# Chapter 2 — Answers

Prepared for Michael

## Exercise 2.1 (Fill in the blanks)

A) A(n) left brace { begins the body of every method, and a(n) right brace } ends the body of every method.  
B) You can use the if statement to make decisions.  
C) // begins an end-of-line comment.  
D) space, tab and newline are called white space.  
E) Keywords are reserved for use by Java.  
F) Java applications begin execution at method main.  
G) Methods System.out.print, System.out.println and System.out.printf display information in a command window.

## Exercise 2.2 (True / False)

A) FALSE — Comments are ignored by the compiler and are NOT printed. (They are for programmers.)  
B) TRUE — In Java you must give a type when declaring a variable (for local variables and fields).  
C) FALSE — Java is case-sensitive; number and NuMbEr are different identifiers.  
D) FALSE — The remainder operator (%) can be used with integer and floating-point (double) operands.  
E) FALSE — \*, / and % have higher precedence than + and -; they are not all the same precedence.

## Exercise 2.3 (Statements)

A) int c, thisIsAVariable, q76354, number;  
B) System.out.print("Enter an integer: ");  
C) int value = input.nextInt(); // assuming Scanner input has been created  
D) System.out.println("This is a Java program");  
E) System.out.printf("%s%n%s%n", "This is a Java", "program"); // first line ends with 'Java'  
F) if (number != 7) System.out.println("The variable number is not equal to 7");

## Exercise 2.4 (Fix the errors)

A) Original: if (c < 7); System.out.println("c is less than 7");  
 Corrected: if (c < 7) System.out.println("c is less than 7");  
 Explanation: the semicolon ends the if, so the print always executes. Remove the semicolon.  
B) Original: if (c => 7) System.out.println("c is equal to or greater than 7");  
 Corrected: if (c >= 7) System.out.println("c is equal to or greater than 7");

## Exercise 2.5 (Declarations and statements)

A) // Declare variables  
int x, y, z, result;  
B) // Prompt user to enter first integer  
System.out.print("Enter the first integer: ");  
C) // Read first integer (Scanner input assumed)  
x = input.nextInt();  
D) System.out.print("Enter the second integer: ");  
E) y = input.nextInt();  
F) System.out.print("Enter the third integer: ");  
G) z = input.nextInt();  
H) // Compute product and assign  
result = x \* y \* z;  
I) // Display using printf  
System.out.printf("Product is %d%n", result);  
J) // Create Scanner  
Scanner input = new Scanner(System.in);  
K) // Comment about the program  
// This program computes the product of three integers.

## Exercise 2.6 (Complete program: product of three integers)

// Product.java  
import java.util.Scanner;  
  
public class Product {  
 public static void main(String[] args) {  
 Scanner input = new Scanner(System.in);  
 System.out.print("Enter the first integer: ");  
 int x = input.nextInt();  
 System.out.print("Enter the second integer: ");  
 int y = input.nextInt();  
 System.out.print("Enter the third integer: ");  
 int z = input.nextInt();  
 int result = x \* y \* z;  
 System.out.printf("Product is %d%n", result);  
 input.close();  
 }  
}

## Exercise 2.7 (Fill in the blanks)

A) Comments are used to document a program and improve its readability.  
B) A decision can be made in a Java program with an if statement.  
C) Calculations are normally performed by assignment statements (using arithmetic operators).  
D) The arithmetic operators with the same precedence as multiplication are / and %.  
E) When parentheses are nested, the innermost set of parentheses is evaluated first.  
F) A location in memory that may contain different values is called a variable.

## Exercise 2.8 (Short statements)

A) System.out.print("Enter an integer: ");  
B) a = b \* c; // assign product of b and c to a  
C) // This program performs a sample payroll calculation

## Exercise 2.9 (True / False)

A) FALSE — Operators have precedence rules; not simply left-to-right. When precedence is the same, evaluation is left-to-right for operators of the same precedence.  
B) TRUE — All listed names are valid Java identifiers (identifiers may contain letters, digits, underscore, and $ and cannot begin with a digit).  
C) TRUE — A valid arithmetic expression with no parentheses is evaluated using operator precedence and left-to-right among operators with the same precedence.  
D) FALSE — Those listed beginning with digits (3g, 87, 67h2, 2h) are invalid because identifiers cannot begin with a digit; h22 would be valid.

