Chapter 1
Introduction to Java

OBJECTIVES

In this chapter you will learn:

- History of Java
- Features of Java
- Three edition for Java SDK
- A typical Java development environment
- To test-drive a Java application using the command line

1.1 History of Java

- 1991
  - Sun Microsystems funded an internal corporate research project led by James Gosling
- 2009
  - Sun Microsystems was acquired by Oracle

Java SE versions history

Version Name	Code Name	Release Date
JDK 1.0	Oak	January 1996
JDK 1.1	(none)	February 1997
J2SE 1.2	Playground	December 1998
J2SE 1.3	Kestrel	May 2000
J2SE 1.4	Merlin	February 2002
J2SE 5.0	Tiger	September 2004
JDK 6	Mustang	December 2006
JDK 7	Dolphin	July 2011
JDK 8		March 2014
JDK 9		September, 21st 2017
JDK 10		March, 20th 2018
JDK 11		September, 25th 2018
JDK 12		March, 19th 2019
JDK 13		September, 10th 2019

1.2 Features of Java

- **simple**
- **object-oriented**
- **distributed**
- **Interpreted(解析性的）**
- **Robust(强壮的）**
  - compile-time error checking/ not support memory pointers / automatic garbage collection

1.2 Features of Java (Cont.)

- secure
- architecture neutral(体系结构中立）
- portable
- high performance
- multithreaded
- dynamic language

1.3 Three edition for Java SDK

- Java SDK Standard Edition（J2SE)
  - contains the basic core Java classes.
- Java SDK Micro Edition (J2ME)
  - for developers that code to portable devices, such as a palm pilot or a cellular phone.
- Java SDK Enterprise Edition (J2EE)
  - contains classes that go above and beyond J2SE

1.4 Introduction to Object-Oriented Technology

- Object
- Methods and classes
- Instance
- Reuse
- Messages and methods calls
- Attributes and instance variables
- Encapsulation
- Inheritance
- Polymorphism

1.5 Java and a Typical Java Development Environment (Cont.)

- Java Class Libraries
  - Rich collections of existing classes and methods
  - Also known as the Java APIs (Application Programming Interfaces).

Performance Tip 1.1

Using Java API classes and methods instead of writing your own versions can improve program performance, because they're carefully written to perform efficiently. This also shortens program development time.

Portability Tip 1.1

Although it's easier to write portable programs (i.e., programs that can run on many different types of computers) in Java than in most other programming languages, differences between compilers, JVMs and computers can make portability difficult to achieve. Simply writing programs in Java does not guarantee portability.

1.5 Java and a Typical Java Development Environment (Cont.)

- Java programs normally go through five phases
  - edit
  - compile
  - load
  - verify
  - execute

1.5 Java and a Typical Java Development Environment (Cont.)

- Phase 1 consists of editing a file with an editor program
  - Type a Java program (source code) using the editor.

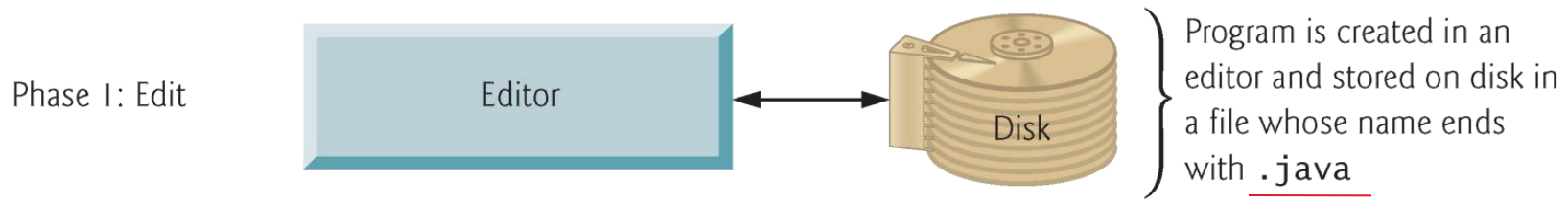

Fig. 1.6 | Typical Java development environment—editing phase.

