile-time errors exist, an executable version of the program is not created
- ▶ A problem can occur during program execution, such as trying to divide by zero, which causes a program to produce incorrect results or to terminate abnormally (run-time errors)



7-Basic Program Development





9-Problem Solving

- ▶ The purpose of writing a program is to solve a problem
- ▶ Solving a problem consists of multiple activities:
 - Understand the problem
 - Design a solution
 - Consider alternatives and refine the solution
 - Implement the solution
 - Test the solution



10- Object-Oriented Programming

- ▶ Java is an object-oriented programming language
- ▶ As the term implies, an object is a fundamental entity in a Java program
- ▶ Objects can be used effectively to represent real-world entities



10- Object-Oriented Programming (cont.)

- ▶ An **object** has:
 - state - descriptive characteristics
 - behaviors - what it can do (or what can be done to it)
- ▶ The state of a person includes his id, firstName, lastName, age, phoneNumber, ...
- ▶ The behaviors associated with a person provide the ability to ask for person's firstName, lastName, age or phoneNumber
- ▶ Note that the behavior of an object might change its state

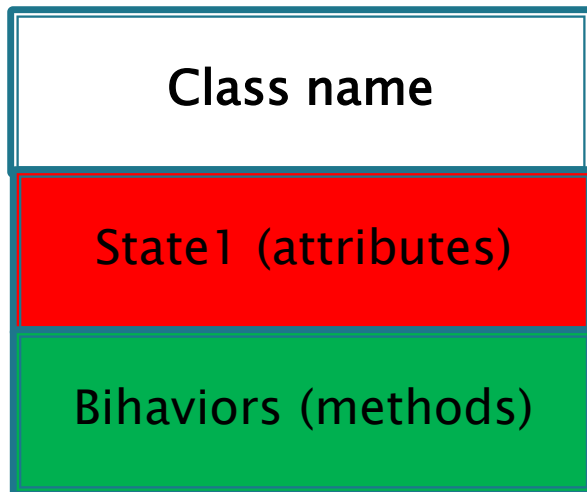


10- Object-Oriented Programming (cont.)

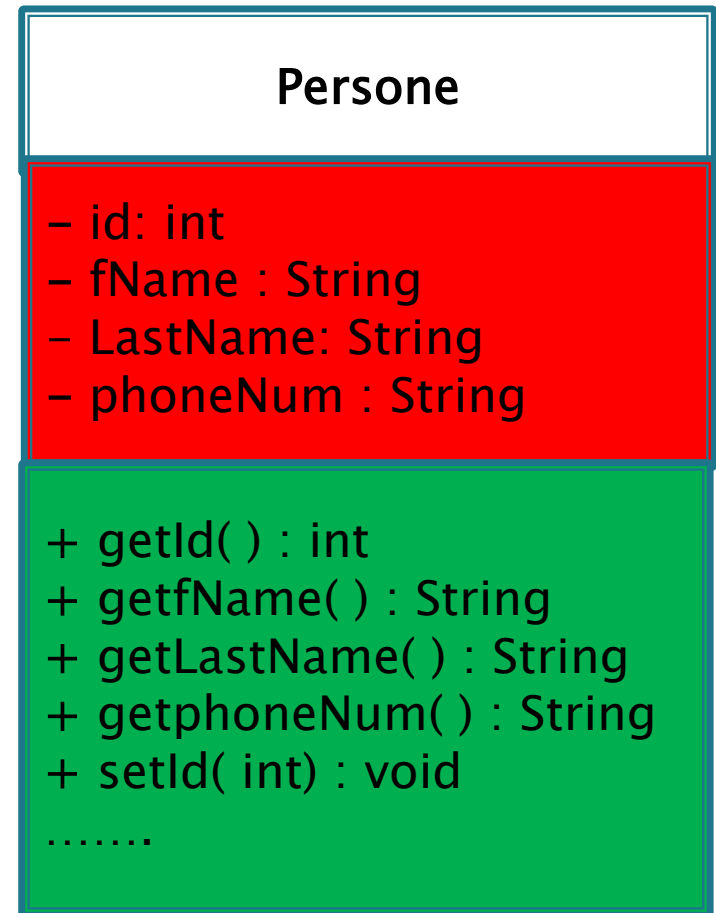
- ▶ An **object** is defined by a **class**
- ▶ A **class** is the blueprint of an object
- ▶ The **class** uses methods to define the behaviors of the object
- ▶ The **class** that contains the main method of a Java program represents the entire program
- ▶ A **class** represents a concept, and an object represents the embodiment of that concept
- ▶ Multiple **objects** can be created from the same **class**

10- Object-Oriented Programming (cont.)

► **Class** ULM presentation



- → private
+ → public





10- Object-Oriented Programming (cont.)

► Instantiation

- Just as someone has to build a car from its engineering drawings before you can actually drive a car, you must build an object of a class before a program can perform the tasks that the class's methods define.
- An object is then referred to as an **instance** of its class.



10- Object-Oriented Programming (cont.)

► Methods and Classes

- Performing a task in a program requires a **method**.
- The method houses the program statements that actually perform its tasks.
- In Java, we create a program unit called a **class** to house the set of methods that perform the class's tasks.



10- Object-Oriented Programming (cont.)

► Messages and Methods Calls

- When you ask for person's firstName sends a *message* to the person to perform a task—that is, **getFirstName()**.
- Similarly, you send messages to an object.
- Each message is implemented as a **method call** that tells a method of the object to perform its task.



10- Object-Oriented Programming (cont.)

► Attributes and Instance Variables

- A person has *attributes*, like *a* firstName, lastName, id, phoneNumber
- The person's attributes are represented as part of its design in its engineering diagrams.
- Every person maintains its *own attributes*.



10- Object-Oriented Programming (cont.)

- An **object**, has attributes that it carries along as it's used in a program.
- Specified as part of the object's class.
- A bank account object has a *balance* attribute that represents the amount of money in the account.
- Each bank account object knows the balance in the account it represents, but *not* the balances of the other accounts in the bank.
- Attributes are specified by the class's **instance variables**.



10- Object-Oriented Programming (cont.)

► Encapsulation

- Classes **encapsulate** (i.e., wrap) attributes and methods into objects—an object's attributes and methods are intimately related.
- Objects may communicate with one another, but they're normally not allowed to know how other objects are implemented—implementation details are *hidden* within the objects themselves.
- **Information hiding**, as we'll see, is crucial to good software engineering.



10- Object-Oriented Programming (cont.)

► Inheritance

- A new class of objects can be created quickly and conveniently by **inheritance**—the new class absorbs the characteristics of an existing class, possibly customizing them and adding unique characteristics of its own.



10- Object-Oriented Programming (cont.)

► Polymorphism

- Enables you to “program in the general” rather than “program in the specific.”
- Polymorphism enables you to write programs that process objects that share the same superclass as if they’re all objects of the superclass; this can simplify programming.