socrative

Answer Key
February 24, 2021

OCA Practice Questions Ch. 3 ('Copy')

1. Which of the following variable types is not permitted in a switch statement?

A String

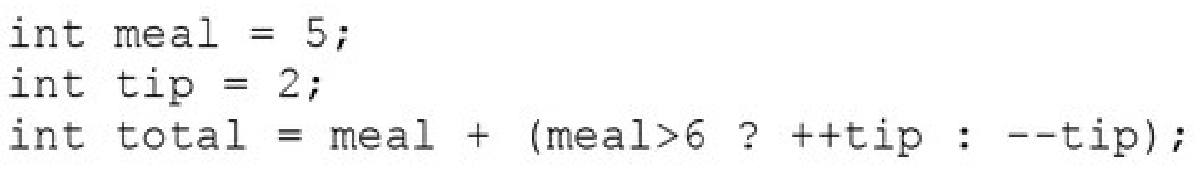
[✓ B] double

C int

D char

i A switch statement supports the primitive types byte, short, char, and int and the classes String, Character, Byte, Short, and Integer. It also supports enumerated types. Floating-point types like float and double are not supported, therefore Option B is the correct answer.

2.



```
int meal = 5;
int tip = 2;
int total = meal + (meal>6 ? ++tip : --tip);
```

What is the value of tip after executing the following code snippet?

[✓ A] 1

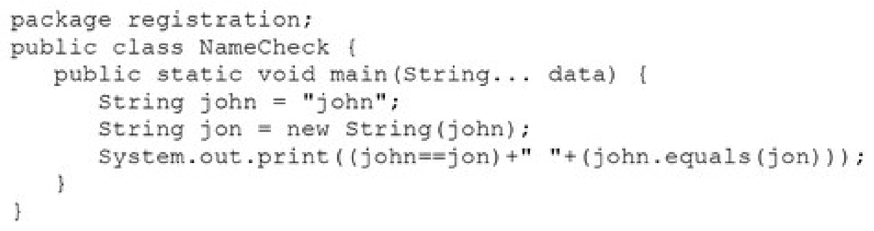
B 2

C 3

D 6

i Remember that in ternary expressions, only one of the two right-most expressions are evaluated. Since meal>6 is false, --tip is evaluated and ++tip is skipped. The result is that tip is changed from 2 to 1, making Option A the correct answer. The value of total is 6, since the pre-increment operator was used on tip, although you did not need to know this to solve the question.

3.



```
package registration;
public class NameCheck {
   public static void main(String... data) {
      String john = "john";
      String jon = new String(john);
      System.out.print((john==jon)+" "+(john.equals(jon)));
   }
}
```

What is the output of the following application?

A true true

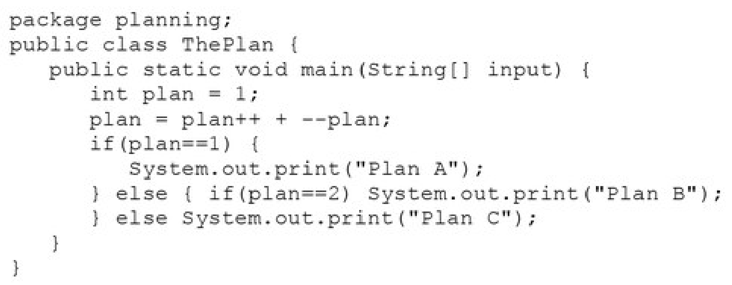
B true false

[✓ C] false true

D false false

i The first assignment creates a new String "john" object. The second line explicitly uses the new keyword, meaning a new String object is created. Since these objects are not the same, the == test on them evaluates to false. The equals() test on them returns true because the values they refer to are equivalent. Therefore, the correct answer is C.

4.



```
package planning;
public class ThePlan {
   public static void main(String[] input) {
      int plan = 1;
      plan = plan++ + --plan;
      if(plan==1) {
         System.out.print("Plan A");
      } else { if(plan==2) System.out.print("Plan B");
      } else System.out.print("Plan C");
   }
}
```

What is the output of the following application?

A Plan A

B Plan B

C Plan C

[✓ D] None of the above

i This code does not compile because it has two else statements as part of a single if-then statement. Notice that the second if statement is not connected to the last else statement. For this reason, Option D, none of the above, is the correct answer.

5. Which of the following statements about a default branch in a switch statement is correct?

A All switch statements must include a default statement.

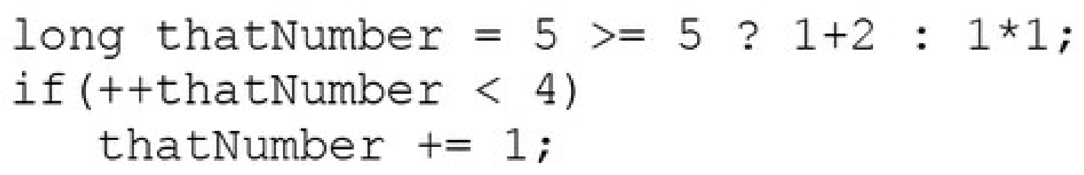
B The default statement is required to be placed after all case statements.

