Chapter 2
Introduction to Java Applications

Java™ How to Program, 11th Edition

Instructor: Zhuozhao Li

Outline

© 1992-2010 by Pearson Education, Inc. All Rights Reserved.

2.1 Introduction

- Java application
  - A computer program that executes when you use the `java` command to launch the JVM.

- Use tools from the JDK to compile and run programs

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.2 Our First Program in Java: Printing a Line of Text

```
// The first Java program
// Text-printing program


public class Welcome1
{

   // main method begins execution of Java application
   public static void main ( String[] args )
   {
      System.out.println( "Welcome to Java Programming!" );
   } //end method main

} // end class Welcome1
```

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.2 Our First Program in Java: Printing a Line of Text (Cont.)

- Comments

```
// The first Java program
```

  - // indicates that the line is a comment.
  - Used to document programs and improve their readability （可读性）.
  - Compiler ignores comments.

- Traditional comment, can be spread over several lines

```
/* This is a traditional comment. It
   can be split over multiple lines */
```

  - This type of comment begins with /* and ends with */

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

Traditional vs. End-of-Line Comments

- Traditional comments do not nest （嵌套）, the first */ after the first /* will terminate the comment

```
/*
  /* comment 1 */
     comment 2 */
```

Syntax Error

- End-of-line comments can contain anything

```
// /* this comment is okay */
```

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.2 Our First Program in Java: Printing a Line of Text (Cont.)

Class (类) declaration

```
public class Welcome1
```

- Every Java program consists of at least one class that you define
- class keyword （关键字、保留字） introduces a class declaration and is immediately followed by the class name
- Keywords are reserved for use by Java and are always spelled with all lowercase letters.

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.2 Our First Program in Java: Printing a Line of Text (Cont.)

Class names

- By convention, begin with a capital letter and capitalize the first letter of each word they include
- Name cannot start with a digit
- Java is case sensitive （区分大小写）—uppercase and lowercase letters are distinct (not in comments)
  - Welcome1 and welcome1 are different names

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.2 Our First Program in Java: Printing a Line of Text (Cont.)

The braces

- A left brace, {, begins the body of every class declaration
- A corresponding right brace, }, must end each class declaration
- Code between braces should be indented (good practice, 缩进)

```
public class Welcome1
{
    // main method begins execution of Java application
    public static void main ( String[] args )
    {
        System.out.println( "Welcome to Java Programming!" );
    } //end method main
} // end class Welcome1
```

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.2 Our First Program in Java: Printing a Line of Text (Cont.)

Declaring the `main` method （主方法）

`public static void main( String[] args )`

- Starting point of Java applications
- Parentheses after the identifier `main` indicate that it's a program building block called a method （方法）
- Java class declarations normally contain one or more methods
- Keyword `void` （空白） indicates that this method will not return any information

```
// main method begins execution of Java application
public static void main ( String[] args )
{
   System.out.println( "Welcome to Java Programming!" );
} //end method main
```

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.2 Our First Program in Java: Printing a Line of Text (Cont.)

- Body of the method declaration
  - Enclosed in left and right braces
- Statement

```
System.out.println("Welcome to Java Programming!");
```

Instructs the computer to perform an action， print line

- Print the string of characters contained between the double quotation marks （双引号）

```
public class Welcome1
{
   // main method begins execution of Java application
   public static void main ( String[] args )
   {
      System.out.println( "Welcome to Java Programming!" );
   } //end method main
} // end class Welcome1
```

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.2 Our First Program in Java: Printing a Line of Text (Cont.)

- `System.out` object
  - Standard output object （标准输出）
  - Allows Java applications to display strings in the command window （命令行窗口）from which the Java application executes

- `System.out.println` method
  - Displays (or prints) a line of text in the command window.
  - The string in the parentheses is the argument to the method.
  - Positions the output cursor （光标） at the beginning of the next line in the command window

```
public static void main ( String[] args )
  {
     System.out.println( "Welcome to Java Programming!" );
  } //end method main
```

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.2 Our First Program in Java: Printing a Line of Text (Cont.)

- Compiling and executing your first Java application
  - To compile the program, type

    ```
    javac Welcome1.java
    ```

  - If the program contains no syntax errors, preceding command creates a **`Welcome1.class`** file (known as the class file) containing the platform-independent Java bytecodes that represent the application

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.2 Our First Program in Java: Printing a Line of Text (Cont.)

