CO225-Apr2025 : Software Construction

Java Basics

Steps in Writing a Java Program

Step 1: Write the source code `Xxx.java` using a programming text editor (such as Sublime Text, Atom, Notepad++, Textpad, gEdit) or an IDE (such as Eclipse or NetBeans).

Step 2: Compile the source code `Xxx.java` into Java portable bytecode `Xxx.class` using the JDK Compiler by issuing command:

```
javac Xxx.java
```

Step 3: Run the compiled bytecode `Xxx.class` with the input to produce the desired output, using the Java Runtime by issuing command:

```
java Xxx
```

Java Program Template

You can use the following *template* to write your Java programs. Choose a meaningful "*Classname*" that reflects the *purpose* of your program, and write your programming statements inside the body of the `main()` method. Don't worry about the other terms and keywords now. I will explain them in due course. Provide comments in your program!

```
/**
 * Comment to state the purpose of the program
 */
public class Classname {   // Choose a meaningful Classname. Save as
"Classname.java"
   public static void main(String[] args) {  // Entry point of the program
      // Your programming statements here!!!
   }
}
```

A Sample Program Illustrating Sequential, Decision and Loop Constructs

Below is a simple Java program that demonstrates the three basic programming constructs: *sequential*, *loop*, and *conditional*.

```
/**
 * Find the sums of the running odd numbers and even numbers from a given
lowerbound
 * to an upperbound. Also compute their absolute difference.
 */
public class OddEvenSum {  // Save as "OddEvenSum.java"
   public static void main(String[] args) {
      // Declare variables
      final int LOWERBOUND = 1;
      final int UPPERBOUND = 1000;  // Define the bounds
      int sumOdd  = 0;    // For accumulating odd numbers, init to 0
      int sumEven = 0;    // For accumulating even numbers, init to 0
      int absDiff;        // Absolute difference between the two sums

      // Use a while loop to accumulate the sums from LOWERBOUND to
UPPERBOUND
      int number = LOWERBOUND;   // loop init
      while (number <= UPPERBOUND) {  // loop test
            // number = LOWERBOUND, LOWERBOUND+1, LOWERBOUND+1, ...,
UPPERBOUND
         // A if-then-else decision
         if (number % 2 == 0) {  // Even number
            sumEven += number;   // Same as sumEven = sumEven + number
         } else {                // Odd number
            sumOdd += number;    // Same as sumOdd = sumOdd + number
         }
         ++number;  // loop update for next number
      }
      // Another if-then-else Decision
      if (sumOdd > sumEven) {
         absDiff = sumOdd - sumEven;
      } else {
         absDiff = sumEven - sumOdd;
      }
      // OR using one liner conditional expression
      //absDiff = (sumOdd > sumEven) ? sumOdd - sumEven : sumEven - sumOdd;
```

```
      // Print the results
      System.out.println("The sum of odd numbers from " + LOWERBOUND + " to
" + UPPERBOUND + " is: " + sumOdd);
      System.out.println("The sum of even numbers from " + LOWERBOUND + "
to " + UPPERBOUND + " is: " + sumEven);
      System.out.println("The absolute difference between the two sums is:
" + absDiff);
   }
}
```

The expected outputs are:

```
The sum of odd numbers from 1 to 1000 is: 250000
The sum of even numbers from 1 to 1000 is: 250500
The absolute difference between the two sums is: 500
```

Flow Control

Syntax	Example

`// if-then`
`if (booleanTest) {`
`  trueBlock;`
`}`
`// next statement`
	`int mark = 80;`
	`if (mark >= 80) {`
	`  System.out.println("Well Done!");`
	`  System.out.println("Keep it up!");`
	`}`
	`System.out.println("Life goes on!);`
	
	`double temperature = 80.1;`
	`if (temperature > 80) {`
	`  System.out.println("Too Hot!");`
	`}`
	`System.out.println("yummy!");`

`// if-then-else`
`if (booleanTest) {`
`  trueBlock;`
`} else {`
`  falseBlock;`
`}`
`// next statement`
	`int mark = 50;  // Assume that mark is [0, 100]`
	`if (mark >= 50) {  // [50, 100]`
	`  System.out.println("Congratulation!");`
	`  System.out.println("Keep it up!");`
	`} else {  // [0, 49]`
	`  System.out.println("Try Harder!");`
	`}`
	`System.out.println("Life goes on!");`
	
	`double temperature = 80.1;`
	`if (temperature > 80) {`
	`  System.out.println("Too Hot!");`
	`} else {`
	`  System.out.println("Too Cold!");`
	`}`
	`System.out.println("yummy!");`

Example code :

```
// if-then
int absValue = -5;
if (absValue < 0) absValue = -absValue;   // Only one statement in the
block, can omit { }

int min = 0, value = -5;
if (value < min) {     // More than one statements in the block, need { }
   min = value;
```

```
   System.out.println("Found new min");
}

// if-then-else
int mark = 50;
if (mark >= 50)
   System.out.println("PASS");   // Only one statement in the block, can
omit { }
else {                          // More than one statements in the block,
need { }
   System.out.println("FAIL");
   System.out.println("Try Harder!");
}

// Harder to read without the braces
int number1 = 8, number2 = 9, absDiff;
if (number1 > number2) absDiff = number1 - number2;
else absDiff = number2 - number1;
```