[✓ C] Unlike a case statement, the default statement does not take a value.

D A default statement can only be used when at least one case statement is present.

i A default statement inside a switch statement is optional and can be placed in any order within the switch’s case statements, making Options A and B incorrect. Option D is an incorrect statement as a switch statement can be composed of a single default statement and no case statements. Option C is correct because a default statement does not take a value, unlike a case statement.

6.



```
long thatNumber = 5 >= 5 ? 1+2 : 1*1;
if(++thatNumber < 4)
   thatNumber += 1;
```

What is the value of thatNumber after the execution of the following code snippet?

A 3

[✓ B] 4

C 5

D The answer cannot be determined until runtime.

i The initial assignment of thatNumber follows the first branch of the ternary expression. Since 5 >= 5 evaluates to true, a value of 3 is assigned to thatNumber. In the next line, the pre-increment operator increments the value of thatNumber to 4 and returns a value of 4 to the expression. Since 4 < 4 evaluates to false, the if-then block is skipped. This leaves the value of thatNumber as 4, making Option B the correct answer.

7. Which statement immediately exits a switch statement, skipping all remaining case or default branches?

A exit

[✓ B] break

C goto

D continue

i The break statement exits a switch statement, skipping all remaining branches, making Option B the correct answer. In Option A, exit is not a statement in Java. In Option C, goto is a reserved word but unused in Java. Finally, in Option D, continue is a statement but only used for loops.

8. Which statement about ternary expressions is true?

A In some cases, both expressions to the right of the conditional operator in a ternary expression will be evaluated at runtime.

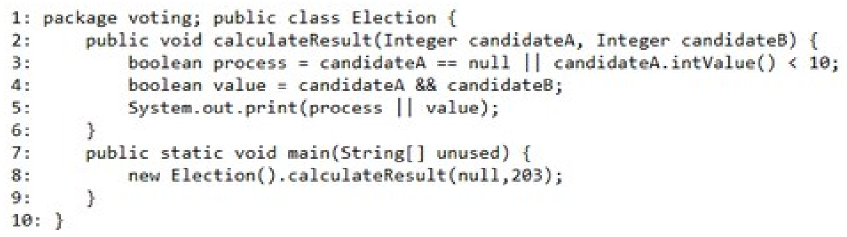
B Ternary expressions require parentheses for proper evaluation.

[✓ C] The ternary expressions are a convenient replacement for an if-then-else statement.

D Ternary expressions support int and boolean expressions for the left-most operand.

i Option A is incorrect as only one of the two right-hand expressions is evaluated at runtime. Parentheses are often helpful for reading ternary expressions but are not required, making Option B incorrect. Option C is a correct statement about ternary operators as they are commonly used to replace short if-then-else statements. Finally, Option D is incorrect as only boolean expressions are permitted in the left-most operand of a ternary expression.

9.



```
1: package voting; public class Election {
2:     public void calculateResult(Integer candidateA, Integer candidateB) {
3:         boolean process = candidateA == null || candidateA.intValue() < 10;
4:         boolean value = candidateA && candidateB;
5:         System.out.print(process || value);
6:     }
7:     public static void main(String[] unused) {
8:         new Election().calculateResult(null,203);
9:     }
10: }
```

What is the output of the following application?

A true

B false

[✓ C] The code does not compile.

D The code compiles but throws a NullPointerException on line 3 at runtime.

i On line 4, candidateA and candidateB are numbers, but the && operation can only be applied to boolean expressions. Therefore, the code does not compile because of line 4, making C the correct answer. All of the other lines are correct. Note that if line 4 is fixed, line 3 does not produce a NullPointerException at runtime. The conditional || and the preceding null check allows the code to only call intValue() if candidateA is not null.

10.

```
package dinosaur;
public class Park {
   public final static void main(String... arguments) {
      int pterodactyl = 6;
      long triceratops  = 3;
      if(pterodactyl % 3 >= 1)
         triceratops++;
         triceratops--;
      System.out.print(triceratops);
   }
}
```

What is the output of the following application?

[✓ A] 2

B 3

C 4

D The code does not compile.

i The first step is to determine whether or not the if-then statement's expression is executed. The expression 6 % 3 evaluates to 0, since there is no remainder, and since 0 >= 1 is false, the expression triceratops++ is not called. Notice there are no brackets {} in the if-then statement. Despite the triceratops-- line being indented, it is not part of the if-then statement. Recall that Java does not use indentation to determine the beginning or end of a statement. Therefore, triceratops-- is always executed, resulting in a value of 2 for triceratops and making Option A the correct answer.

11. Which statement about if-then statements is true?

A An if-then statement is required to have an else statement.

B If the boolean test of an if-then statement evaluates to false, then the target clause of the if-then statement will still be evaluated.

C An if-then statement is required to cast an object.

