Chapter 6 Methods: A Deeper Look

Section 6.2 Program Modules in Java

6.2 Q1: Information is passed to a method in ________.
a. the method name
b. that method's return
c. the method body
d. the arguments to the method
ANS: d. the arguments to the method.

6.2 Q2: A well-designed method ________.
a. performs multiple unrelated tasks
b. repeats code found in other methods
c. contains thousands of lines of code
d. performs a single, well-defined task
ANS: d. performs a single, well-defined task.

Section 6.3 `static` Methods, `static` Fields and Class `Math`

6.3 Q1: To declare a method as static, place the keyword `static` before ________ in the method's declaration.
a. the method modifier
b. the return type
c. the method name
d. the argument list
ANS: b. the return type

6.3 Q2: Which is a correct `static` method call of `Math` class method `sqrt`?
a. `sqrt(900);`
b. `math.sqrt(900);`
c. `Math.sqrt(900);`
d. `Math math = new Math();`
`math.sqrt(900);`
ANS: c. `Math.sqrt(900);`

6.3 Q3: Which of the following methods is *not* in the `Math` class?
a. `ceil`
b. `abs`
c. `parseInt`
d. `log`
ANS: c. `parseInt`

6.3 Q4: Which of the following can be an argument to a method?
a. Constants.
b. Variables.
c. Expressions.
d. All of the above.
ANS: d. All of the above.

6.3 Q5: Method `log` takes the logarithm of its argument with respect to what base?

a. 10
b. e
c. 2
d. pi
ANS: b. e

`Math` Class Constants `PI` and `E`

6.3 Q6: Any field declared with keyword ________ is constant.
a. `static`
b. `const`
c. `constant`
d. `final`
ANS: d. `final`

Why Is Method `main` Declared `static`?

6.3 Q7: Declaring `main` as `static` allows the JVM to invoke `main` ________.
a. without knowing the name of the class in which `main` is declared.
b. by creating an object of the class in which `main` is declared.
c. without creating an instance of the class in which `main` is declared.
d. None of the above.
ANS: c. without creating an instance of the class in which `main` is declared.

Section 6.4 Declaring Methods with Multiple Parameters

6.4 Q1: Variables should be declared as *fields* only if ________.
a. they are local variables
b. they are used only within a method
c. they are required for use in more than one method or their values must be saved between calls to the class's methods
d. they are arguments
ANS: c. they are required for use in more than one method or their values must be saved between calls to the class's methods

6.4 Q2: Consider the following Java statements:

```
int x = 9;
double y = 5.3;
result = calculateValue(x, y);
```

Which of the following statements is *false*?
a. A method is called with its name and parentheses.
b. `x` and `y` are parameters.
c. Copies of `x` and `y` are passed to the method `calculateValue`.
d. `x` and `y` are arguments.
ANS: b. `x` and `y` are paramters.

6.4 Q3: The parameter list in the method header and the arguments in the method call must agree in:
a. number
b. type
c. order

d. all of the above
ANS: d. all of the above

Assembling Strings with String Concatenation

6.4 Q4: Which operator can be used in string concatenation?
a. `*`
b. `+=`
c. `++`
d. `=+`
ANS: b. `+=`

6.4 Q5: When an object is concatenated with a `String`, ________.
a. a compilation error occurs
b. a runtime error occurs
c. the object's `toString` method is implicitly called to obtain the `String` representation of the object
d. the object's class name is concatenated with the `String`
ANS: c. the object's `toString` method is implicitly called to obtain the `String` representation of the object.

Section 6.5 Notes on Declaring and Using Methods

6.5 Q1: A `static` method can ________.
a. call only other `static` methods of the same class directly
b. manipulate only `static` fields in the same class directly
c. be called using the class name and a dot (`.`)
d. All of the above.
ANS: d. All of the above.

6.5 Q2: Which of the following statements is *false*?
a. If a method does *not* return a value, the *return-value-type* in the method declaration can be omitted.
b. Placing a semicolon after the right parenthesis enclosing the parameter list of a method declaration is a syntax error.
c. Redeclaring a method parameter as a local variable in the method's body is a compilation error.
d. Forgetting to return a value from a method that should return a value is a compilation error.
ANS: a. If a method does not return a value, the *return-value-type* in the method declaration can be omitted. In this case the *return-value-type* must be declared `void`.

Section 6.6 Method Call Stack and Stack Frames

6.6 Q1: Stacks are known as ________ data structures.
a. FIFO.
b. FILO.
c. LIFO.
d. LILO.
ANS: c. LIFO.

6.6 Q2: If more method calls occur than can have their activation records stored on the program execution stack, an error known as a ________ occurs.
a. stack overflow.
b. stack rewind.
c. stack full.
d. stack empty.

ANS: a. stack overflow.

Section 6.7 Argument Promotion and Casting

6.7 Q1: Which of the following promotions of primitive types is *not* allowed to occur?
a. `char` to `int`.
b. `double` to `float`.
c. `int` to `double`.
d. `short` to `long`.
ANS: b. `double` to `float`.

6.7 Q2: Which of the following primitive types is *never* promoted to another primitive type?
a. `double`.
b. `byte`.
c. `boolean`.
d. Both a and c.
ANS: d. Both a and c.

Section 6.8 Java API Packages

6.8 Q1: Which of the following statements is *false*?
a. The Java API consists of packages.
b. The Java API helps programmers avoid "reinventing the wheel."
c. The Java API consists of `import` declarations.
d. None of the above.
ANS: c. The Java API consists of `import` declarations. (The Java API is built from packages.)