1.5 Java and a Typical Java Development Environment (Cont.)

- Linux editors: `vi` and `emacs`.
- Windows editors:
  - Notepad
  - EditPlus (`www.editplus.com`)
  - TextPad (`www.textpad.com`)
  - jEdit (`www.jedit.org`).

1.5 Java and a Typical Java Development Environment (Cont.)

- Popular Integrated development environments（IDEs）
  - Eclipse (`www.eclipse.org`)
  - NetBeans (`www.netbeans.org`).
  - jGRASP™ IDE (`www.jgrasp.org`)
  - DrJava IDE (`www.drjava.org/download.shtml`)
  - BlueJ IDE (`www.bluej.org/`)
  - TextPad® Text Editor for Windows® (`www.textpad.com/`)

1.5 Java and a Typical Java Development Environment (Cont.)

- Phase 2: Compiling a Java Program into Bytecodes
  - Use the command javac (the Java compiler) to compile a program
    - `javac Welcome.java`

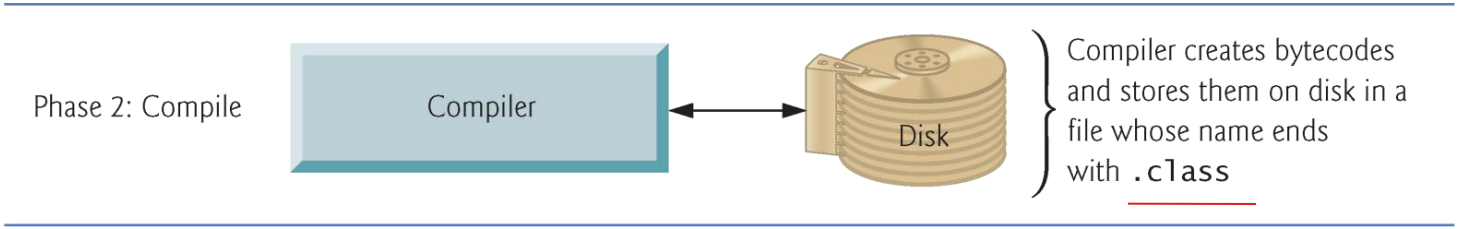

Fig. 1.7 | Typical Java development environment—compilation phase.