[✓ D] An if-then statement can execute a single statement or a block {}.

i Option A is incorrect because else statements are entirely optional. Option B is also incorrect. The target of an if-then statement is not evaluated if the boolean test is false. Option C is incorrect. While an if-then statement is often used to test whether an object is of a particular type in order to cast it, it is not required to cast an object. Option D is correct as an if-then statement may execute a single statement or a block of code {}.

12.

```
package restaurant;
public class Pieces {
   public static void main(String[] info) {
      int flair = 15;
      if(flair >= 15 && flair < 37) {
         System.out.print("Not enough");
      } if(flair==37) {
         System.out.print("Just right");
      } else {
         System.out.print("Too many");
      }
   }
}
```

What is the output of the following application?

- **A** Not enough
- **B** Just right
- **C** Too many
- ✓ **D** None of the above

i For this question, it helps to notice that the second if-then statement is not connected to the first if-then statement, as there is no else joining them. When this code executes, the first if-then statement outputs Not enough since flair is >= 15 and < 37. The second if-then statement is then evaluated. Since flair is not 37, the expression Too many is outputted. Since two statements are outputted, Option D, none of the above, is the correct answer.

13. Which statement about case statements of a switch statement is not true?

- **A** A case value can be final.
- ✓ **B** A case statement must be terminated with a break statement.
- **C** A case value can be a literal expression.
- **D** A case value must match the data type of the switch variable, or be able to be promoted to that type.

i A case value must be a constant expression, such as a literal or final variable, so Options A and C are true statements about case values. A case statement may be terminated by a break statement, but it is not required, making Option B the false statement and correct answer. Option D is also a true statement about case values.

14. Given the following truth table, which operator for the boolean expressions x and y corresponds to this relationship?

	x = true	x = false
y = true	true	false
y = false	false	false

- **A** --
- **B** ++
- **C** ||
- ✓ **D** &&

i The question is about boolean operators. Since Options A and B are numeric operators, they can be instantly disregarded. The question then simplifies to which boolean expression, && or ||, corresponds to the truth table that only evaluates to true if both operands are true. Only the conjunctive logical && operator represents this relationship, making Option D the correct answer.

15.

```
int hops = 0;
int jumps = 0;
jumps = hops++;
if(jumps)
   System.out.print("Jump!");
else
   System.out.print("Hop!");
```

What is the output of the following code snippet?

A Jump!

B Hop!

[✓ C] The code does not compile.

D The code compiles but throws an exception at runtime.

i The value of jumps and hops is unimportant because this code does not compile, making Option C the correct answer. Unlike some other programming languages, Java does not automatically convert integers to boolean values for use in if-then statements. The statement if(jumps) evaluates to if(0), and since 0 is not a boolean value, the code does not compile. Note that the value of the jumps variable is irrelevant in this example; no integer evaluates to a boolean value in Java.

16. The ________________ operator increases the value of a variable by 1 and returns the new value, while the ________________ operator decreases the value of a variable by 1 and returns the original value.

A pre-increment [++v], pre-decrement [--v]

[✓ B] pre-increment [++v], post-decrement [v--]

C post-increment [v++], pre-decrement [--v]

D post-increment [v++], post-decrement [v--]

i Prefix operators modify the variable and evaluate to the new value, while postfix operators modify the variable but return the original value. Therefore, Option B is the correct answer.

17.

```
package jungle;
public class TheBigRace {
   public static void main(String[] in) {
      int tiger = 2;
      short lion = 3;
      long winner = lion+2*(tiger + lion);
      System.out.print(winner);
   }

}
```

What is the output of the following application?

A 11

[✓ B] 13

C 25

D None of the above

i For this problem, it helps to recognize that parentheses take precedence over the operations outside the parentheses. Once we replace the variables with values, the expression becomes: 3+2*(2+3). We then calculate the value inside the parentheses to get 3+2*5. Since the multiplication operator has higher precedence than addition, we evaluate it first, resulting in 3+10 = 13, making Option B the correct answer.

18.

```
final _____ saturday = 6;
switch(dayOfWeek) {
   default:
      System.out.print("Another Weekday");
      break;
   case saturday:
      System.out.print("Weekend!");
}
```

Given the following code snippet, assuming dayOfWeek is an int, what variable type of saturday is not permitted?

A byte

[✓ B] long

C int

D None of the above

i Any value that can be implicitly promoted to int will work for the case statement with an int input. Since switch statements do not support long values, and long cannot be converted to int without a possible loss of data, Option B is the correct answer.

19.

```
int time = 11;
int day = 4;
String dinner = time > 10 ? day ? "Takeout" : "Salad" : "Leftovers";
```

Given the following code snippet, what is the value of dinner after it is executed?

- A Takeout
- B Salad
- C The code does not compile but would compile if parentheses were added.
- [✓ D] None of the above

i While parentheses are recommended for ternary operations, especially embedded ones, they are not required, so Option C is incorrect. The code does not compile because day is an int, not a boolean expression, in the second ternary operation, making Option D the correct answer. Remember that in Java, numeric values are not accepted in place of boolean expressions in if-then statements or ternary operations.