- To execute the program, type `java Welcome1`
- Launches the JVM, which loads the `.class` file for class `Welcome1`
- Note that the `.class` file-name extension is omitted from the preceding command; otherwise, the JVM will not execute the program.

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

```
Administrator: Command Prompt

C:\examples\ch02\fig02_01>javac Welcome1.java

C:\examples\ch02\fig02_01>java Welcome1
Welcome to Java Programming!

C:\examples\ch02\fig02_01>
```

The program outputs to the screen
`Welcome to Java Programming!`

.2 | Executing `Welcome1` from the **Command Prompt**.

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.3 Modifying Our First Java Program

```
// Welcome2 Java program
// Text-printing program

public class Welcome2
{
   // main method begins execution of Java application
   public static void main ( String[] args )
   {
      System.out.print( "Welcome to " );
      System.out.println( "Java Programming!" );
   } //end method main
} // end class Welcome2
```

Print “Welcome to ” and leaves cursor on the same line

Print “Java programming!” starting where the cursor was positioned previously, then outputs a new line character

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.3 Modifying Our First Java Program

- Class `Welcome2` uses two statements to produce the same output as class `Welcome1`
- `System.out`'s method `print` displays a string
- Unlike `println`, `print` does not position the output cursor at the beginning of the next line in the command window

```
System.out.print( "Welcome to " );
System.out.println( "Java Programming!" );
```

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.3 Modifying Our First Java Program (Cont.)

```
// The first Java program
// Text-printing program

public class Welcome3
{
   // main method begins execution of Java application
   public static void main ( String[] args )
   {
      System.out.println( "Welcome\nto\nJava\nProgramming!" );
   } //end method main
} // end class Welcome3
```

\n

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.3 Modifying Our First Java Program (Cont.)

```
System.out.println("Welcome\nto\nJava\nProgramming");
```

- Newline character （换行符） '\n' tells the print methods when to position the output cursor at the beginning of the next line in the command window
- Newline characters are white-space characters
  - White-space characters do not correspond to a visible mark, but represent horizontal (水平) or vertical （垂直） space in typography
- The backslash (\) is called an escape character （转义字符、换码字符）, and forms an escape sequence together with its next character
  - Invokes an alternative interpretation on subsequent character

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

Common Escape Sequences

Escape Sequence	Description
`\n`	Newline. Position the cursor at the beginning of the next line.
`\t`	Horizontal tab. Move the cursor to the next tab stop.
`\r`	Carriage return. Position the cursor at the beginning of the current line (do not advance to the next line). Any characters output after the carriage return overwrite the characters previously output on that line.
`\\`	Use to print a backslash character.
`\"`	Used to print a double-quote character. `System.out.println("\"in quotes\"");` displays `"in quotes"`

© 1992-2010 by Pearson Education, Inc. All Rights Reserved.

2.4 Displaying Text with `printf`

```
// The first Java program
// Text-printing program

public class Welcome4
{
   // main method begins execution of Java application
   public static void main ( String[] args )
   {
      System.out.printf( "%s\n%s\n",
         "Welcome to", "Java Programming!" );
   } //end method main
} // end class Welcome4
```

