2

Programming with Java Statements

TABLE 2-1 Java Statements

Statement Name	Definition	On the Exam
The `assert` statement	Used to determine whether code is functioning as expected. When its expression is evaluated to false, an exception is thrown.	
The `break` statement	Used to exit the body of a `switch` statement or loop.	✓
The `case` statement	Used as part of the `switch` statement to execute statements when its value matches the `switch` statement's conditional value.	✓
The `continue` statement	Used to terminate the current iteration of a `do-while`, `while`, or `for` loop and continue with the next iteration.	✓
The `while` statement	Used for iteration based on a condition.	✓
The `do-while` statement	Used for iteration based on a condition. The body of the `do-while` statement is executed at least once.	✓
The `empty` statement	Used for trivial purposes where no functionality is needed. It is represented by a single semicolon.	
The expression statements	Used to evaluate expressions. See Table 2-2.	✓
The `for` loop statement	Used for iteration. Main components are an initialization part, an expression part, and an update part.	✓
The enhanced `for` loop statement	Used for iteration through an iterable object or array.	✓
The `if` statement	Used for the conditional execution of statements.	✓
The `if-then` statement	Used for the conditional execution of statements by providing multiple conditions.	✓
The `if-then-else` statement	Used for the conditional execution of statements by providing multiple conditions and fall-through when no conditions are met.	✓

TABLE 2-1 Java Statements

Statement Name	Definition	On the Exam
The labeled statement	Used to give a statement a prefixed label.	
The `return` statement	Used to exit a method and return a specified value.	✓
The `switch` statement	Used for branching code based on conditions.	✓
The `synchronized` statement	Used for access control of threads.	
The `throw` statement	Used to throw an exception.	Chapter 9
The `try-catch-finally` statement	Used for exception handling.	Chapter 9

Expression statements are used for the evaluation of expressions. The assignment expression statements allow assignments to be performed on variables.

Conditional statements, also known as decision statements, assist in directing the flow of control when a decision needs to be made. Conditional statements include the if, if-then, if-then-else, and switch statements.

Iteration statements provide support in looping through blocks of code. Iteration statements include the for loop, the enhanced for loop, the while statement, and the do-while statement.

Transfer of control statements provide a means of stopping or interrupting the normal flow of control. Transfer of control statements include the continue, break, and return statements. Transfer of control statements are always seen within other types of statements.

Understand Assignment Statements

An assignment statement sets a value within a variable. All assignment statements are considered to be expression statements. Expressions in Java are anything that has a value or is reduced to a value. All expressions can be used as statements; the only requirement is that they end with a semicolon.

TABLE 2-2

Expression Statements

Expression Statement	Expression Statement Example	Coverage
Assignment	`variableName = 7;`	Chapter 2
Prefix-increment	`++variableName;`	Chapter 3
Prefix-decrement	`--variableName;`	Chapter 3
Postfix-increment	`variableName++;`	Chapter 3
Postfix-decrement	`variableName--;`	Chapter 3
Method invocation	`performMethod();`	Chapter 5
Object creation	`new ClassName();`	Chapter 4

The Assignment Expression Statement

Assignment statements, are designed to assign values to variables. All assignment statements must be terminated with a semicolon.

variable = *value*;

On the left is the variable that will be associated with the memory and type necessary to store the value. On the right is a literal value. If an expression is on the right, such as (1+2), it must be evaluated down to its literal value before it can be assigned. Lastly, an equal sign resides between the variable and value of an assignment statement.

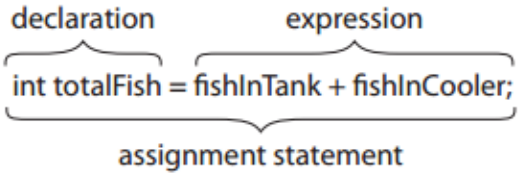
```
int variableName; // Declaration of an integer

variableName = 100; // Assignment expression statement
```

Trying to save an invalid literal to a declared primitive type variable will result in a compiler error. For example, the compilation error Exception in thread "xxxx" java.lang.RuntimeException: Uncompilable source code - incompatible types…

FIGURE 2-1

Combined statements



Create and Use Conditional Statements

Conditional statements are used when there is a need for determining the direction of flow based on conditions. Conditional statements include the if, if-then, if-then-else, and switch statements.