20.

```
package recreation;
public class Dancing {
   public static void main(String[] vars) {
      int leaders = 10 * (2 + (1 + 2 / 5);
      int followers = leaders * 2;
      System.out.print(leaders + followers < 10 ? "Too few" : "Too many");
   }
}
```

What is the output of the following application?

- A Too few
- B Too many
- [✓ C] The code does not compile.
- D The code compiles but throws a division by zero error at runtime.

i While the code involves numerous operations, none of that matters for solving this problem. The key to solving itis to notice that the line that assigns the leaders variable has an uneven number of parentheses. Without balanced parentheses, the code will not compile, making Option C the correct answer.

21.

```
package schedule;
public class PrintWeek {
   public static final void main(String[] days) {
      System.out.print(5 + 6 + "7" + 8 + 9);
   }
}
```

What is the output of the following application?

- A 56789
- [✓ B] 11789
- C 11717
- D The code does not compile.

i Remember that Java evaluates + from left to right. The first two values are both numbers, so the + is evaluated as numeric addition, resulting in a reduction to 11 + "7" + 8 + 9. The next two terms, 11 + "7", are handled as string concatenation since one of the terms is a String. This allows us to reduce the expression to "117" + 8 + 9. Likewise, the final two terms are each evaluated one at a time with the String on the left. Therefore, the final value is 11789, making Option B the correct answer.

22. The ________________ operator is used to find the difference between two numbers, while the ________________ operator is used to find the remainder when one number is divided by another.

A /, %

[✓ B] –, %

C %, <

D -, ||

i The subtraction - operator is used to find the difference between two numbers, while the modulus % operator is used to find the remainder when one number is divided by another, making Option B the correct answer. The other options use operators that do not match this description.

23.

```
package transporter;
public class Rematerialize {
   public static void main(String[] input) {
      int dog = 11;
      int cat = 3;
      int partA = dog / cat;
      int partB = dog % cat;
      int newDog = partB + partA * cat;
      System.out.print(newDog);
   }
}
```

What is the output of the following application?

A 9

[✓ B] 11

C 15

D The code does not compile.

i The code compiles without issue, making Option D incorrect. The focus of this question is showing how the division and modulus of two numbers can be used to reconstitute one of the original operands. In this example, partA is the integer division of the two numbers. Since 3 does not divide 11 evenly, it is rounded down to 3. The variable partB is the remainder from the first expression, which is 2. The newDog variable is an expression that reconstitutes the original value for dog using the division value and the remainder. Note that due to operator precedence, the multiplication * operation is evaluated before the addition + operation. The result is the original value of 11 for dog is outputted by this program.

24.

```
package dessert;
public class IceCream {
   public final static void main(String... args) {
      int flavors = 30;
      int eaten = 0;
      switch(flavors) {
         case 30: eaten++;
         case 40: eaten+=2;
         default: eaten--;
      }
      System.out.print(eaten);
   }
}
```

What is the output of the following application?

- A 1
- ✓ B 2
- C 3
- D The code does not compile.

i The code compiles without issue, so Option D is incorrect. In this question's switch statement, there are no break statements. Once the matching case statement, 30, is reached, all remaining case statements will be executed. The variable eaten is increased by 1, then 2, then reduced by 1, resulting in a final value of 2, making Option B the correct answer.

25.

```
package mode;
public class Transportation {
   public static String travel(int distance) {
      return distance<1000 ? "train" : 10;
   }
   public static void main(String[] answer) {
      System.out.print(travel(500));
   }
}
```

What is the output of the following application?

- A train
- B 10
- ✓ C The code does not compile.
- D The code compiles but throws an exception at runtime.

i Ternary operations require both right-hand expressions to be of compatible data types. In this example, the first right-hand expression of the outer ternary operation is of type String, while the second right-hand expression is of type int. Since these data types are incompatible, the code does not compile, and Option C is the correct answer.

26. Given two non-null String objects with reference names apples and oranges, if apples ________________ oranges evaluates to true, then apples ________________ oranges must also evaluate to true.

- ✓ A ==, equals()
- B !=, equals()
- C equals(), ==
- D equals(), =!

i For this question, remember that if two String objects evaluate to true using ==, then they are the same object. If they are the same String object, equals() will trivially return true. Option A correctly reflects this principle. Option B is incorrect as two String objects that are not the same may still be equivalent in terms of equals(). For example, apples == new String(apples) evaluates to false, but equals() will evaluate to true on these String objects. Likewise, Options C and D are also incorrect because two String objects that are equivalent in terms of equals() may be different objects.

27. For a given non-null String myTestVariable, what is the resulting value of executing the statement myTestVariable.equals(null)?