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

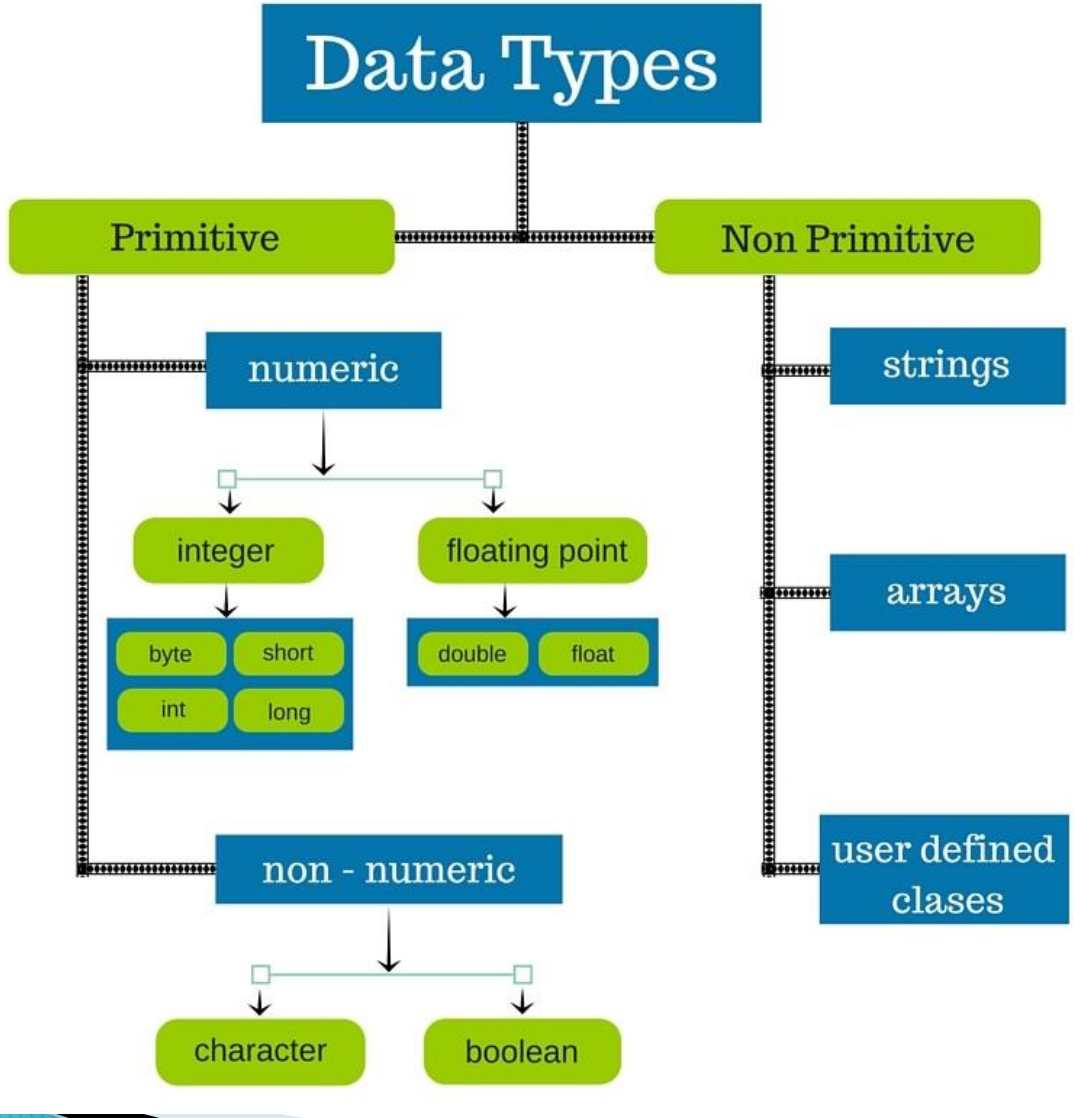
2.4 Displaying Text with printf

```
System.out.printf("%s\n%s\n", "Welcome to",
"Java Programming!");
```

- The method `printf` （格式化输出） displays "formatted" data.
- It takes a format string （格式字符串） as an argument.
- Format specifiers （格式说明符）begin with a percent sign (%) and are followed by a character that represents the data type.
  - Format specifier `%s` is a placeholder (占位符) for a string.

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

Primitive Data Types (基本数据类型)



Primitive Data Types (基本数据类型)

Data Type	Size	Description
byte	1 byte	Stores whole numbers from -128 to 127
short	2 bytes	Stores whole numbers from -32,768 to 32,767
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
boolean	1 bit	Stores true or false values
char	2 bytes	Stores a single character/letter or ASCII values

Primitive Data Types: Integers

Type	Size	Range
byte	8 bits	-128 to +127
short	16 bits	-32,768 to +32,767
int	32 bits	(about) -2 billion to +2 billion
long	64 bits	(about) -10E18 to +10E18

<u>Example:</u> `int year = 2018;`

Primitive Data Types: Floating Point Numbers

Type	Size	Range
float	32 bits	-3.4E+38 to +3.4E+38
double	64 bits	-1.7E+308 to 1.7E+308

<u>Example:</u> `double pi = 3.1415926;`

2.5 Another Application: Adding Integers

```
import java.util.Scanner; // program uses class Scanner

public class Addition {
   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);
      int number1; // first number to add
      int number2; // second number to add
      int sum; // sum of number1 and number2

      System.out.print("Enter first integer: "); // prompt
      number1 = input.nextInt(); // read first number from user

      System.out.print("Enter second integer: "); // prompt
      number2 = input.nextInt(); // read second number from user

      sum = number1 + number2; // add numbers, then store total in sum
      System.out.printf("Sum is %d%n", sum); // display sum
   } // end method main
} // end class Addition
```

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.5 Another Application: Adding Integers

```
import java.util.Scanner
```

- import declaration: Helps the compiler locate a class that is used in this program.
- Related classes are grouped into packages
- java.util package provides commonly-used classes. These classes are collectively called Java class library, or Java Application Programming Interface (Java API).