1.5 Java and a Typical Java Development Environment (Cont.)

- The JVM is invoked by the java command.
  - `java Welcome`

1.5 Java and a Typical Java Development Environment (Cont.)

- Phase 3: Loading a Program into Memory

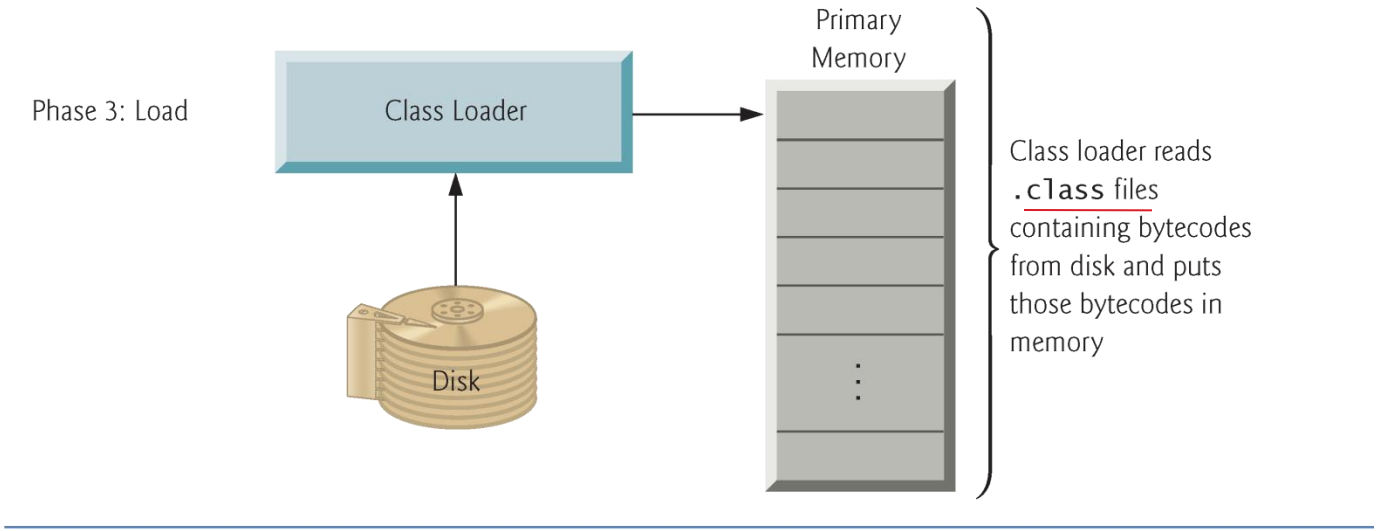

Fig. 1.8 | Typical Java development environment—loading phase.

1.5 Java and a Typical Java Development Environment (Cont.)

- Phase 4: Bytecode Verification
  - As the classes are loaded, the bytecode verifier examines their bytecodes
  - Ensures that they're valid and do not violate Java's security restrictions.

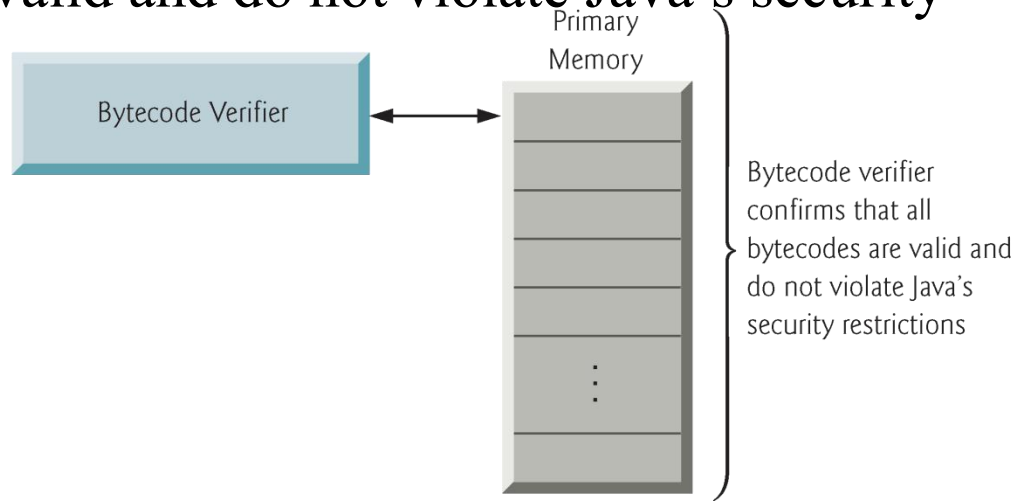

Fig. 1.9 Typical Java development environment—verification phase.

1.5 Java and a Typical Java Development Environment (Cont.)

- Phase 5: Execution
  - The JVM executes the program's bytecodes.
  - JVMs typically execute bytecodes using a combination of interpretation(解释) and so-called just-in-time (JIT) compilation.
  - Analyzes the bytecodes as they're interpreted
  - A just-in-time (JIT) compiler—known as the Java HotSpot compiler [illegible]tes the bytecodes into the underlying computer [illegible]he language.