TABLE 2-3 Conditional Statements

Formal Name	Keywords	Expression Types	Example
`if`	`if`, `else` (optional)	`boolean`	`if (value == 0) {}`

`if-then`	`if`,
	`else if`,
	`else if` (optional)
		`boolean`	`if (value == 0) {}`
			`else if (value == 1) {}`
			`else if (value >= 2) {}`

`if-then-else`	`if`,
	`else if` (optional),
	`else if` (optional),
	`else`
		`boolean`	`if (value == 0) {}`
			`else if (value >=1) {}`
			`else {}`

Ternary operator	`?`, `:` (not official Java keyword)	`boolean`	`minVal = a < b ? a : b;`

`switch`	`switch`,
	`case`,
	`default` (optional),
	`break` (optional)
		`char`, `byte`, `short`, `int`, `Character`, `Byte`, `Short`, `String`, `Integer`, `enumeration` types	`switch (100) {`
			`  case 100: break;`
			`  case 200: break;`
			`  case 300: break;`
			`  default: break;`
			`}`

INSIDE THE EXAM

The if, if-then, and if-then-else Statements

The distinction between the `if`, `if-then`, and `if-then-else` statements may seem blurred. This is partially because the `then` keyword used in some other programming languages is not used in Java, even though the Java constructs are formally known as `if-then` and `if-then-else`. Let's clarify some confusing points about the `if`-related statements by providing some facts.

- The `if` statement allows for the optional use of the `else` branch. This may be a little confusing since you may expect the `if` statement to stand alone without any branches.
- The `if-then` statement must have at least one `else if` branch. Optionally, an unlimited number of `else if` branches may be included. You cannot use an `else` statement in an `if-then` statement or the statement would be considered an `if-then-else` statement.
- The `if-then-else` statement must have at least one `else if` branch. The `else if` branch is not optional, because if it were not present, the statement would be considered to be an `if` statement that includes the optional `else` branch.

The if Conditional Statement

The if statement is designed to conditionally execute a statement or conditionally decide between a choice of statements. The if statement will execute only one statement upon the condition unless braces are supplied. Braces, also known as curly brackets, allow for multiple enclosed statements to be executed. This group of statements is also known as a *block*. The expression that is evaluated within if statements must evaluate to a boolean value or the application will not compile. The else clause is optional and may be omitted.

```
if (expression)
        statementA;
else
        statementB;
```

Even though relational operators (such as >=) are commonly used, assignment statements are always allowed.

```
if (bValue = false)
        System.out.println("TRUE");
else
```

```
System.out.println("FALSE")
```

The assignment statements of all primitives will return their primitive values:

```
if (i=1) {}; // will not compile
```

The if-then Conditional Statement

The if-then conditional statement is used when multiple conditions need to flow through a decision-based scenario.

```
if (expressionA)
        statementA;
else
        if (expressionB)
                statementB;
```

The if-then-else Conditional Statement

As with the if and if-then statements, all expressions must evaluate to true or false as the expected primitive type is boolean. The main difference in the if-then-else statement is that the code will fall through to the final stand-alone else when the expression fails to return true for any condition. Each statement may optionally be a group of statements enclosed in braces. There is no limit to the number of else if clauses.

```
if (expressionA)
        statementA;
else if (expressionB)
        statementB;
else if (expressionC)
        statementC;
...
else
        statementZZ;
```

The Ternary Operator

The ternary operator is a variation of the if-then-else statement. It is also sometimes referred to as a conditional operator. The ternary operator derives its name from the fact that it is the only operator to use three operands. The ? and : characters are used in this operation.

The ternary operator behaves similarly to the if-then-else statement but never includes any optional else if. The first part must be an expression that results in a boolean value. In this case,

testCondition is tested to determine if it is greater than 0. If this expression is true, the ternary operation returns value1, the first value after the ? character. A false will result in value2, the value after the : character, to be returned.

```
result = testCondition ? value1 : value2
```

Ternary operators are great for checking and returning simple values. However, in a more complex situation, a normal if-then-else statement will often result in code that is easier to read.

The switch Conditional Statement

The switch conditional statement is used to match the value from a switch statement expression against a value associated with a case keyword. Once matched, the enclosed statement(s) associated with the matching case value are executed and subsequent case statements are executed, unless a break statement is encountered. The break statements are optional and will cause the immediate termination of the switch conditional statement.

When two case statements within the same switch statement have the same value, a compiler error will be thrown (Compiler error, Error: duplicate case label).