## Exercise 2.10 (Assume x=2 and y=3)

A) System.out.printf("x = %d%n", x); // prints: x = 2  
B) System.out.printf("Value of %d + %d is %d%n", x, x, (x + x)); // prints: Value of 2 + 2 is 4  
C) System.out.printf("x ="); // prints: x =  
D) System.out.printf("%d = %d%n", (x + y), (y + x)); // prints: 5 = 5

## Exercise 2.11 (Which statements modify variables?)

A) p = i + j + k + 7; // modifies p (assignment)  
B) System.out.println("variables whose values are modified"); // does not modify program variables  
C) System.out.println("a = 5"); // does not modify variables (prints a string)  
D) value = input.nextInt(); // modifies value (assigns input)

## Exercise 2.12 (y = a x^3 + 7)

A) y = a \* x \* x \* x + 7; // CORRECT  
B) y = a \* x \* x \* (x + 7); // INCORRECT (expands to a\*x^3 + 7\*a\*x^2)  
C) y = (a \* x) \* x \* (x + 7); // INCORRECT (same problem as b)  
D) y = (a \* x) \* x \* x + 7; // CORRECT (equivalent to a\*x\*x\*x + 7)  
E) y = a \* (x \* x \* x) + 7; // CORRECT  
F) y = a \* x \* (x \* x + 7); // INCORRECT

## Exercise 2.13 (Order of evaluation and values)

A) x = 7 + 3 \* 6 / 2 - 1;  
 Evaluate: 3 \* 6 = 18; 18 / 2 = 9; so x = 7 + 9 - 1 = 15.  
B) x = 2 % 2 + 2 \* 2 - 2 / 2;  
 Evaluate: 2 % 2 = 0; 2 \* 2 = 4; 2 / 2 = 1; so x = 0 + 4 - 1 = 3.  
C) x = (3 \* 9 \* (3 + (9 \* 3 / (3))));  
 Innermost: 9 \* 3 = 27; 27 / 3 = 9; then (3 + 9) = 12; then 3 \* 9 \* 12 = 324.  
 So x = 324.

## Exercise 2.14 (Display numbers 1 to 4)

A) One println: System.out.println("1 2 3 4");  
B) Four print statements:  
 System.out.print("1 ");  
 System.out.print("2 ");  
 System.out.print("3 ");  
 System.out.println("4");  
C) One printf: System.out.printf("%d %d %d %d%n", 1, 2, 3, 4);

## Exercise 2.15 (Arithmetic program)

Sample solution (outline):  
import java.util.Scanner;  
Scanner input = new Scanner(System.in);  
System.out.print("Enter first integer: ");  
int a = input.nextInt();  
System.out.print("Enter second integer: ");  
int b = input.nextInt();  
System.out.printf("Sum = %d%nProduct = %d%nDifference = %d%nQuotient = %d%n", a + b, a \* b, a - b, a / b);  
// Note: integer division truncates; for floating-point quotient use double.

## Exercise 2.16 (Compare two integers)

Outline:  
System.out.print("Enter two integers: ");  
int a = input.nextInt();  
int b = input.nextInt();  
if (a > b) System.out.printf("%d is larger%n", a);  
else if (b > a) System.out.printf("%d is larger%n", b);  
else System.out.println("These numbers are equal");

## Exercise 2.17 (Sum, average, product, smallest, largest)

Outline approach:  
Read three ints a,b,c.  
sum = a + b + c;  
average = (a + b + c) / 3; // integer average as requested  
product = a \* b \* c;  
smallest = Math.min(a, Math.min(b, c));  
largest = Math.max(a, Math.max(b, c));  
Print results using printf.

## Exercise 2.18 (Displaying shapes with asterisks)

Print fixed strings to display shapes. For example, a box can be several println statements like:  
System.out.println("\*\*\*\*\*");  
System.out.println("\* \*");  
System.out.println("\* \*");  
System.out.println("\*\*\*\*\*");  
Produce similar groups for oval, arrow, diamond using appropriate lines of '\*'.