A true

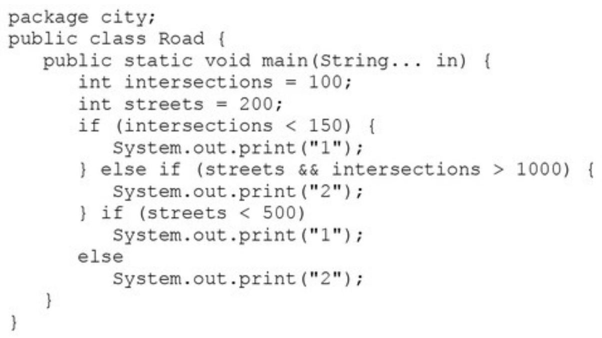
✓ **B** false

C The statement does not compile.

D The statement compiles but will produce an exception when used at runtime.

i The statement compiles and runs without issue, making Options C and D incorrect. Since we are given that myTestVariable is not null, the statement will always evaluate to false, making Option B the correct answer. Note that if myTestVariable was null, then the code would still compile but throw a NullPointerException calling equals() at runtime.

28.



```
package city;
public class Road {
   public static void main(String... in) {
      int intersections = 100;
      int streets = 200;
      if (intersections < 150) {
         System.out.print("1");
      } else if (streets && intersections > 1000) {
         System.out.print("2");
      } if (streets < 500)
         System.out.print("1");
      else
         System.out.print("2");
   }
}
```

How many 1s are outputted when the following application is compiled and run?

A None

B One

C Two

✓ **D** The code does not compile.

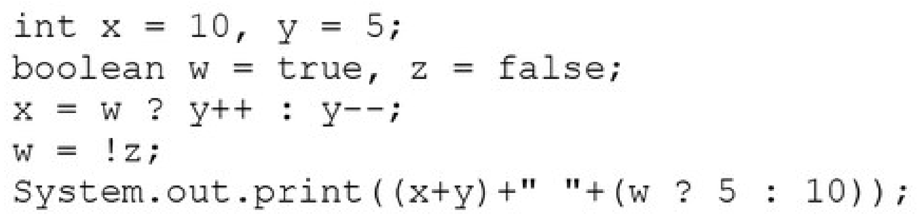
i The code does not compile, making Option D the correct answer. The reason the code does not compile is due to the test in the second if-then statement. The expression (streets && intersections > 1000) is invalid because streets is not a boolean expression and cannot be used as the left-hand side of the conjunctive logical && operator. The line of code is designed to resemble the corrected expression (streets > 1000 && intersections > 1000. Notice the fixed expression requires two relational > operators. If the second if-then statement was corrected, then the application would compile and produce two 1's, making Option C the correct answer.

29. Which statement about the logical operators & and && is true?

- **A** The & and && operators are interchangeable, always producing the same results at runtime.
- [✓ **B**] The & operator always evaluates both operands, while the && operator may only evaluate the left operand.
- **C** Both expressions evaluate to true if either operand is true.
- **D** The & operator always evaluates both operands, while the && operator may only evaluate the right operand.

i The & and && (AND) operators are not interchangeable, as the conjunctive & operator always evaluates both sides of the expression, while the conditional conjunctive && operator only evaluates the right-hand side of the expression if the left side is determined to be true. This is why conditional operators are often referred to as short-circuit operators, skipping the right-hand side expression at runtime. For these reasons, Option B is the correct answer. Note that Option C is an incorrect statement as well, since it describes disjunctive (OR) operators.

30.



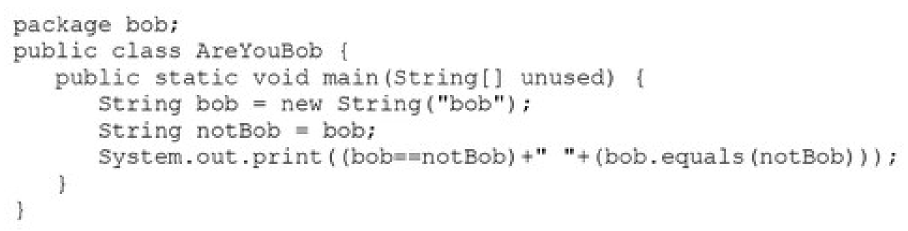
```java
int x = 10, y = 5;
boolean w = true, z = false;
x = w ? y++ : y--;
w = !z;
System.out.print((x+y)+" "+(w ? 5 : 10));
```

What is the output of the following code snippet?

- **A** The code does not compile.
- **B** 10 10
- [✓ **C**] 11 5
- **D** 12 5

i The code compiles, so Option A is incorrect. Since w starts out true, the third line takes the first right-hand side of the ternary expression returning and assigning 5 to x (post-increment operator) while incrementing y to 6. Note that the second right-hand side of the ternary expression y-- is not evaluated since ternary operators only evaluate one right-hand expression at runtime. On the fourth line, the value of w is set to !z. Since z is false, the value of w remains true. The final line outputs the value of (5+6) and (true ? 5 : 10), which is 11 5, making Option C the correct answer.

31.



```java
package bob;
public class AreYouBob {
   public static void main(String[] unused) {
      String bob = new String("bob");
      String notBob = bob;
      System.out.print((bob==notBob)+" "+(bob.equals(notBob)));
   }
}
```

What is the output of the following application?

