

## Answers to review questions from Chapter 1

1. When you write a Java program, do you prepare a source file or a class file?

**A source file**

2. What characters are used to mark comments in a Java program?

`/* ... */`

3. What is the significance of each component in the package declaration

```
package edu.stanford.cs.javacs2.ch1;
```

**edu indicates the educational domain**

**stanford indicates Stanford University**

**cs indicates the Computer Science department**

**javacs1 indicates that this package goes with the Java book for CS1**

**ch1 indicates that this package is associated with Chapter 1**

4. How would you define a constant called `CENTIMETERS_PER_INCH` with the value 2.54?

```
public static final double CENTIMETERS_PER_INCH = 2.54;
```

5. What is the name of the method that must be defined in every Java program?

**A public static void method named `main`**

6. What is meant by the term *boilerplate*? What is the boilerplate form of the `main` method used throughout this book?

***Boilerplate* refers to idiomatic programming patterns that are repeated in different programs but have no real significance for the programmer. The boilerplate form of `main` is**

```
public static void main(String[] args) {  
    new classname().run();  
}
```

7. Indicate which of the following are legal variable names in Java:

a. <code>x</code>	Legal
b. <code>formula1</code>	Legal
c. <code>average_rainfall</code>	Legal
d. <code>%correct</code>	Illegal (begins with %)
e. <code>short</code>	Illegal (reserved word)
f. <code>tiny</code>	Legal
g. <code>total output</code>	Illegal (contains a space)
h. <code>aVeryLongVariableName</code>	Legal
i. <code>12MonthTotal</code>	Illegal (begins with a digit)
j. <code>marginal-cost</code>	Illegal (contains a hyphen)
k. <code>b4hand</code>	Legal (but perhaps too clever)
l. <code>_stk_depth</code>	Legal

8. What are the two attributes that define a data type?

**Types are defined by a *domain* and a *set of operations*.**

9. What sizes does Java assign to the types **byte**, **short**, **int**, and **long**?

**byte = 8 bits**  
**short = 16 bits**  
**int = 32 bits**  
**long = 64 bits**

10. What coding system does Java use for characters?

**Unicode**

11. What does ASCII stand for? What relationship does the ASCII code have to the code used to represent characters in Java?

**American Standard Code for Information Interchange**  
**The ASCII code matches the first 256 characters in the Unicode standard.**

12. List all possible values of type **boolean**.

**false** and **true**

13. What statements would you include in a program to read a value from the user and store it in the variable **x**, which is declared as a **double**?

**You would first need to declare a scanner using**

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

**Later in the program you could read the value of **x** using lines like this:**

```
System.out.print("Enter x: ");  
double x = scanner.nextDouble();
```

14. Indicate the values and types of the following expressions:

- |               |                      |
|---------------|----------------------|
| a. $2 + 3$    | <b>5 (int)</b>       |
| b. $19 / 5$   | <b>3 (int)</b>       |
| c. $19.0 / 5$ | <b>3.8 (double)</b>  |
| d. $3 * 6.0$  | <b>18.0 (double)</b> |
| e. $19 \% 5$  | <b>4 (int)</b>       |
| f. $2 \% 7$   | <b>2 (int)</b>       |

15. What is the difference between the unary minus and the subtraction operator?

**The unary minus operator is written before a single operand and denotes negation, as in  $-x$ . The subtraction operator is written between two operands as in  $x - 2$ .**

16. Calculate the result of each of the following expressions:

- |  |           |
|--|-----------|
| a. $6 + 5 / 4 - 3$                                 | <b>4</b>  |
| b. $2 + 2 * (2 * 2 - 2) \% 2 / 2$                  | <b>2</b>  |
| c. $10 + 9 * ((8 + 7) \% 6) + 5 * 4 \% 3 * 2 + 1$  | <b>42</b> |
| d. $1 + 2 + (3 + 4) * ((5 * 6 \% 7 * 8) - 9) - 10$ | <b>42</b> |

17. What does the term *truncation* mean?

**Truncation** converts a floating-point number to an integer by throwing away any digits after the decimal point. Thus, the floating-point number 3.9999 truncates to the integer 3.

18. What is a *type cast* and how do you indicate one in Java?

A **type cast** is used to indicate conversion of one type to another. In Java, type casts are written by enclosing the type name in parentheses before the value to be converted, as in the expression `(int) x`, which truncates `x` to an integer.

19. How do you specify a shorthand assignment operation?

**Shorthand assignment operators** are written by writing the operator in front of the equal sign used for assignment, as in

```
balance += deposit;
```

20. What is the difference between the expressions `++x` and `x++`?

Both expressions **increment** (add one to) the value of `x`. The difference is the value of the expression. The expression `++x` returns the incremented value; the expression `x++` returns the original value before the increment.

21. What is meant by *short-circuit evaluation*?

**Short-circuit evaluation** evaluates only as much of an expression as is necessary to determine the result. Java uses short-circuit evaluation to evaluate the logical operators `&&` and `||`. With `&&`, for example, there is no need to evaluate the right operand if the left operand is `true`.

22. Write out the general syntactic form for each of the following control statements: `if`, `switch`, `while`, and `for`.

```
if (condition) statement

if (condition) statement else statement

switch (e) {
    case c1:
        statements
        break;
    case c2:
        statements
        break;
    . . . more case clauses . . .
    default:
        statements
        break;
}

while (conditional-expression) {
    statements
}

for (init; test; step) {
    statements
}
```

23. Describe in English the operation of the **switch** statement, including the role of the **break** statement at the end of each **case** clause.

When the program executes a **switch** statement, it evaluates the control expression and compares it against the values  $c_1$ ,  $c_2$ , and so forth, each of which must be a constant. If one of the constants matches the value of the control expression, the statements in the associated **case** clause are executed. When the program reaches the **break** statement at the end of the clause, the operations specified by that clause are complete, and the program continues with the statement that follows the entire **switch** statement. If none of the expression match the value of the control expression, the program executes the statements in the **default** clause, if any.

24. What is a *sentinel*? What is the general form of the *read-until-sentinel* pattern?

A *sentinel* is a value used to indicate some kind of special processing and often signals the end of a list of input values.

25. What **for** loop control line would you use in each of the following situations?

- a. Counting from 1 to 100

```
for (int i = 1; i <= 100; i++)
```

- b. Counting by sevens starting at 0 until the number has more than two digits

```
for (int i = 0; i < 100; i += 7)
```

- c. Counting backward by twos from 100 to 0

```
for (int i = 100; i >= 0; i -= 2)
```

26. Define each of the following terms: *class*, *object*, *method*, *instance variable*, *subclass*, and *superclass*.

A *class* is a template for data objects that share a common representation and set of operations.

An *object* is a data value that is an instance of a particular class.

A *method* is a sequence of program steps that have been collected together and given a name.

An *instance variable* is a particular data value stored within an object.

A *subclass* is a class that extends the behavior of an existing class, which becomes its *superclass*.

27. True or false: In Java, there can be many objects that are instances of a class.

True.

28. In Figure 1-12, what subclasses are shown for the class *Arthropoda*?

The direct subclasses are *Crustacea*, *Insecta*, and *Arachnida*. The descendents of *Insecta* (*Hymenoptera*, *Formicidae*, *Lasius*, and *niger*) are indirect subclasses of *Arthropoda*.

29. What two features does this chapter identify as the most important aspects of the object-oriented programming model?

Encapsulation and inheritance.