## Exercise 2.19 to 2.23 (What code prints)

2.19 prints:  
\*  
\*\*  
\*\*\*  
\*\*\*\*  
\*\*\*\*\*  
  
2.20 prints (each on its own line):  
\*  
\*\*\*  
\*\*\*\*\*  
\*\*\*\*  
\*\*  
  
2.21 prints a single line with concatenated strings from prints and a newline at the end — result is 15 stars: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
2.22 and 2.23 produce the patterns as asked; 2.23 prints three lines: \*, \*\*\*, \*\*\*\*\*.

## Exercise 2.24 (Largest and smallest of five integers)

Approach:  
Read five ints a,b,c,d,e.  
Initialize smallest = largest = a;  
For each of b-e, update:  
if (value < smallest) smallest = value;  
if (value > largest) largest = value;  
Print results.

## Exercise 2.25 (Odd or Even)

Use remainder operator:  
int n = input.nextInt();  
if (n % 2 == 0) System.out.println("Even");  
else System.out.println("Odd");

## Exercise 2.26 (Multiples)

Read two ints a and b.  
If (b != 0 && a % b == 0) print "a is a multiple of b"; else print "a is not a multiple of b".

## Exercise 2.27 (Checkerboard pattern)

Print lines that create a checkerboard pattern. For example, 8x8:  
System.out.println("\* \* \* \* ");  
System.out.println(" \* \* \* \*");  
... repeat alternating rows.

## Exercise 2.28 (Diameter, circumference, area)

Given radius r (int):  
System.out.printf("Diameter = %d%n", 2 \* r);  
System.out.printf("Circumference = %f%n", 2 \* 3.14159 \* r);  
System.out.printf("Area = %f%n", 3.14159 \* r \* r);  
// Or use Math.PI for more precision.

## Exercise 2.29 (Integer value of characters)

Example outputs using cast:  
System.out.printf("The character %c has the value %d%n", 'A', (int) 'A');  
// For required list: A B C a b c 0 1 2 $ \* + / and blank, use similar printf lines.

## Exercise 2.30 (Separating digits of a 5-digit integer)

Given int number (assume 5 digits):  
int d1 = number / 10000;  
int d2 = (number / 1000) % 10;  
int d3 = (number / 100) % 10;  
int d4 = (number / 10) % 10;  
int d5 = number % 10;  
System.out.printf("%d %d %d %d %d%n", d1, d2, d3, d4, d5);  
If number has more or fewer digits, the extraction won't match assumptions.

## Exercise 2.31 (Table of squares and cubes)

Loop from 0 to 10 and compute i\*i and i\*i\*i. Example:  
System.out.printf("%s%8s%8s%n", "Number", "Square", "Cube");  
for (int i = 0; i <= 10; i++) {  
 System.out.printf("%6d%8d%8d%n", i, i \* i, i \* i \* i);  
}

## Exercise 2.32 (Count negatives, positives, zeros)

Read five numbers and maintain three counters negCount, posCount, zeroCount.  
For each value v:  
if (v < 0) negCount++;  
else if (v > 0) posCount++;  
else zeroCount++;  
Print counts.

## Exercise 2.33 (BMI Calculator)

BMI formulas:  
- using pounds & inches: BMI = (weightInPounds \* 703) / (heightInInches \* heightInInches)  
- using kg & meters: BMI = weightKg / (heightM \* heightM)  
Read inputs; compute (use double for fractional results) and print. Also print classification per NIH:  
Underweight: BMI < 18.5  
Normal: 18.5 <= BMI < 25  
Overweight: 25 <= BMI < 30  
Obese: BMI >= 30

## Exercise 2.34 (World Population Growth Calculator)

This exercise asks to use the web to determine current world population and annual growth rate.  
Approach (once values are known): let P0 = current population, r = growth rate (decimal).  
After n years: estimate Pn = P0 \* (1 + r)^n  
Print estimates for years 1..5.

## Exercise 2.35 (Car-pool savings calculator)

Inputs: total miles per day (m), cost per gallon (c), miles per gallon (mpg), parking fees (p), tolls (t).  
Daily fuel cost = (m / mpg) \* c  
Daily total cost = daily fuel cost + p + t  
Compare with carpool cost estimated (fuel divided among riders, shared parking, etc.) to compute savings.