- [✓ **A**] true true
- **B** true false
- **C** false true
- **D** false false

i The first assignment actually uses two String objects, the literal "bob" and the String created with the new keyword. Regardless, only the second object is assigned to the variable bob. The second variable, notBob, is assigned a reference to the value of the bob variable. This means that not only does the equals() test pass, but they are actually the same object, so the == test is true as well. Therefore, the correct answer is Option A.

32. What is the value of 12 + 6 * 3 % (1 + 1) in Java?

A 0

[✓ B] 12

C 14

D None of the above

i The question is about operator precedence and order of operation. The multiplication * and modulus % operators have the highest precedence, although what is inside the parentheses needs to be evaluated first. We can reduce the expression to the following: 12 + 6 * 3 % 2. Since multiplication * and modulus % have the same operator precedence, we evaluate them from left to right as follows: 12 + 6 * 3 % 2 → 12 + 18 % 2 → 12 + 0 → 12. We see that despite all of the operators on the right-hand side of the expression, the result is zero, leaving us a value of 12, making Option B the correct answer.

33. Given the following truth table, the boolean variables p and q, and the expression p ^ q, what are the missing values in the truth table, starting with the first column?

	p = true	p = false
q = true	false	true
q = false	______	______

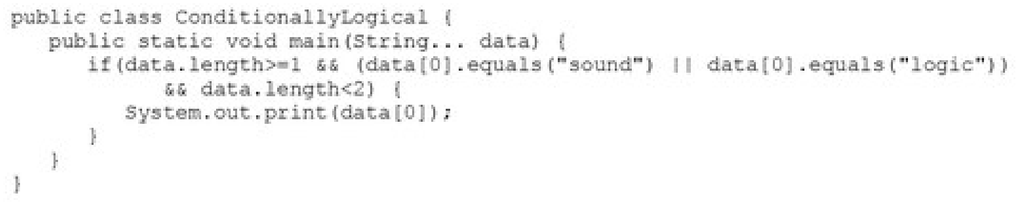
A false and true

B false and false

C true and true

[✓ D] true and false

i The XOR ^ operator evaluates to true if p and q differ and false if they are the same. Therefore, the missing values are true and false, making Option D the correct answer.

34. 

```
public class ConditionallyLogical {
   public static void main(String... data) {
      if(data.length>=1 && (data[0].equals("sound") || data[0].equals("logic"))
            && data.length<2) {
         System.out.print(data[0]);
      }
   }
}
```

Which of the following is not a possible result of executing the following application?

A Nothing is printed.

B sound is printed.

[✓ C] The application throws an exception at runtime.

D logic is printed.

i The key to understanding this question is to remember that the conditional conjunction && operator only executes the right-hand side of the expression if the left-hand side of the expression is true. If data is an empty array, then the expression ends early and nothing is output. The second part of the expression will return true if data's first element is sound or logic. Since we know from the first part of the statement that data is of length at least one, no exception will be thrown. The final part of the expression with data.length<2 doesn't change the output when data is an array of size one. Therefore, sound and logic are both possible outputs. For these reasons, Option C is the only result that is unexpected at runtime.

35. The operators +, ________________, ________________, ________________, and ++ are listed in the same or increasing level of operator precedence.

A *, --, /

B %, -, *

[✓ C] /, *, %

D *, -, /

i In Option A, the division operator / incorrectly comes after the decrement -- operator. In Option B, the subtraction operator - incorrectly comes after the modulus % operator. In Option D, the division operator / incorrectly comes after the subtraction - operator. The correct answer is Option C, where all three operators have the same order of precedence.

36. What statement about the ^ operator is correct?

A If one of the operands of ^ is true, then the result is always true.

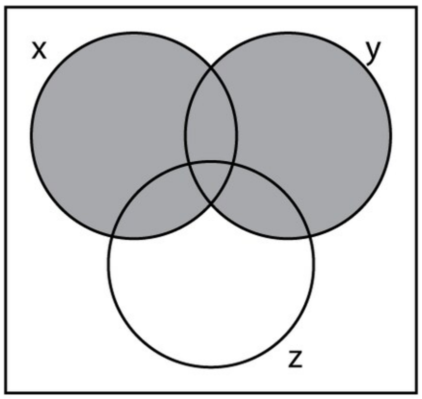
B There is a conditional form of the operator, denoted as ^^.

C If both operands of ^ are true, the result is true.

[✓ D] The ^ operator can only be applied to boolean values.

i The exclusive or (XOR) ^ operator requires evaluating both operands to determine the result. For this reason, Options A and B are incorrect. For Option B, you can't have a short-circuit operation if both operands are always read, therefore ^^ does not exist. Option C is an incorrect statement as the ^ operator only returns true if exactly one operand is true. Finally, Option D is correct as the ^ is only applied to boolean values in Java.

37. Given the following Venn diagram and the variables, x, y, and z, which Java expression most closely represents the filled-in region of the diagram?