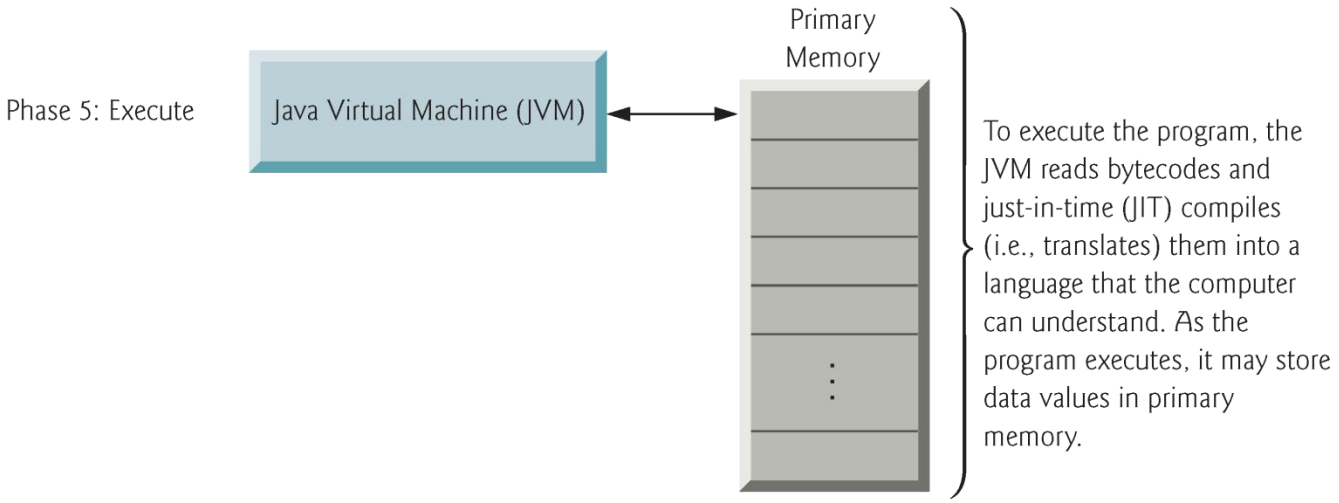

Fig. 1.10 | Typical Java development environment—execution phase.

1.6 Test-Driving a Java Application

- ***Checking your setup.*** Read the Before You Begin section of the book to confirm that you've set up Java properly on your computer and that you've copied the book's examples to your hard drive.

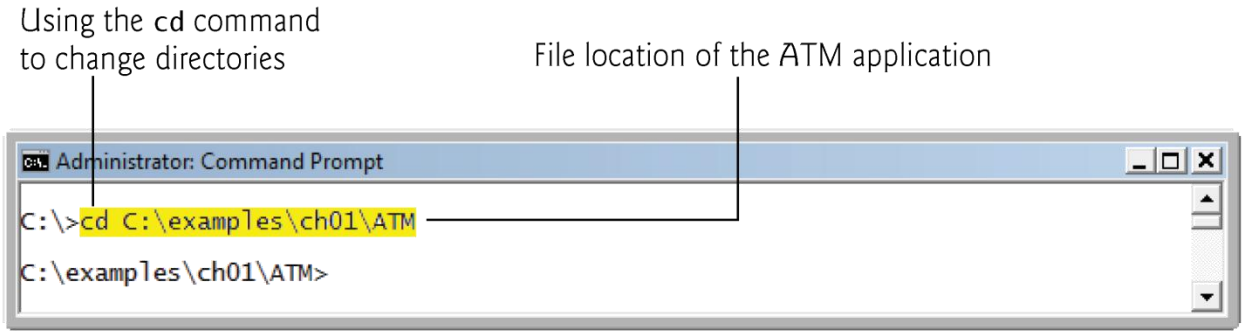

Fig. 1.11 | Opening a **Command Prompt** and changing directories.

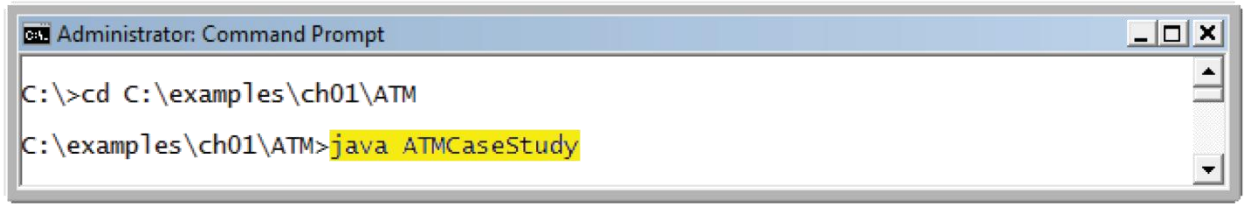

Fig. 1.12 | Using the `java` command to execute the ATM application.

