Chapter 2

Basic Elements of Java

Chapter Objectives

• Become familiar with the basic components of a Java program, including methods, special symbols, and identifiers
• Explore primitive data types
• Discover how to use arithmetic operators
• Examine how a program evaluates arithmetic expressions
• Explore how mixed expressions are evaluated

Chapter Objectives

• Learn about type casting
• Become familiar with the String type
• Learn what an assignment statement is and what it does
• Discover how to input data into memory by using input statements
• Become familiar with the use of increment and decrement operators
Chapter Objectives

- Examine ways to output results using output statements
- Learn how to import packages and why they are necessary
- Discover how to create a Java application program
- Explore how to properly structure a program, including using comments to document a program

Introduction

- Computer program: a sequence of statements whose objective is to accomplish a task
- Programming: process of planning and creating a program

The Basics of a Java Program

- Java program: collection of classes
- There is a main method in every Java application program
- Token: smallest individual unit of a program
Special Symbols

+  -  *  /
.  ;  ?  ,
<=  !=  ==  >=

Word Symbols

• int
• float
• double
• char
• void
• public
• static
• throws
• return

Java Identifiers

• Names of things
• Consists of and must begin with:
  – Letters
  – Digits
  – The underscore character (_)
  – The dollar sign ($)
Illegal Identifiers

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>employee</td>
<td>There can be no space between employee and Salary.</td>
</tr>
<tr>
<td>Hello1</td>
<td>The exclamation mark cannot be used in an identifier.</td>
</tr>
<tr>
<td>one+two</td>
<td>The symbol + cannot be used in an identifier.</td>
</tr>
<tr>
<td>2nd</td>
<td>An identifier cannot begin with a digit.</td>
</tr>
</tbody>
</table>

Data Types

- **Data type**: set of values together with a set of operations

Primitive Data Types

- Integral
- Floating-point
- Boolean
Primitive Data Types

• Floating-Point Data Types
  – Float: precision = 6 or 7
  – Double: precision = 15
• Boolean: two values
  – True
  – False

Integral Data Types

Values and Memory Allocation for Integral Data Types

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Values</th>
<th>Storage (in bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>0 to 65535</td>
<td>1 (8 bits)</td>
</tr>
<tr>
<td>byte</td>
<td>-128 to 127</td>
<td>1 (8 bits)</td>
</tr>
<tr>
<td>short</td>
<td>-32768 to 32767</td>
<td>2 (16 bits)</td>
</tr>
<tr>
<td>int</td>
<td>-2147483648 to 2147483647</td>
<td>4 (32 bits)</td>
</tr>
<tr>
<td>long</td>
<td>-232 to 232</td>
<td>8 (64 bits)</td>
</tr>
</tbody>
</table>
Arithmetic Operators and Operator Precedence

- Five Arithmetic Operators
  - + addition
  - - subtraction
  - * multiplication
  - / division
  - % mod (modulus) operator (integer operands only)
- Unary operator: operator that has one operand
- Binary operator: operator that has two operands

Order of Precedence

1. *       /       %    (same precedence)
2. +       -       (same precedence)

- Operators in 1 have a higher precedence than operators in 2.
- When operators have the same level of precedence, operations are performed from left to right

Expressions

- Integral expressions
- Floating-point or decimal expressions
- Mixed expressions
Integral Expressions

- All operands are integers
- Examples:
  2 + 3 * 5
  3 + x – y/7
  x + 2 * (y – z) + 18

Floating-point Expressions

- All operands are floating-point numbers
- Examples:
  12.8 * 17.5 – 34.50
  x * 10.5 + y - 16.2

Mixed Expressions

- Operands of different types
- Examples:
  2 + 3.5
  6/4 + 3.9
- Integer operands yield an integer result; floating-point numbers yield floating-point results
- If both types of operands are present, the result is a floating-point number
- Precedence rules are followed
Type Conversion (Casting)

- Used to avoid implicit type coercion
- Syntax: `(dataTypeName) expression`
- Expression evaluated first, then type converted to `dataTypeName`
- Examples:
  - `(int)(7.9 + 6.7) = 14`
  - `(int)(7.9) + (int)(6.7) = 13`