A x || z

B y || (y && z)

[✓ C] x || y

D y && x

i The diagram represents the overlap of x and y, corresponding to when one of them is true. Therefore, x || y, Option C, most closely matches this relationship. Note that z is unused in the diagram and therefore is not required in any expression.

38. What variable type of red allows the following application to compile?

```
package tornado;
public class Kansas {
   public static void main(String[] args) {
      int colorOfRainbow = 10;
      _____ red = 5;
      switch(colorOfRainbow) {
         default:
            System.out.print("Home");
            break;
         case red:
            System.out.print("Away");
      }
   }
}
```

A long

B double

C int

[✓ D] None of the above

i The value of a case statement must be constant, a literal value, or final variable. Since red is missing the final attribute, no variable type allows the code to compile, making Option D the correct answer.

39. Which two operators would be used to test if a number is equal to or greater than 5.21 but strictly less than 8.1?

A > and <=

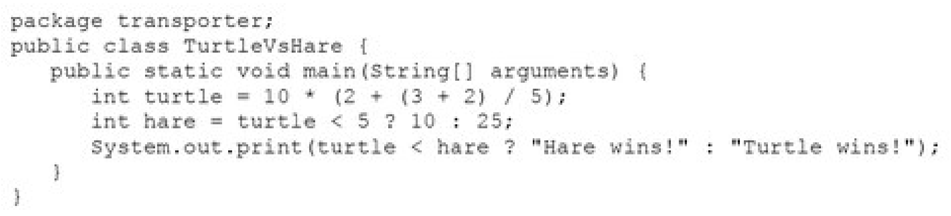
B >= and >

[✓ C] < and >=

D < and >

i The question is asking which operator represents greater than or equal to and which operator is strictly less than. The >= and < correspond to these operators, respectively. Therefore, Option C is the correct answer. Note that the question does not specify which order the operators needed to appear in, only to select the two operators that match the question description.

40.



```
package transporter;
public class TurtleVsHare {
   public static void main(String[] arguments) {
      int turtle = 10 * (2 + (3 + 2) / 5);
      int hare = turtle < 5 ? 10 : 25;
      System.out.print(turtle < hare ? "Hare wins!" : "Turtle wins!");
   }
}
```

What is the output of the following application?

A Hare wins!

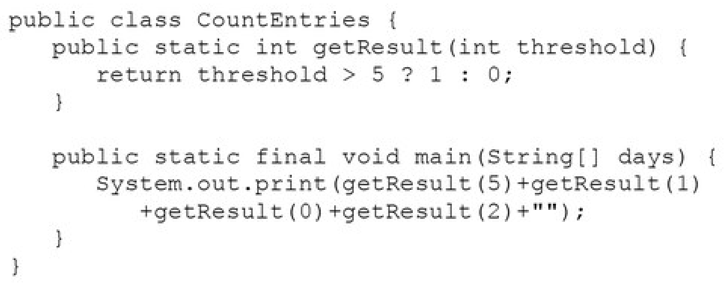
[✓ B] Turtle wins!

C The code does not compile.

D The code compiles but throws a division by zero error at runtime.

i The code compiles and runs without issue, making Options C and D incorrect. The key here is understanding operator precedence and applying the parentheses to override precedence correctly. The first expression is evaluated as follows: 10 * (2 + (3 + 2) / 5) → 10 * (2 + 5 / 5) → 10 * (2 + 1) → 10 * 3, with a final value of 30 for turtle. Since turtle is not less than 5, a value of 25 is assigned to hare. Since turtle is not less than hare, the last expression evaluates to Turtle wins!, which is outputted to the console, making Option B the correct answer.

41.



```
public class CountEntries {
   public static int getResult(int threshold) {
      return threshold > 5 ? 1 : 0;
   }

   public static final void main(String[] days) {
      System.out.print(getResult(5)+getResult(1)
         +getResult(0)+getResult(2)+"");
   }
}
```

What is the output of the following application?

[✓ A] 0

B 1

C 0000

D 1000

i All of the terms of getResult() in this question evaluate to 0, since they are all less than or equal to 5. The expression can therefore be reduced to 0+0+0+0+"". Since Java evaluates the + operator from left to right, the four operands on the left are applied using numeric addition, resulting in the expression 0+"". This expression just converts the value to a String, resulting in an output of 0, making Option A the correct answer.

42.

```
package yoyo;
public class TestGame {
   public String runTest(boolean spinner, boolean roller) {
      if(spinner = roller) return "up";
      else return roller ? "down" : "middle";
   }
   public static final void main(String pieces[]) {
      final TestGame tester = new TestGame();
      System.out.println(tester.runTest(false,true));
   }
}
```

What is the output of the following application?

[✓ A] up

B middle

C down

D The code does not compile.

i The code compiles without issue, so Option D is incorrect. The key here is that the if-then statement in the runTest() method uses the assignment operator (=) instead of the (==) operator. The result is that spinner is assigned a value of true, and the statement (spinner = roller) returns the newly assigned value. The method then returns up, making Option A the correct answer. If the (==) operator had been used in the if-then statement, then the process would have branched to the else statement, with down being returned by the method.

