# Flow Control and Exceptions – Answers

Question

d

The program will display the letter b when run. The second if statement is evaluated since the boolean expression of the first if statement is true. The else clause belongs to the second if statement. Since the boolean expression of the second if statement is false, the if block is skipped and the else clause is executed.

Question

c Prints: v w x x y z z

Cases 1 and 3 have no break statement, so the next case is also executed and x and z are printed twice.

Question

**d Prints: D**

The expression, b = false, appears to be testing the value of b, but it is really setting the value of b. Always look carefully at the boolean expression of an if statement to verify that the expected equality operator (==) has not been replaced by the simple assignment operator (=).

Question

d Run-time error

The index, i, is incremented before the array access expression is evaluated, so the first element accessed is at index one instead of zero. The arguments, BCDEF, are printed before an ArrayIndexOutOfBoundsException is thrown when the attempt is made to access an element beyond the end of the array.

Question

**f**

There is nothing wrong with the code. The case and default labels do not have to be specified in any specific order. The use of the break statement is not mandatory, and without it the control flow will simply fall through the labels of the switch statement.

Question

b Prints: 1212013

A method local variable, i, is declared at line 3. At line 4, the initialization part of the for statement initializes the method local variable to the value zero. The for statement does not declare a new variable, i, so the method local variable is not shadowed within the loop. Suppose a local variable, v, is declared within a statement or block. Variable, v, can shadow a class variable or an instance variable of the same name, but a compile-time error is generated if v shadows a method local variable. Please note that a compile-time error is not generated if a local variable is shadowed within the declaration of a class that is nested within the scope of the local variable.

Question

b and e

If run with one argument, the program will print the given argument followed by "The end". The finally block will always be executed, no matter how control leaves the try block.

Question

d Prints: true,true

Both Error and Exception are subclasses of Throwable.

Question

a d class A extends Object. Compile-time error at 3.

The constructors for class B and class C both invoke the constructor for A. The constructor for class A declares Exception in the throws clause. Since the constructors for B and C invoke the constructor for A, it is necessary to declare Exception in the throws clauses of B and C. A compile-time error is generated at marker 3, because the constructor does not declare Exception in the throws clause.

Question

c

Only (c) contains a valid for loop. The initializer in a for statement can contain either declarations or a list of expression statements, but not both as attempted in (a). The loop condition must be of type boolean. (b) tries to use an assignment of an int value (notice the use of = rather than ==) as a loop condition and is, therefore, not valid. The loop condition in the for loop (d) tries to use the uninitialized variable i, and the for loop in (e) is simply syntactically invalid.

Question

b Prints: 0,0,1,1,0,1

The nested catch clause is able to catch a Level2Exception or any subclass of it. The switch statement throws a Level1Exception that can not be caught by the nested catch clause; so the nested finally block is executed as control passes to the first of the two outer catch clauses. The outer finally block is executed as control passes out of the try statement.

Question

**e**

The loop body is executed twice and the program will print 3. The first time the loop is executed, the variable i changes from 1 to 2 and the variable b changes from false to true. Then the loop condition is evaluated. Since b is true, the loop body is executed again. This time the variable i changes from 2 to 3 and the variable b changes from TRue to false. The loop condition is now evaluated again. Since b is now false, the loop terminates and the current value of i is printed.

Question 13

a and d

"i=1, j=0" and "i=2, j=1" are part of the output. The variable i iterates through the values 0, 1, and 2 in the outer loop, while j toggles between the values 0 and 1 in the inner loop. If the values of i and j are equal, the printing of the values is skipped and the execution continues with the next iteration of the outer loop. The following can be deduced when the program is run: variables i and j are both 0 and the execution continues with the next iteration of the outer loop. "i=1, j=0" is printed and the next iteration of the inner loop starts. Variables i and j are both 1 and the execution continues with the next iteration of the outer loop. "i=2, j=0" is printed and the next iteration of the inner loop starts. "i=2, j=1" is printed, j is incremented, j < 2 fails, and the inner loop ends. Variable i is incremented, i < 3 fails, and the outer loop ends.

Question 14

f Compile-time error

Variables declared inside of a block or method are called local variables; they are not automatically initialized. The compiler will generate an error as a result of the attempt to access the local variables before a value has been assigned.