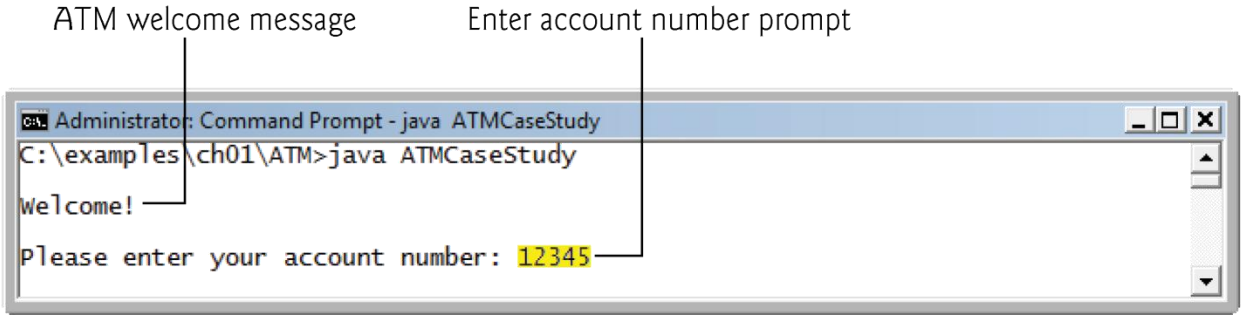

Fig. 1.13 | Prompting the user for an account number.

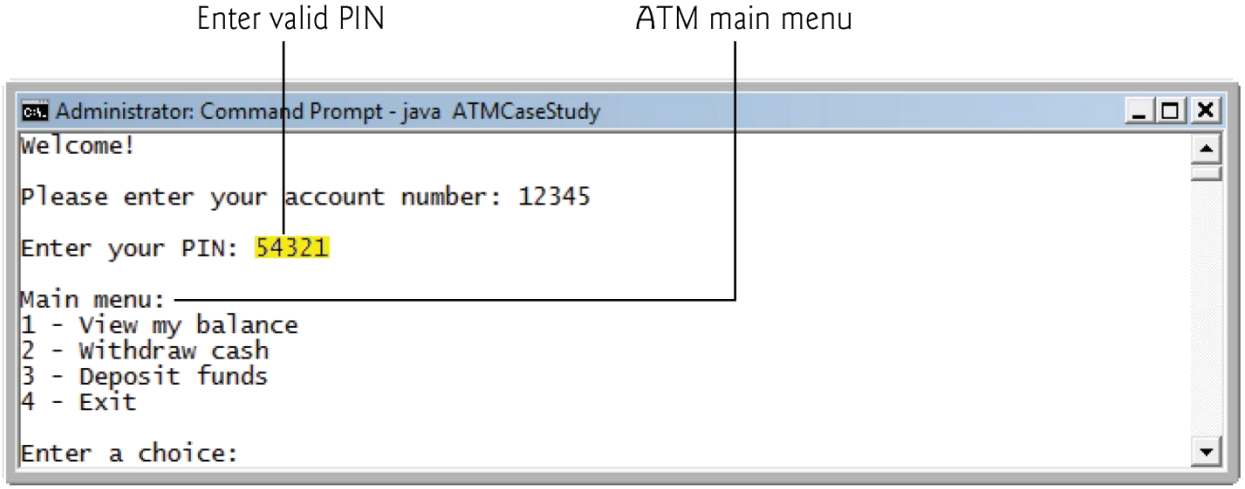

Fig. 1.14 | Entering a valid PIN number and displaying the ATM application's main menu.

