Answer

1. (C)

2. (C)
52pickup is not a legal identifier. The first character of an identifier cannot be a digit.

3. (A), (B), and (C)
Neither main, String, nor args are reserved keywords, but they are legal identifiers. In the declaration `public static void main(String[] args)`, the identifier main denotes the method that is the entry point of a program. In all other contexts, the identifier main has no predefined meaning.

4. (A)
A value of type `char` can be assigned to a variable of type `int`. An widening conversion will convert the value to an `int`.

5. (A), (D), and (E)
A binary expression with any floating-point operand will be evaluated using floating-point arithmetic. Expressions such as 2/3, where both operands are integers, will use integer arithmetic and evaluate to an integer value. In (E), the result of (0x10 * 1L) is promoted to a floating-point value.

6. (B)
The expression evaluates to –6. The whole expression is evaluated as `(((-(-1)) - ((3 * 10) / 5)) - 1)` according to the precedence and associativity rules.

7. (A), (C), and (D)
The left associativity of the + operator makes the evaluation of `(1 + 2 + "3")` proceed as follows: `(1 + 2) + "3"` to `3 + "3"` to `"33"`. Evaluation of the expression `("1"+ 2 + 3)`, however, will proceed as follows: `("1" + 2) + 3` to `"12" + 3` to `"123"`. `(4 +1.0f)` evaluates as `4.0f + 1.0f` to `5.0f` and `(10/9)` performs integer division, resulting in the value 1. The operand `'a'` in the expression `('a' + 1)` will be promoted to `int`, and the resulting value will be of type `int`.

8. (A) and (C)
The expression `(4 <= 4)` is true. The null literal can be compared, so `(null != null)` yields false.

9. (C) and (D)
Unlike the `&` and `|` operators, the `&&` and `||` operators short-circuit the evaluation of their operands if the result of the operation can be determined from the value of the first operand. The second operand of the `||` operator in the program is never evaluated because of short-circuiting. All the operands of the other operators are evaluated. Variable `i` ends up with the value 3, which is the first digit printed, and `j` ends up with the value 1, which is the second digit printed.

10. (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.

11. (A), (B), and (D)
The conditional expression of an if statement can have any subexpressions, including method calls, as long as the whole expression evaluates to a value of type boolean. The expression `(a = b)` does not compare the variables `a` and `b`, but assigns the value of `b` to the variable `a`. The result of the expression is the value being assigned. Since `a` and `b` are boolean variables, the value returned by the expression is also boolean. This allows the expression to be used as the condition for an if statement. An if statement must always have an if block, but the else clause is optional. The expression `if (false) ; else ;` is legal. In this case, both the if block and the else block are simply the empty statement.

12. (C)
The case label value `2 * iLoc` is a constant expression whose value is 6, the same as the switch expression. Fall through results in the printout shown in (C).

13. (B) and (E)
Both the first and the second number printed will be 10. Both the loop body and the increment expression will be executed exactly 10 times. Each execution of the loop body will be directly followed by an execution of the increment expression. Afterwards, the condition j<10 is evaluated to see whether the loop body should be executed again.

14. (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 syntactically invalid, as there is only one semicolon.

15. (C)
As it stands, the program will compile correctly and will print `"3, 2"` when run. If the `break` statement is replaced with a `continue` statement, the loop will perform all four iterations and will print `"4, 3"`. If the `break` statement is replaced with a `return` statement, the whole method will end when `i` equals 2, before anything is printed. If the `break` statement is simply removed, leaving the empty statement (`;`), the loop will complete all four iterations and will print `"4, 4"`.

16. (E)
The expression `"Hello there".toLowerCase().equals("hello there")` will evaluate to true. The `equals()` method in the `String` class will only return true if the two strings have the same sequence of characters.

17. (C)
The variable `middle` is assigned the value 6. The variable `nt` is assigned the string `"nt"`. The substring `"nt"` occurs three times in the string `"Contentment!"`, starting at indices 2, 5, and 9. The call `s.lastIndexOf(nt, middle)` returns the start index of the last occurrence of `"nt"`, searching backwards from position 6.

18. (B)
The reference value in the reference `str1` never changes and it refers to the string literal `"lower"` all the time. The calls to `toUpperCase()` and `replace()` return a new String object whose reference value is ignored.