Loop Flow Control

Syntax	Example

`// while-do loop`
`while (booleanTest) {`
`  body;`
`}`
`// next statement`
	`// Sum from 1 to upperbound`
	`int sum = 0;`
	`final int UPPERBOUND = 100;`
	`int number = 1;  // init`
	`while (number <= UPPERBOUND) {`
	`  // number = 1, 2, 3, ..., UPPERBOUND`
	`  //for each iteration`
	`  sum += number;`
	`  ++number;  // update`
	`}`
	`System.out.println("sum is: " + sum);`
	
	`// Factorial of n (=1*2*3*...*n)`
	`int n = 5;`
	`int factorial = 1;`
	`int number = 1;  // init`
	`while (number <= n) {`
	`  // num = 1, 2, 3, ..., n for each iteration`
	`  factorial *= number;`
	`  ++num;  // update`
	`}`
	`System.out.println("factorial is: " + factorial);`

`// do-while loop`
`do {`
`  body;`
`} while (booleanTest;`
`// next statement`
`  // Need a semi-colon to`
`  // terminate statement`
	`// Sum from 1 to upperbound`
	`int sum = 0;`
	`final int UPPERBOUND = 100;`
	`int number = 1;  // init`
	`do {`
	`  // number = 1, 2, 3, ..., UPPERBOUND`
	`  // for each iteration`
	`  sum += number;`
	`  ++number;  // update`
	`} while (number <= UPPERBOUND);`
	`System.out.println("sum is: " + sum);`
	
	`// Factorial of n (=1*2*3*...*n)`
	`int n = 5;`
	`int factorial = 1;`
	`int number = 1;  // init`
	`do {`
	`  // num = 1, 2, 3, ..., n for each iteration`
	`  factorial *= number;`
	`  ++number;  // update`
	`} while (number <= n);`
	`System.out.println("factorial is: " + factorial);`

```
// for-loop
for (init; booleanTest; update) {
   body;
}
// next statement
```

```
// Sum from 1 to upperbound
int sum = 0;
final int UPPERBOUND = 100;
for (int number = 1; number <= UPPERBOUND; ++number) {
   // num = 1, 2, 3, ..., UPPERBOUND
   sum += number;
}
System.out.println("sum is: " + sum);

// Factorial of n (=1*2*3*...*n)
int n = 5;
int factorial = 1;
for (int number = 1; number <= n; ++number) {
   // number = 1, 2, 3, ..., n
   factorial *= number;
}
System.out.println("factorial is: " + factorial);
```

Input/Output

Formatted Output via "printf()" (JDK 5)

`System.out.print()` and `println()` do not provide output formatting, such as controlling the number of spaces to print an `int` and the number of decimal places for a `double`.

Java SE 5 introduced a new method called `printf()` for *formatted* output (which is modeled after C Language's `printf()`). `printf()` takes the following form:

```
printf(formattingString, arg1, arg2, arg3, ... );
```

Example	Output
`// Without specifying field-width` `System.out.printf("Hi,|%s|%d|%f|,@xyz%n", "Hello", 123, 45.6);`	`Hi,|Hello|123|45.600000|,@xyz`
`// Specifying the field-width and decimal places for double` `System.out.printf("Hi,|%6s|%6d|%6.2f|,@xyz%n", "Hello", 123, 45.6);`	`Hi,| Hello|  123| 45.60|,@xyz`

`// Various way to format integers:`
`// flag '-' for left-align, '0' for padding with 0`
`System.out.printf("Hi,|%d|%5d|%-5d|%05d|,@xyz%n", 111, 222, 333, 444);`
	`Hi,|111|  222|333  |00444|,@xyz`

`// Various way to format floating-point numbers:`
`// flag '-' for left-align`
`System.out.printf("Hi,|%f|%7.2f|%.2f|%-7.2f|,@xyz%n", 11.1, 22.2, 33.3, 44.4);`
	`Hi,|11.100000|  22.20|33.30|44.40  |,@xyz`

`// To print a '%', use %% (as % has special meaning)` `System.out.printf("The rate is: %.2f%%.%n", 1.2);`	`The rate is: 1.20%.`

Input From Keyboard via "Scanner" (JDK 5)

```
import java.util.Scanner;    // Needed to use the Scanner
/**
 * Test input scanner
 */
public class ScannerTest {
   public static void main(String[] args) {
      // Declare variables
      int num1;
      double num2;
      String str;

      // Read inputs from keyboard
      // Construct a Scanner named "in" for scanning System.in (keyboard)
      Scanner in = new Scanner(System.in);
      System.out.print("Enter an integer: ");  // Show prompting message
```

```
      num1 = in.nextInt();         // Use nextInt() to read an int
      System.out.print("Enter a floating point: ");  // Show prompting
message
      num2 = in.nextDouble();      // Use nextDouble() to read a double
      System.out.print("Enter a string: ");  // Show prompting message
      str  = in.next();            // Use next() to read a String token, up
to white space
      in.close();   // Scanner not longer needed, close it

      // Formatted output via printf()
      System.out.printf("%s, Sum of %d & %.2f is %.2f%n", str, num1, num2,
num1+num2);
   }
}
```

You can also use method nextLine() to read in the entire line, including white spaces, but excluding the terminating newline.

```
/**
 * Test Scanner's nextLine()
 */
import java.util.Scanner;    // Needed to use the Scanner
public class ScannerNextLineTest {
   public static void main(String[] args) {
      Scanner in = new Scanner(System.in);
      System.out.print("Enter a string (with space): ");
      // Use nextLine() to read entire line including white spaces,
      //   but excluding the terminating newline.
      String str = in.nextLine();
      in.close();
      System.out.printf("%s%n", str);
   }
}
```

Try not to mix nextLine() and nextInt()|nextDouble()|next() in a program (as you may need to flush the newline from the input buffer).