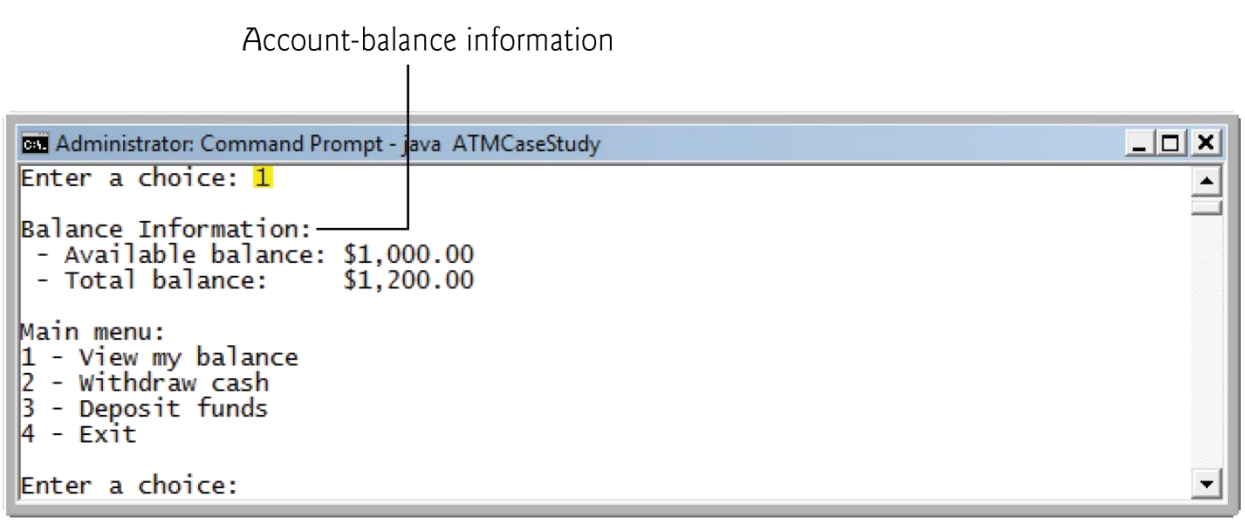

Fig. 1.15 | ATM application displaying user account-balance information.