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.5 Another Application: Adding Integers

- **Variable （变量） declaration statement**

```
Scanner input = new Scanner( System.in );
```

- Variable represents a location in the computer's memory where a value can be stored for use later in a program
- Must be declared with a name and a type before use
- A variable's name enables the program to access the value of the variable in memory
- A variable's type specifies what kind of information is stored at that location in memory

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.5 Another Application: Adding Integers

```
Scanner input = new Scanner( System.in );
```

- The Scanner class enables a program to read data for use in a program
- The data can come from many sources, such as the user at the keyboard or a file on disk
- The assignment operator（赋值）, =.
- The equals sign (=) in a declaration indicates that the variable should be initialized (初始化，i.e., prepared for use in the program) with the result of the expression to the right of the equals sign

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.5 Another Application: Adding Integers

```
Scanner input = new Scanner( System.in );
```

- The new keyword creates an object (we will talk more on objects later)
- Standard input object, System.in, enables applications to read bytes of information typed by the user.
- Scanner object translates these bytes into types that can be used in a program.

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.5 Another Application: Adding Integers

- **Variable declaration statements （变量声明）**

```
int number1; // first number to add
int number2; // second number to add
int sum; // sum of number1 and number2
```

- Several variables of the same type may be declared in one declaration with the variable names separated by commas

```
int number1, number2, sum;
```

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.5 Another Application: Adding Integers

```
System.out.print("Enter first integer: "); // prompt
Number1 = input.nextInt(); // read number from user
```

- Prompt is an output statement that directs the user to take a specific action
- `System` is a class and part of package `java.lang`
- The class `System` does not need to be imported at the beginning of the program because the classes in the `java.lang` package are imported by default)

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.5 Another Application: Adding Integers

```
//read the first number from user
number1 = input.nextInt();
```

- `Scanner` method `nextInt` obtains an integer from the user at the keyboard.
- Program waits for the user to type the number and press the Enter key to submit the number.
- The result of the call to method `nextInt` is placed in variable `number1` by using the assignment operator（赋值）, `=`.
  - `number1` gets the value of `input.nextInt()`

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.5 Another Application: Adding Integers

- **Arithmetic operations**

```
sum = number1 + number2; // add numbers
```

- sum gets the value of `number1 + number2`
- Portions of statements that contain calculations are called expressions （运算式）.

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.5 Another Application: Adding Integers

- **Integer formatted output**

```
System.out.printf( "Sum is %d\n", sum );
```

- Format specifier %d is a placeholder for an `int` value
- The letter d stands for “decimal integer” (十进制整数)

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.6 Memory Concepts

- **Variables**
  - Every variable has a name, a type, a size (in bytes) and a value
  - When a new value is placed into a variable, the new value replaces the previous value (if any)
  - The previous value is lost

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.7 Arithmetic Operations

Operator	Description
+	Additive operator (also used for String concatenation)
-	Subtraction operator
*	Multiplication operator
/	Division operator
%	Remainder operator

- These are binary operators because they operate on two operands
- Integer division yields an integer quotient. The fractional part is simply discarded (int x = 3, int y = 2, int z = x / y, z should be 1 in this case)
- % yields the remainder (余数) after division (10 % 3 = 1)

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

Java operation	Operator	Algebraic expression	Java expression
Addition	+	$f + 7$	`f + 7`
Subtraction	–	$p - c$	`p - c`
Multiplication	*	bm	`b * m`
Division	/	x / y or $\frac{x}{y}$ or $x \div y$	`x / y`
Remainder	%	$r \bmod s$	`r % s`

Fig. 2.11 | Arithmetic operators.

