Chapter 2

Elementary Programming

2.1 Introduction

- You will learn elementary programming using Java **primitive** data types and related subjects, such as variables, constants, operators, expressions, and input and output.

2.2 Writing Simple Programs

- Writing a program involves designing algorithms and data structures, as well as translating algorithms into programming code.
- An ***Algorithm*** describes how a problem is solved in terms of the actions to be executed, and it specifies the order in which the actions should be executed.
- Computing an area of a circle. The algorithm for this program can be described as follows:

  1. Read in the Radius
  2. Compute the area using the following formula:
     Area = radius * radius * ∏
  3. Display the area.

- Java provides data types for representing integers, floating-point numbers, characters, and Boolean types. These types are known as ***primitive data types***.
- When you *code*, you translate an algorithm into a **programming language** understood by the computer.
- The outline of the program is:

```
public class ComputeArea {
  public static void main(String[] args) {
    double radius; // Declare radius
    double area; // Declare area

    // Assign a radius
    radius = 20; // New value is radius

    // Compute area
    area = radius * radius * 3.14159;

    // Display results
    System.out.println("The area for the circle of radius " +
      radius + " is " + area);
  }
}
```

- The program needs to **declare** a symbol called a variable that will represent the radius. **Variables** are used to store data and computational results in the program.
- Use descriptive names rather than x and y. Use radius for radius, and area for area. Specify their data types to let the compiler know what radius and area are, indicating whether they are integer, float, or something else.
- The program declares radius and area as double-precision variables. The reserved word **double** indicates that radius and area are double-precision floating-point values stored in the computer.
- For the time being, we will assign a fixed number to radius in the program. Then, we will compute the area by assigning the expression radius * radius * 3.14159 to area.
- The program's output is:
  The area for the circle of radius 20.0 is 1256.636
- A string constant should not cross lines in the source code. Use the **concatenation** operator (+) to overcome such problem.

2.3 Reading Input from the Console

Getting Input Using Scanner

- Create a Scanner object

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

- Use the methods next( ), nextByte( ), nextShort( ), nextInt( ), nextLong( ), nextFloat( ), nextDouble( ), or nextBoolean( ) to obtain to a string, byte, short, int, long, float, double, or boolean value. For example,

```
System.out.print("Enter a double value: ");
Scanner scanner = new Scanner(System.in);
double d = scanner.nextDouble( );
```

- Listing 2.2 ComputeAreaWithConsoleInput.java

```
import java.util.Scanner; // Scanner is in the java.util package

public class ComputeAreaWithConsoleInput {
  public static void main(String[] args) {
    // Create a Scanner object
    Scanner input = new Scanner(System.in);

    // Prompt the user to enter a radius
    System.out.print("Enter a number for radius: ");
    double radius = input.nextDouble();

    // Compute area
    double area = radius * radius * 3.14159;

    // Display result
    System.out.println("The area for the circle of radius " +
      radius + " is " + area);
  }
}
```

```
Enter a number for radius: 23
The area for the circle of radius 23.0 is 1661.90111
```

- **Caution**

  By default a Scanner object reads a string separated by **whitespaces** (i.e. ‘ ‘, ‘\t’, ‘\f’, ‘\r’, and ‘\n’).

2.4 Identifiers

- Programming languages use special symbols called ***identifiers*** to name such programming entities as variables, constants, methods, classes, and packages.
- The following are the rules for naming identifiers:
    - An identifier is a sequence of characters that consist of **letters, digits, underscores (_), and dollar signs ($)**.
    - An identifier must start with a letter, an underscore (_), or a dollar sign ($). It **cannot** start with a digit.
    - An identifier cannot be a **reserved** word. (See Appendix A, "Java Keywords," for a list of reserved words).
    - An identifier **cannot** be true, false, or null.
    - An identifier can be of **any** length.
- For example:
    - Legal identifiers are for example: $2, ComputeArea, area, radius, and showMessageDialog.
    - Illegal identifiers are for example: 2A, d+4.
    - Since Java is **case-sensitive**, X and x are different identifiers.

