Cape Peninsula University of Technology

Software Engineering 2 (SDN260S)

Introduction to Java Programming

H. Mataifa

Department of Electronic, Electrical and Computer Engineering

Programming in Java

- Java is an *object-oriented programming (**OOP**)* language, every Java program is written as a combination of one or more classes

```
1  // Fig. 2.1: Welcome1.java
2  // Text-printing program.
3
4  public class Welcome1
5  {
6     // main method begins execution of Java application
7     public static void main( String[] args )
8     {
9        System.out.println( "Welcome to Java Programming!" );
10    } // end method main
11 } // end class Welcome1
```

Fig. 2.1: Example of a simple Java program

Notes:

- Comments are used in a program to improve the understandability of the program's functionality
- Java uses the end-of-line (//) and traditional (/* */) comment types
- Errors that commonly occur in computer programs are syntax and logic errors. *Syntax errors* violate the rules of programming language (similar to grammar rules in natural language), and prevent the program from being compiled (i.e. cause *compile-time errors*). *Logic errors* lead to the program producing results different from those which are expected. They do not prevent the program from executing successfully

Class/method declaration

- Class declaration: class keyword introduces a class declaration and is immediately followed by the class name. public is another keyword that usually appears in a class declaration line

- By convention, class names begin with a capital letter, and capitalize the first letter of each other word in the case of multiple-word class names. A class name is an identifier – a series of characters consisting of letters, digits, and certain special symbols (e.g. underscore, dollar sign), may not begin with a digit, and does not contain any spaces

- Java is case-sensitive – uppercase and lowercase letters are distinct – thus value and Value are two distinct identifiers

- Declaring a method: the starting point of every Java application is the main method. Notice how it is declared in line 7. The main method is required for the Java Virtual Machine (**JVM**) to be able to execute the application

```
6     // main method begins execution of Java application
7     public static void main( String[] args )
8     {
9        System.out.println( "Welcome to Java Programming!" );
10    } // end method main
```

- The main parts of the method declaration in line 7 are (1) access specifier (public), access modifier (static), return type (void), method name (main), and parameter list ((String[] args))

Escape sequences

- Escape sequences are special character sequences with a special meaning. They are formed by a combination of a backslash and a letter or other character. They are frequently used in the System.out object along with the print method to print to the output console

Escape sequence	Description
`\n`	Newline. Position the screen cursor at the beginning of the next line.
`\t`	Horizontal tab. Move the screen cursor to the next tab stop.
`\r`	Carriage return. Position the screen 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.
`\\`	Backslash. Used to print a backslash character.
`\"`	Double quote. Used to print a double-quote character. For example, `System.out.println( "\"in quotes\"" );` displays `"in quotes"`.

Formatted output

- Formatted output can be displayed using method System.out.printf(); the first argument of the method is a format string, which may consist of fixed text and format specifiers; each format specifier is a placeholder for a value and specifies the type of data to output. A format specifier begins with the percentage (%) character, followed by the character that represents the data type. For example, the format specifier %s is a placeholder for a string data type

- The format string is followed by a comma-separated list of values to replace the format specifiers in the format string, listed sequentially

```
1  // Fig. 2.6: Welcome4.java
2  // Displaying multiple lines with method System.out.printf.
3
4  public class Welcome4
5  {
6     // main method begins execution of Java application
7     public static void main( String[] args )
8     {
9        System.out.printf( "%s\n%s\n",
10          "Welcome to", "Java Programming!" );
11    } // end method main
12 } // end class Welcome4
```

Another application: adding integers

- The application requests the user to enter two integers, adds them up, then displays the result
- Notice the use of variables to keep track of the data the program is working with; a variable is an identifier for a location in memory where a value can be stored and be used later in a program. Declaring a variable in Java requires specifying the data type and variable name

```
1  // Fig. 2.7: Addition.java
2  // Addition program that displays the sum of two numbers.
3  import java.util.Scanner; // program uses class Scanner
4
5  public class Addition
6  {
7     // main method begins execution of Java application
8     public static void main( String[] args )
9     {
10       // create a Scanner to obtain input from the command window
11       Scanner input = new Scanner( System.in );
12
13       int number1; // first number to add
14       int number2; // second number to add
15       int sum; // sum of number1 and number2
16
17       System.out.print( "Enter first integer: " ); // prompt
18       number1 = input.nextInt(); // read first number from user
19
20       System.out.print( "Enter second integer: " ); // prompt
21       number2 = input.nextInt(); // read second number from use
22
23       sum = number1 + number2; // add numbers, then store total
24
25       System.out.printf( "Sum is %d\n", sum ); // display sum
26    } // end method main
27 } // end class Addition
```