The class String

- Used to manipulate strings
- String:
  - Sequence of zero or more characters
  - Enclosed in double quotation marks
  - Null or empty strings have no characters
  - Numeric strings consist of integers or decimal numbers
  - Length is the number of characters in a string

Parsing Numeric Strings

- String to int: `Integer.parseInt(strExpression)`
- String to float: `Float.parseFloat(strExpression)`
- String to double: `Double.parseDouble(strExpression)`

*strExpression: expression containing a numeric string
Input

- **Named constant**
  - cannot be changed during program execution
  - declared by using the reserved word `final`
  - initialized when it is declared

- **Variable (name, value, data type, size)**
  - content may change during program execution
  - must be declared before it can be used
  - may not be automatically initialized
  - if new value is assigned, old one is destroyed
  - value can only be changed by an assignment statement or an input (read) statement

Input

- Standard input stream object: System.in
- Input numeric data to program
  - Separate by blanks, lines, or tabs
- To read a line of characters:
  1. Create an input stream object of the class `BufferedReader`
  2. Use the method `readLine`

Increment and Decrement Operators

- `++` increments the value of its operand by 1
- `--` decrements the value of its operand by 1

Syntax
- Pre-increment: `++variable`
- Post-increment: `variable++`
- Pre-decrement: `--variable`
- Post-decrement: `variable--`
Strings and the Operator +

- Operator + can be used to concatenate two strings or a string and a numeric value or character
- Example:

```java
String str;
int num1, num2;
num1 = 12;
num2 = 26;
str = “The sum = ” + num1 + num2;
After this statement executes, the string assigned to str is:
str = “The sum = 122”;```

Output

- Standard output object: System.out
- Methods
  - print
  - println
  - flush
- Syntax
  ```java
  System.out.print(stringExp);
  System.out.println(stringExp);
  System.out.flush();
  ```

Commonly Used Escape Sequences

<table>
<thead>
<tr>
<th>Escape Sequence</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\n</td>
<td>Newline</td>
</tr>
<tr>
<td>\t</td>
<td>Tab</td>
</tr>
<tr>
<td>\b</td>
<td>Backspace</td>
</tr>
<tr>
<td>\r</td>
<td>Return</td>
</tr>
<tr>
<td>\f</td>
<td>Formfeed</td>
</tr>
<tr>
<td>‘</td>
<td>Single quotation</td>
</tr>
<tr>
<td>&quot;</td>
<td>Double quotation</td>
</tr>
</tbody>
</table>
Packages, Classes, Methods, and the import Statement

- **Package**: collection of related classes
- **Class**: consists of methods
- **Method**: designed to accomplish a specific task

import Statement

- Used to import the components of a package into a program
- Reserved word
- `import java.io.*;` imports the components of the `package java.io` into the program
- Primitive data types and the `class String`
  - Part of the Java language
  - Don’t need to be imported

Creating a Java Application Program

- Syntax of a class
  ```java
  public class ClassName
  {
      classMembers
  }
  ```
- Syntax of the main method
  ```java
  public static void main (String[] args) {
      statements
  }
  ```
Programming Style and Form

- Know common syntax errors and rules
- Use blanks appropriately
- Semicolon: statement terminator
- Important to have well-documented code
- Good practice to follow traditional rules for naming identifiers

More on Assignment Statements

- variable = variable * (expression);
  is equivalent to
- variable *= expression;
  Similarly,
- variable = variable + (expression);
  is equivalent to:
- variable += expression;

Programming Examples

- Convert Length Program
  – Input: Length in feet and inches
  – Output: Equivalent length in centimeters
- Make Change Program
  – Input: Change in cents
  – Output: Equivalent change in half-dollars, quarters, dimes, nickels, and pennies
Chapter Summary

- Basic Elements of a Java program include:
  - The main method
  - Reserved words
  - Special symbols
  - Identifiers
  - Data types
  - Expressions
  - Input
  - Output
  - Statements

Chapter Summary

- To create a Java application it is important to understand:
  - Syntax rules
  - Semantic rules
  - How to manipulate strings and numbers
  - How to declare variables and named constants
  - How to receive input and display output
  - Good programming style and form