2.5 Variables

- Variables are used to **store** data in a program.
- You can write the code shown below to compute the area for different radius:

```
// Compute the first area
radius = 1.0;
area = radius*radius*3.14159;
System.out.println("The area is " + area + " for radius "+radius);

// Compute the second area
radius = 2.0;
area = radius*radius*3.14159;
System.out.println("The area is " + area + " for radius "+radius);
```

Declaring Variables

- Variables are used for representing data of a certain **type**.
- To use a variable, you declare it by telling the compiler the name of the variable as well as what type of data it represents. This is called variable **declaration**.
- Declaring a variable tells the compiler to allocate appropriate memory space for the variable based on its data type. The following are examples of variable declarations:

```
int x;          // Declare x to be an integer variable;
double radius; // Declare radius to be a double variable;
char a;         // Declare a to be a character variable;
```

- If variables are of the same type, they can be declared together using **short-hand** form:

```
Datatype var1, var2, …, varn;    ➔ variables are separated by commas
```

Declaring and Initializing Variables in One Step

- You can declare a variable and initialize it in one step.

```
int x = 1;
```

This is equivalent to the next two statements:

```
int x;
x = 1;
```

```
// shorthand form to declare and initialize vars of same type
int i = 1, j = 2;
```

- **Tip**: A variable must be declared **before** it can be assigned a value.

2.6 Assignment Statements and Assignments Expressions

- After a variable is declared, you can assign a value to it by using an assignment statement. The syntax for assignment statement is:

  variable = expression;

```
x = 1;                    // Assign 1 to x;  ➔ Thus 1 = x is wrong
radius = 1.0;             // Assign 1.0 to radius;
a = 'A';                  // Assign 'A' to a;
x = 5 * (3 / 2) + 3 * 2;  // Assign the value of the expression to x;
x = y + 1;                // Assign the addition of y and 1 to x;
```

- The variable can also be used in the expression.

```
x = x + 1;      // the result of x + 1 is assigned to x;
```

- To assign a value to a variable, the variable name must be on the **left** of the assignment operator.

```
1 = x;          // would be wrong
```

- In Java, an assignment statement can also be treated as an expression that evaluates to the value being assigned to the variable on the left-hand side of the assignment operator. For this reason, an assignment statement is also known as an **assignment expression**, and the symbol = is referred to as the **assignment operator**.

```
System.out.println(x = 1);
```

  which is equivalent to

```
x = 1;
System.out.println(x);
```

  The following statment is also correct:

```
i = j = k = 1;
```

  which is equivalent to

```
k = 1; j = k; i = j;
```

2.7 Named Constants

- The value of a variable may change during the execution of the program, but a constant represents permanent data that **never** change.
- The syntax for declaring a constant:

```
final datatype CONSTANTNAME = VALUE;

final double PI = 3.14159;  ➔ // Declare a constant
final int SIZE = 3;
```

- A constant **must** be declared and initialized before it can be used. You **cannot** change a constant's value once it is declared. By convention, constants are named in **uppercase**.

```
import java.util.Scanner; // Scanner is in the java.util package

public class ComputeAreaWithConstant {
  public static void main(String[] args) {
    final double PI = 3.14159; // Declare a constant

    // Create a Scanner object
    Scanner input = new Scanner(System.in);

    // Prompt the user to enter a radius
    System.out.print("Enter a number for radius: ");
    double radius = input.nextDouble();

    // Compute area
    double area = radius * radius * PI;

    // Display result
    System.out.println("The area for the circle of radius " +
      radius + " is " + area);
  }
}
```

- **Note:** There are three benefits of using constants:
  - You don't have to **repeatedly** type the same value.
  - The value can be changed in a **single** location.
  - The program is easy to **read**.