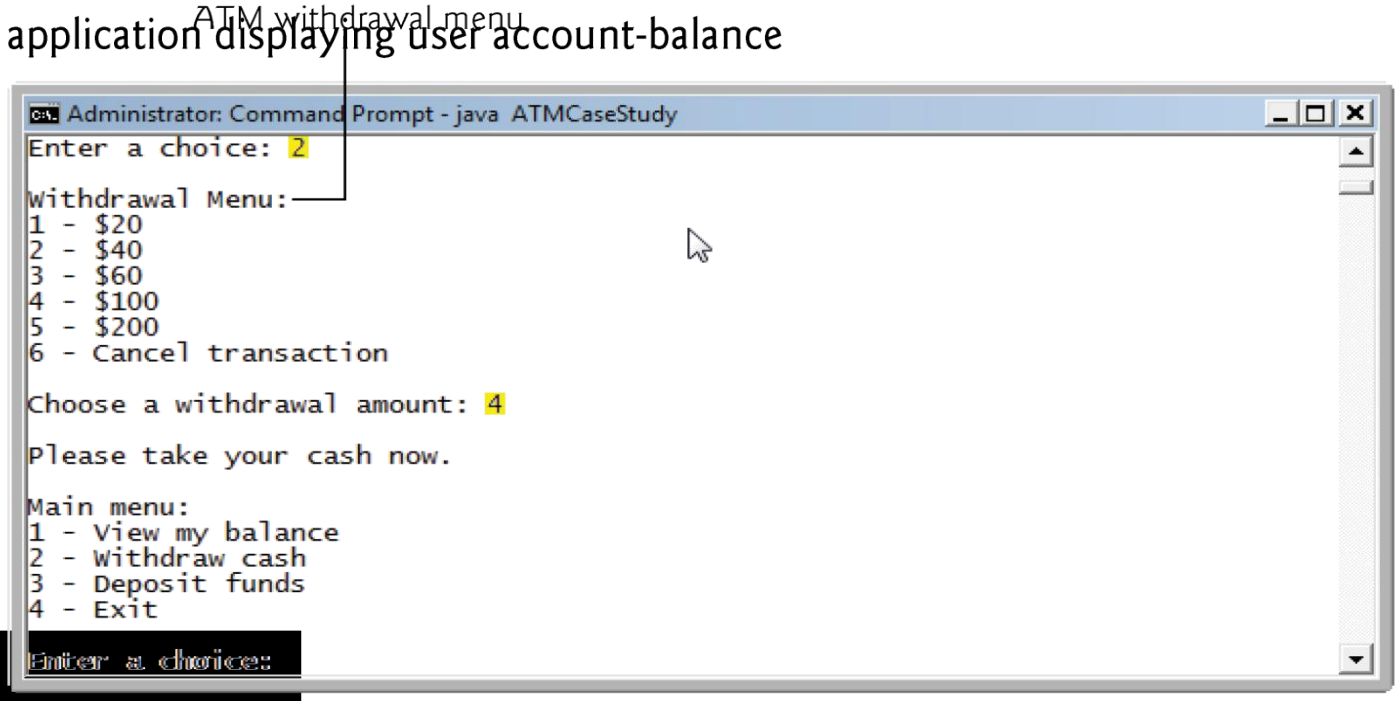

Fig. 1.16 | rawing money from the account and returning to the main men

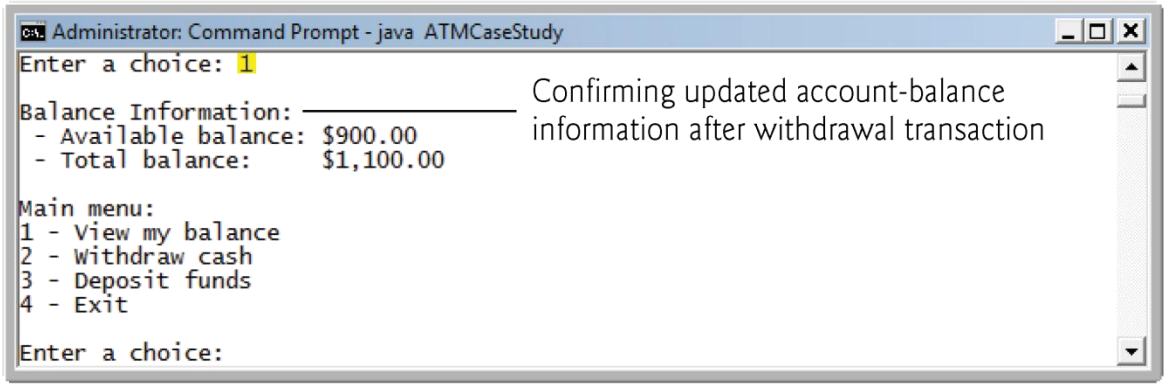

Fig. 1.17 | Checking the new balance.

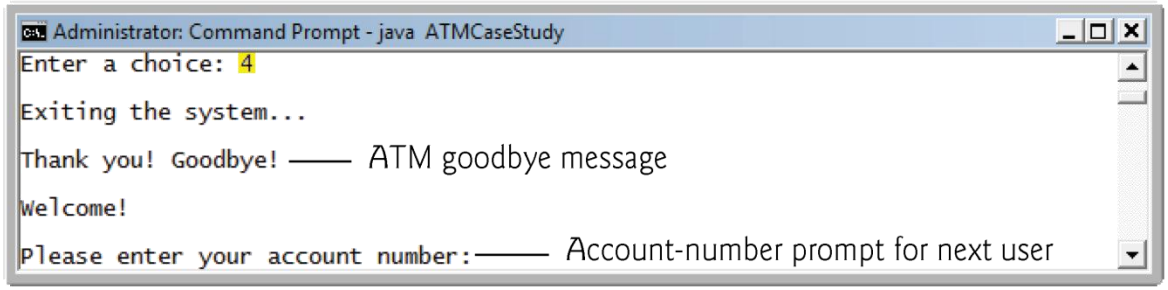

Fig. 1.18 | Ending an ATM transaction session.