Another application: adding integers

- Java packages: a Java package is a named group of related classes that makes it easy to reuse predefined classes in Java applications. Together, the collection of Java packages are referred to as the Java Class Library or the Java Application Programming Interface (**API**)
- To use a predefined class in a Java application, we have to import it; for example, the statement:

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

- is an import declaration that gives the program access to class Scanner which is part of package java.util
- Notice that all import declarations need to come before any class definitions in the program. That is, they have to be the placed at the top of the program
- To use the Scanner class in the program, we have to create an object of the class, as done by the statement:

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

- A Scanner enables a program to read data (e.g. numbers, strings) for use in a program; the source of data needs to be specified when creating the Scanner object. In this case, the source of the data is the keyboard, and the standard input object, System.in, acts as the channel through which the Scanner object reads from the keyboard

Another application: adding integers

- Primitive and reference data types: Java has two main types of data types that can be used to define variables: primitive data types and reference data types.
  - Primitive data types include *boolean* (boolean), *byte* (byte), *character* (char), *short* (short), *integer* (int), *long* (long), *float* (float), *double* (double).
  - Reference types are essentially user-defined types, defined as classes (e.g. String)

- To request (prompt) the user for input, we use System.out.print():

```
System.out.print( "Enter first integer: " ); // prompt
```

- To capture the (integer) input entered by the user from the keyboard, we use the Scanner object's nextInt() method:

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

- To display the result of the calculation, we use System.out.printf():

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

```
System.out.printf( "Sum is %d\n", ( number1 + number2 ) );
```

Arithmetic operations in Java

- Most Java applications perform arithmetic operations, so it's useful for us to be familiar with how arithmetic operations are implemented in Java, as well as the concept of arithmetic operator precedence

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.

Operator(s)	Operation(s)	Order of evaluation (precedence)
* / %	Multiplication Division Remainder	Evaluated first. If there are several operators of this type, they're evaluated from left to right.
+ -	Addition Subtraction	Evaluated next. If there are several operators of this type, they're evaluated from left to right.
=	Assignment	Evaluated last.

Fig. 2.12 | Precedence of arithmetic operators.

Algebra: $z = pr\%q + w/x - y$

Java:

```
z = p * r % q + w / x - y;
```

6 1 2 4 3 5

```
y = a * x * x + b * x + c;
```

6 1 2 4 3 5

Examples of arithmetic operator precedence

Decision making: equality and relational operators

- Equality and relational operators are used in conditional statements, which evaluate to either true or false, to control program execution. Use of equality and relational operators is summarized in Figure 2.14

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.

Example of program making use of equality and relational operators

```
1  // Fig. 2.15: Comparison.java
2  // Compare integers using if statements, relational operators
3  // and equality operators.
4  import java.util.Scanner; // program uses class Scanner
5
6  public class Comparison
7  {
8     // main method begins execution of Java application
9     public static void main( String[] args )
10    {
11       // create Scanner to obtain input from command line
12       Scanner input = new Scanner( System.in );
13
14       int number1; // first number to compare
15       int number2; // second number to compare
16
17       System.out.print( "Enter first integer: " ); // prompt
18       number1 = input.nextInt(); // read first number from user
19
20       System.out.print( "Enter second integer: " ); // prompt
21       number2 = input.nextInt(); // read second number from user
22
23       if ( number1 == number2 )
24          System.out.printf( "%d == %d\n", number1, number2 );
25
26       if ( number1 != number2 )
27          System.out.printf( "%d != %d\n", number1, number2 );
28
29       if ( number1 < number2 )
30          System.out.printf( "%d < %d\n", number1, number2 );
31
32       if ( number1 > number2 )
33          System.out.printf( "%d > %d\n", number1, number2 );
34
35       if ( number1 <= number2 )
36          System.out.printf( "%d <= %d\n", number1, number2 );
37
38       if ( number1 >= number2 )
39          System.out.printf( "%d >= %d\n", number1, number2 );
40    } // end method main
41 } // end class Comparison
```

Exercises

2.15 *(Arithmetic)* Write an application that asks the user to enter two integers, obtains them from the user and prints their sum, product, difference and quotient (division). Use the techniques shown in Fig. 2.7.

2.16 *(Comparing Integers)* Write an application that asks the user to enter two integers, obtains them from the user and displays the larger number followed by the words `"is larger"`. If the numbers are equal, print the message `"These numbers are equal"`. Use the techniques shown in Fig. 2.15.

2.17 *(Arithmetic, Smallest and Largest)* Write an application that inputs three integers from the user and displays the sum, average, product, smallest and largest of the numbers. Use the techniques shown in Fig. 2.15. [*Note:* The calculation of the average in this exercise should result in an integer representation of the average. So, if the sum of the values is 7, the average should be 2, not 2.3333….]