The expression of the switch statement must evaluate to byte, short, int, or char. Wrapper classes of type Byte, Short, Integer, and Character are also allowed because they are automatically unboxed to primitive types. Enumerated types (that is, enum) are permitted as well. Additionally, Java SE 7 added support for evaluation of the String object in the expression.

```
switch (expression) {
        case valueA:
                // Sequences of statements
                break;
        case valueB:
                // Sequences of statements
                break;
        default:
                // Sequences of statements
                ...
}
```

Remember that without break statements, the switch block will continue with its fall-through from the point that the condition has been met.

The default case is often listed last for code readability. Default can be placed anywhere in switch. If the default is placed anywhere but in the end, the cases will be first evaluated. If the case matching the value is after the default statement, the case will be executed. But in case none of the cases match the value, the default will be executed and if the “break” is missing, all cases that precedes it will be executed.

SCENARIO & SOLUTION	
To ensure that your statement is bug-free, which type of statements should you include within the switch?	Both `break` statements and the `default` statement are commonly used in the switch. Forgetting these statements can lead to improper fall-throughs or unhandled conditions. Note that many bug-finding tools will flag missing `default` statements.
You want to use a range in a `case` statement (for instance, `case 7-35`). Is this a valid feature in Java, as it is with other languages?	Ranges in `case` statements are *not* allowed. Consider setting up a condition in an `if` statement. For example: `if (x >=7 && x <=35){}`
You want to use the `switch` statement, using `String` values where the expression is expected, as is possible with other languages. Is this a valid feature in Java?	Strings are not valid at the decision point for `switch` statements prior to Java SE 7. For Java SE 6 and earlier, consider using an `if` statement instead. For example: `if (strValue.equals("S1")){}`

Create and Use Iteration Statements

Iteration statements are used when there is a need to iterate through pieces of code. Iteration statements include the for loop, enhanced for loop, and the while and do-while statements. The break statement is used to exit the body of any iteration statement. The continue statement is used to terminate the current iteration and continue with the next iteration.

TABLE 2-4 Iteration Statements

Formal Name	Keywords	Main Expression Components	Example
`for` loop	`for`, `break` (optional), `continue` (optional)	Initializer, expression, update mechanism	`for (i=0; i<j; i++) {}`
Enhanced `for` loop	`for`, `break` (optional), `continue` (optional)	Element, array, or collection	`for (Fish f : listOfFish) {};`
`while`	`while`, `break` (optional), `continue` (optional)	Boolean expression	`while (value == 1) { }`
`do-while`	`do`, `while`, `break` (optional), `continue` (optional)	Boolean expression	`do { } while (value == 1);`

The for Loop Iteration Statement

It has main parts that include an initialization part, an expression part, and an iteration part. The initialization does not need to declare a variable as long as the variable is declared before the for statement. Be aware, though, that the scope of the variable declared within the initialization part of the for loop ends once the for loop terminates. The expression within the for loop statement

must evaluate to a boolean value. The iteration, also known as the update part, provides the mechanism that will allow the iteration to occur.

Here's the general usage of the for statement:

```
for ( initialization; expression; iteration) {
        // Sequence of statements
}
```

The Enhanced for Loop Iteration Statement

The enhanced for loop is used to iterate through an array, a collection, or an object that implements the interface iterable. The enhanced for loop is also commonly known as the for each loop and the for in loop. Iteration occurs for each element in the array or iterable class. Remember that the loop can be terminated at any time by the inclusion of a break statement. And as with the other iteration statements, the continue statement will terminate the current iteration and start with the next iteration.

Here's the general usage of the for statement:

```
for (type variable : collection) statement-sequence
```

The while Iteration Statement

The while statement is designed to iterate through code. The while loop statement evaluates an expression and executes the while loop body only if the expression evaluates to true. Typically, an expression within the body will affect the result of the expression.

Here's the general usage of the while statement:

```
while (expression) {
        // Sequences of statements
}
```

The do-while Iteration Statement

The do-while statement is designed to iterate through code. It is very similar to the while loop statement, except that it always executes the body at least once. The do-while loop evaluates an expression and continues to execute the body only if it evaluates to true. Typically, an expression within the body will affect the result of the expression.

Here’s the general usage of the do-while statement:

```
do {
        // Sequence of statements
}
while (expression)
```

Exercise 2-4

Table 2-5 represents all of the statement-related Java keywords. This exercise will enable you to use the table to assist in deducing the keywords you might see while using the various types of statements.