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.7 Arithmetic Expressions

- Arithmetic expressions must be written in straight-line form to facilitate entering programs into the computer

- Expression "a divided by b" must be written as a / b (rather than $\frac{a}{b}$), so that all constants, variables and operators appear in a straight line

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.7 Operator Precedence （操作符优先级）

- *, / and % operations have higher precedence (the three operators have the same level of precedence) than +, -

- If an expression contains several such operations (*, /, %), they are applied from left to right

- If an expression contains several such operations (+，−), the operators are applied from left to right

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

Step 1.	`y = 2 * 5 * 5 + 3 * 5 + 7;` `2 * 5 is 10`	*(Leftmost multiplication)*
Step 2.	`y = 10 * 5 + 3 * 5 + 7;` `10 * 5 is 50`	*(Leftmost multiplication)*
Step 3.	`y = 50 + 3 * 5 + 7;` `3 * 5 is 15`	*(Multiplication before addition)*
Step 4.	`y = 50 + 15 + 7;` `50 + 15 is 65`	*(Leftmost addition)*
Step 5.	`y = 65 + 7;` `65 + 7 is 72`	*(Last addition)*
Step 6.	`y = 72`	*(Last operation—place* 72 *in* y*)*

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.7 Operator Precedence （操作符优先级）

```
public class Precedence
{
    // main method begins execution of Java application
    public static void main ( String[] args )
    {
        int a = 9;
        int b = 2;
        float f = a / b - 1.5f;
        System.out.printf("The result is %f\n", f);
    } //end method main
}
```

```
The result is 2.500000
```

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.7 Arithmetic Expressions

- Parentheses are used to group terms in expressions in the same manner as in algebraic expressions: a * (b + c)

- In case of nested parentheses, the expression in the innermost set of parentheses is evaluated first: ((a + b) * c)

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.8 Decision Making: Equality and Relational Operators

- Condition (条件表达式)
  - An expression that can be `true` or `false`
- Equality operators (`==` and `!=`)
- Relational operators (`>`, `<`, `>=` and `<=`)

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

Standard algebraic equality or relational operator	Java equality or relational operator	Sample Java condition	Meaning of Java condition
Equality operators			
=	==	x == y	x is equal to y
≠	!=	x != y	x is not equal to y
Relational operators			
>	>	x > y	x is greater than y
<	<	x < y	x is less than y
≥	>=	x >= y	x is greater than or equal to y
≤	<=	x <= y	x is less than or equal to y

Fig. 2.14 | Equality and relational operators.

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.8 Decision Making: Equality and Relational Operators

- Equality and relational operators have lower precedence than the arithmetic operators +, -, *, /, %

- 1 + 3 != 5 / 3

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.8 Decision Making: Equality and Relational Operators

- `if` selection statement allows a program to make a decision based on a condition's value

```
if (condition) {
   do something;
}
```

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

```java
// Comparison.java
// Comparison program that inputs two numbers then compare them.
import java.util.Scanner; // program uses class Scanner

public class Comparison {
 // main method begins execution of Java application
   public static void main(String[] args) {
   // create a Scanner to obtain input from the command window
      Scanner input = new Scanner(System.in);
      int number1; // first number to add
      int number2; // second number to add

      System.out.print("Enter first integer: "); // prompt
      number1 = input.nextInt(); // read first number from user

      System.out.print("Enter second integer: "); // prompt
      number2 = input.nextInt(); // read second number from user
```

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

```
    if (number1 == number2)
       System.out.printf("%d == %d%n", number1, number2);
    if (number1 != number2)
       System.out.printf("%d != %d%n", number1, number2);
    if (number1 < number2)
       System.out.printf("%d < %d%n", number1, number2);
    if (number1 > number2)
       System.out.printf("%d > %d%n", number1, number2);
    if (number1 <= number2)
       System.out.printf("%d <= %d%n", number1, number2);
    if (number1 >= number2)
       System.out.printf("%d >= %d%n", number1, number2);

   } // end method main
} // end class Comparison
```

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

```
Enter first integer: 777
Enter second integer: 777
777 == 777
777 <= 777
777 >= 777
```

```
Enter first integer: 1000
Enter second integer: 2000
1000 != 2000
1000 < 2000
1000 <= 2000
```

```
Enter first integer: 2000
Enter second integer: 1000
2000 != 1000
2000 > 1000
2000 >= 1000
```

Fig. 2.15 | Compare integers using `if` statements, relational operators and equality operators. (Part 3 of 3.)

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.

2.8 Summary

- Simple Java applications
- Input and output statements
- Java's primitive types
- Basic memory concepts
- Arithmetic operators
- Precedence of arithmetic operators
- Write decision-making statements
- Relational and equality operators

© 1992-2010 by Pearson Education, Inc.
All Rights Reserved.