6.8 Q2: Which of the following is *not* a package in the Java API?
a. `java.component`
b. `java.awt`
c. `javax.swing.event`
d. `java.lang`
ANS: a. `java.component`

6.8 Q3: The `java.text` package contains classes for manipulating all of the following items *except* ________.
a. classes
b. numbers
c. strings
d. characters
ANS: a. classes

Section 6.9 Case Study: Random-Number Generation

6.9 Q1: `Math` `static` method `random` generates a random double value in the range from 0.0
a. up to but not including 1.0
b. up to and including 1.0
c. up to and including 100.0
d. up to but not including 100.0
ANS: a. up to but not including 1.0

6.9 Q2: Which statement below could be used to simulate the outputs of tossing a quarter to get heads or tails? Suppose `randomNumbers` is a `SecureRandom` object.
a. `randomNumbers.nextInt(7);`
b. `randomNumbers.nextInt(2);`
c. `randomNumbers.nextInt(1);`

d. `randomNumbers.nextInt(25);`
ANS: b. `randomNumbers.nextInt(2);`

Rolling a Six-Sided Die

6.9 Q3: Which statement below could be used to simulate the outputs of rolling a six-sided die? Suppose `randomNumbers` is a `SecureRandom` object.
a. `1 + randomNumbers.nextInt(6);`
b. `1 + randomNumbers.nextInt(2);`
c. `6 + randomNumbers.nextInt(1);`
d. `3 + randomNumbers.nextInt(3);`
ANS: a. `1 + randomNumbers.nextInt(6);`

6.9 Q4: Which statement creates a random value from the sequence 2, 5, 8, 11 and 14. Suppose `randomNumbers` is a `SecureRandom` object.
a. `2 + 5 * randomNumbers.nextInt(3);`
b. `3 + 2 * randomNumbers.nextInt(5);`
c. `5 + 3 * randomNumbers.nextInt(2);`
d. `2 + 3 * randomNumbers.nextInt(5);`
ANS: d. `2 + 3 * randomNumbers.nextInt(5);`

Section 6.10 Case Study: A Game of Chance; Introducing `enum` Types)

6.10 Q1: A set of named constants that start with the value 0 for the first constant and increment by 1 for each subsequent constant can be declared as a(n) ________.
a. `class`
b. `enum`
c. `enumeration`
d. None of the above.
ANS: b. `enum`.

6.10 Q2: The identifiers in an enumeration ________.
a. must be unique.
b. may be duplicated.
c. must be lowercase letters and cannot contain numbers.
d. must be uppercase letters and cannot contain numbers.
ANS: a. must be unique.

Section 6.11 Scope of Declarations

6.11 Q1: Identifiers in Java have ________ and ________ scopes?
a. method, class.
b. class, block.
c. block, statement.
d. statement, file.
ANS: b. class, block.

6.11 Q2: Which of the following statements describes *block scope*?
a. It begins at the opening `{` of the class declaration and terminates at the closing `}`.
b. It limits label scope to only the method in which it is declared.
c. It begins at the identifier's declaration and ends at the terminating right brace (`}`).
d. It is valid for one statement only.

ANS: c. It begins at the identifier's declaration and ends at the terminating right brace (}).

6.11 Q3: Which of these statements best defines *scope*?
a. Scope refers to the classes that have access to a variable.
b. Scope determines whether a variable’s value can be altered.
c. Scope is the portion of a program that can refer to an entity by its simple name.
d. Scope allows the programmer to use a class without using its fully qualified name.
ANS: c. Scope is the portion of a program that can refer to an entity by its simple name.

Section 6.12 Method Overloading

6.12 Q1: Overloaded methods always have the same _________.
a. method name
b. return type
c. number of parameters
d. order of the parameters
ANS: a. method name

6.12 Q2: An overloaded method is one that ________.
a. has a different name than another method, but the same parameters
b. has the same name as another method, but different parameters (by number, types or order of the types)
c. has the same name and parameters as a method defined in another class
d. has the same name and parameters, but a different return type as another method
ANS: b. has the same name as another method, but different parameters (by number, types or order of the types)

Declaring Overloaded Methods

6.12 Q3: Which of the following methods are overloaded with respect to one another?

```
public int max (int a, int b) { … }
public double max (double a, double b) { … }
public int max (int a, int b, int c) { … }
public double max (double a, double b, double c) { … }
```

a. A and B are overloaded; C and D are overloaded.
b. A and C are overloaded; B and D are overloaded.
c. A, B and C are overloaded.
d. All four methods are overloaded.
ANS: d. All four methods are overloaded.

Distinguishing Between Overloaded Methods

6.12 Q4: A Java class can have which of the following methods?

```
A.  void foo(int a)
B.  void foo(int a, int b)
C.  void foo(double a)
D.  void foo(double a, double b)
E.  void foo(int b)
```

a. All of the above.
b. A, B, D, E.
c. A, B, C, D.
d. A, C, D, E.
ANS: c. A, B, C, D.

Return Types of Overloaded Methods

6.12 Q5: Method calls cannot be distinguished by ________.
a. method name
b. return type
c. parameter lists
d. method signature
ANS: b. return type.

6.12 Q6: In a class containing methods with the same name, the methods are distinguished by ________.
a. Number of arguments
b. Types of arguments
c. Return type
d. (a) and (b)
e. (b) and (c)
ANS: d. (a) and (b).

Section 6.13 (Optional) GUI and Graphics Case Study: Colors and Filled Shapes

6.13 Q1: Java uses class ________ to represent colors using their RGB values.
a. `Color`
b. `Colors`
c. `RGBColor`
d. `RGBColors`
ANS: a. `Color`

6.13 Q2: Filled rectangles and filled circles are drawn using `Graphics` method ________ and ________.
a. `fillRect`, `fillCircle`
b. `filledRect`, `filledCircle`
c. `fillRect`, `fillOval`
d. `filledRect`, `filledOval`
ANS: c. `fillRect`, `fillOval`