TABLE 2-5

Java Statement-Related Keywords

Java Keywords				
break	continue	else	if	throw
case	default	finally	return	try
catch	do	for	switch	while

1. List the primary keywords you may see in conditional statements:
    - If
    - Else
    - Switch
    - Case
    - Default.
2. List the primary keywords you may see in iteration statements:
    - For
    - While
    - Do.
3. List the primary keywords you may see in transfer of control statements:
    - Break
    - Continue
    - Return.
4. Bonus: List the primary keywords you may see in exception handling statements:
    - Try
    - Catch
    - Finally
    - Throw.

Create and Use Transfer of Control Statements

Transfer of control statements include the break, continue, and return statements. Transfer of control statements provide a means to stop or interrupt the normal flow of control. They are always used within other types of statements. A transfer of control statement always works with the labeled statement.

The break Transfer of Control Statement

The break statement is used to exit or force an abrupt termination of the body of the switch conditional statement as well as the body of the do, for loop; enhanced for loop; and while and do-while iteration statements.

The general usage of the break statement is simple:

break;

The continue Transfer of Control Statement

The continue statement is used to terminate the current iteration of a do, for loop; enhanced for loop; or while or do-while loop and continue with the next iteration.

The general usage of the continue statement is also simple:

continue;

The return Transfer of Control Statement

The return statement is used to exit a method and optionally return a specified value as an expression. A return statement with no return value must have the keyword void in the method declaration.

Here's the general usage of the return statement:

return [expression];

If a return statement is the last statement in a method, and the method doesn't return anything, the return statement is optional. In this optional case, the return statement is typically not used.

The labeled Statement

The labeled statement is used to give a statement a prefixed label. It can be used in conjunction with the continue and break statements. Use labeled statements sparingly; you should se them over other approaches only on a few occasions.

The general usage of the labeled statement is the addition of a label followed by a colon with the appropriate statement immediately following it:

labelIdentifier:

Statement (such as a for loop)

Here are the general usages of the break and continue statements in conjunction with the labeled statement:

break *labelIdentifier*;

and

continue *labelIdentifier*;

CERTIFICATION SUMMARY

- Expression statements, with a focus on the assignment statement
- Conditional statements (if, if-then, if-then-else, and switch)
- Iteration statements (for, enhanced for, while, and do-while)
- Transfer of control statements (continue, break, and return)

TWO-MINUTE DRILL

Understand Assignment Statements

- Assignment statements assign values to variables.
- Assignment statements that do not return boolean types will cause the compiler to report an error when used as the expression in an if statement.
- Trying to save an invalid literal to a declared primitive type variable will result in a compiler error.

Create and Use Conditional Statements

- Conditional statements are used for determining the direction of flow based on conditions.
- Types of conditional statements include the if, if-then, if-then-else, and switch statements.

- The ternary operator is another form of the if-then-else statement.
- The default case statement can be placed anywhere in the body of the switch statement.
- The expressions used in if statements must evaluate to boolean values, or the application will fail to compile.
- Boolean wrapper classes are allowed as expressions in if statements because they are unboxed. Remember that unboxing is the automatic production of primitive values from their related wrapper classes when the primitive value is required.

Create and Use Iteration Statements

- Iteration statements are designed for iterating over pieces of code.
- Iteration statements include the for loop, enhanced for loop, and the while and do-while statements.
- The for loop statement has main components that include an initialization part, an expression part, and an update part.
- The enhanced for loop statement is used for iteration through an iterable object or array.
- The while loop statement is used for iteration based on a condition.
- The do-while statement is used for iteration based on a condition. The body of this statement is always executed at least once.

Create and Use Transfer of Control Statements

- Transfer of control statements interrupt or stop the flow of execution.
- Transfer of control statements include continue, break, and return statements.
- The continue statement is used to terminate the current iteration of a do, for loop; enhanced for loop; and while or do-while loop and continue with the next iteration.
- The break statement is used to exit or force an abrupt termination of the body of the switch conditional statement as well as the body of the do, for loop, enhanced for loop, and while and do-while iteration statements.
- The return statement is used to exit a method and may return a specified value.
- The labeled statement is used to give a statement a prefixed label. It is used with the continue and break statements.
- A block is a sequence of statements within braces—for example, { int x=0; int y=1 }.