43. The ________________ operator is true if either of the operands are true, while the ________________ operator flips a boolean value.

A +, -

B &&, !

C |, -

[✓ D] ||, !

i The conditional disjunction (OR) || operator is true if either of the operands are true, while the logical complement (!) operator reverses or flips a boolean value, making Option D the correct answer. The other options use operators that do not match this description. In particular, Options A and C include operators that can only be applied to numerical values, not boolean ones.

44.

```
int characters = 5;
int story = 3;
double movieRating = characters <= 4 ? 3 : story>1 ? 2 : 1;
```

Given the following code snippet, what is the value of movieRating after it is executed?

[✓ A] 2.0

B 3.0

C The code does not compile but would compile if parentheses were added.

D None of the above

i While parentheses are recommended for ternary operations, especially embedded ones, they are not required, so Option C is incorrect.. The first ternary operation evaluates characters <= 4 as false, so the second ternary operation is executed. Since story > 1 is true, the final value of movieRating is 2.0, making Option A the correct answer.

45. A switch statement can have ____ case statements and ____ default statements.

A at most one, at least one

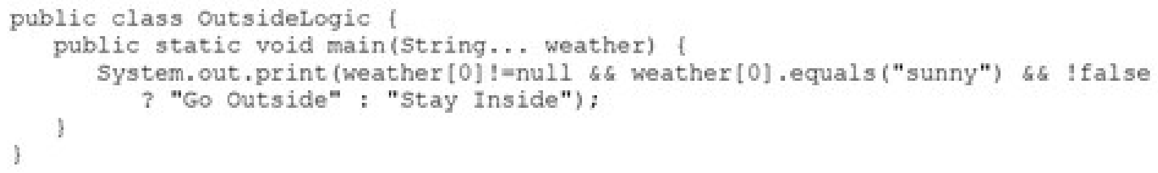
[✓ B] any number of, at most one

C at least one, any number of

D at least one, at most one

i Barring any JVM limitations, a switch statement can have any number of case statements (including none) but at most one default statement, with Option B correctly identifying this relationship.

46.



```
public class OutsideLogic {
   public static void main(String... weather) {
      System.out.print(weather[0]!=null && weather[0].equals("sunny") && !false
         ? "Go Outside" : "Stay Inside");
   }
}
```

Which of the following is not a possible result of executing the following application?

[✓ A] Nothing is printed.

B The application throws an exception at runtime.

C Go Outside is printed.

D Stay Inside is printed.

i The application uses the conditional conjunction && operator to test if weather[0] is null, but unfortunately this test does not work on zero-length arrays. Therefore, it is possible this code will throw an ArrayIndexOutOfBoundsException at runtime. The second part of the expression evaluates to true if the first input of weather matches sunny. The final part of the expression, && !false, is a tautology in that it is always true and has no impact on the expression. Either an exception will be thrown or text will be output, based on the value of weather, therefore Option A is the correct answer.

47. What is the value of (5 + (!2 + 8) * 3 - 3 % 2)/2 in Java?

A 2

B 11

C 16

[✓ D] None of the above

i The question looks a lot more difficult than it is. In fact, to solve it you don't have to compute anything! You just have to notice that the logical complement operator (!), which can only be applied to boolean values, is being applied to a numeric value. Therefore, the answer is that the expression wouldn't compile or run, making Option D the correct answer.

48. Given the following truth table, the boolean variables w and z, and the expression w || z, what are the missing values in the truth table, starting with the first row?

	w = true	w = false
z = true	true	______
z = false	______	false

A false and false

B true and false

✓ C true and true

D false and true

i The disjunctive logical || operator evaluates to true if either operand is true. Another way to look at it is that it only evaluates to false if both operands are false. Therefore, the missing values are both true, making Option C the correct answer.

49. The operators -, ______________, ______________, ______________, and % are listed in the same or increasing level of operator precedence.

✓ A +, /, *

B --, -, *

C ++, /, *

D *, ++, %

i In Option B, the subtraction operator - incorrectly comes after the decrement -- operator. In Option C, the division operator / incorrectly comes after the increment ++ operator. In Option D, the modulus operator % incorrectly comes after the increment ++ operator. The correct answer is Option A, where the subtraction - and addition + operators are followed by the division / and multiplication * operators.

50.

```
public class Baby {
   public static String play(int toy, int age) {
      final String game;
      if(toy<2)
         game = age > 1 ? 1 : 10; // p1
      else
         game = age > 3 ? "Ball" : "Swim"; // p2
      return game;
   }
   public static void main(String[] variables) {
      System.out.print(play(5,2));
   }
}
```

What is the output of the following application?

A Ball

B Swim

✓ C The code does not compile due to p1.

D The code does not compile due to p2.

i The key to solving this problem is remembering that the type of the value returned by a ternary operation is determined by the expressions on the right-hand side. On line p1, the expressions are of type int, but the assignment is to the variable game, of type String. Since the assignment is invalid, the code does not compile, and Option C is correct.