1. Which of the following data types can be used in a switch expression? (Choose all that apply.)

A. enum

B. int

C. Byte

D. long

E. String

F. char

G. var

H. double

A, B, C, E, F, G

Una expresión ***switch*** admite solo los primitivos int, byte, short y char, junto con sus clases contenedoras asociadas Integer, Byte, Short y Character, respectivamente, lo que hace que las opciones B, C y F sean correctas y descarta las opciones D y H. También admite enum y String, lo que hace que las opciones A y E sean correctas. Finalmente, switch admite var si el tipo se puede resolver a un tipo de datos switch compatible, lo que hace que la opción G sea correcta.

2. What is the output of the following code snippet? (Choose all that apply.)

```
3: int temperature = 4;
4: long humidity = -temperature + temperature * 3;
5: if (temperature>=4)
6: if (humidity < 6) System.out.println("Too Low");
7: else System.out.println("Just Right");
8: else System.out.println("Too High");
```

A. Too Low

B. Just Right

C. Too High

D. A NullPointerException is thrown at runtime.

E. The code will not compile because of line 7.

F. The code will not compile because of line 8.

B

El código compila y se ejecuta sin problemas, por lo que las opciones D, E y F son incorrectas. Aunque dos sentencias else consecutivas en las líneas 7 y 8 parecen un poco extrañas, están asociadas con sentencias if separadas en las líneas 5 y 6, respectivamente. El valor de humidity en la línea 4 es igual a -4 + 12, que es 8. La primera sentencia if se evalúa como verdadera en la línea 5, por lo que la línea 6 se ejecuta y se evalúa como falsa. Esto hace que la sentencia else en la línea 7 se ejecute, imprimiendo Just Right y haciendo que la opción B sea la respuesta correcta.

3. Which of the following data types are permitted on the right side of a for-each expression? (Choose all that apply.)

A. Double[][]

B. Object

C. Map

D. List

E. String

F. char[]

G. Exception

H. Set

A, D, F, H

Un bucle for-each admite matrices, lo que hace que las opciones A y F sean correctas. Para Double[][], cada elemento del bucle for-each sería un Double[]. Un bucle for-each también admite clases que implementan java.lang.Iterable. Aunque esto incluye muchas de las clases de Collections Framework, no todas implementan java.lang.Iterable. Por esta razón, la opción C es incorrecta y las opciones D y H son correctas. Las opciones B, E y G son incorrectas, ya que no implementan java.lang.Iterable. Aunque una String es una lista de caracteres ordenados, la clase no implementa la interfaz requerida para un bucle for-each.

4. What is the output of calling printReptile(6)?

```
void printReptile(int category) {
 var type = switch(category) {
    case 1,2 -> "Snake";
    case 3,4 -> "Lizard";
    case 5,6 -> "Turtle";
    case 7,8 -> "Alligator";
 };
 System.out.print(type);
}
```

A. Snake

B. Lizard

C. Turtle

D. Alligator

E. TurtleAlligator

F. None of the above

F

El código no compila porque la expresión switch requiere que se manejen todos los valores de caso posibles, lo que hace que la opción F sea correcta. Si se agregara una sentencia default válida, entonces el código compilaría e imprimiría Turtle en tiempo de ejecución. A diferencia de las sentencias switch tradicionales, las expresiones switch ejecutan exactamente una rama y no usan sentencias break entre las sentencias case.

5. What is the output of the following code snippet?

```
List<Integer> myFavoriteNumbers = new ArrayList<>();
myFavoriteNumbers.add(10);
myFavoriteNumbers.add(14);
for (var a : myFavoriteNumbers) {
  System.out.print(a + ", ");
  break;
}

for (int b : myFavoriteNumbers) {
  continue;
  System.out.print(b + ", ");
}

for (Object c : myFavoriteNumbers)
  System.out.print(c + ", ");
```

A. It compiles and runs without issue but does not produce any output.

B. 10, 14,

C. 10, 10, 14,

D. 10, 10, 14, 10, 14,

E. Exactly one line of code does not compile.

F. Exactly two lines of code do not compile.

G. Three or more lines of code do not compile.

H. The code contains an infinite loop and does not terminate.

E

El segundo bucle for-each contiene un continue seguido de una sentencia print(). Debido a que el continue no es condicional y siempre se incluye como parte del cuerpo del bucle for-each, la sentencia print() no es alcanzable. Por esta razón, la sentencia print() no compila. Como este es el único error de compilación, la opción E es correcta. Las otras líneas de código compilan sin problemas.

6. Which statements about decision structures are true? (Choose all that apply.)

A. A for-each loop can be executed on any Collections Framework object.

B. The body of a while loop is guaranteed to be executed at least once.

C. The conditional expression of a for loop is evaluated before the first execution of the loop body.

D. A switch expression that takes a String and assigns the result to a variable requires a default branch.

E. The body of a do/while loop is guaranteed to be executed at least once.

F. An if statement can have multiple corresponding else statements.

C, D, E

Un bucle for-each se puede ejecutar en cualquier objeto Collections que implemente java.lang.Iterable, como List o Set, pero no en todas las clases de Collections, como Map, por lo que la opción A es incorrecta. El cuerpo de un bucle do/while se ejecuta una o más veces, mientras que el cuerpo de un bucle while se ejecuta cero o más veces, lo que hace que la opción E sea correcta y la opción B incorrecta. La expresión condicional de los bucles for se evalúa al inicio de la ejecución del bucle, lo que significa que el bucle for puede ejecutarse cero o más veces, lo que hace que la opción C sea correcta. Una expresión switch que toma un String requiere una rama default si el resultado se asigna a una variable, lo que hace que la opción D sea correcta. Finalmente, cada sentencia if tiene como máximo una sentencia else coincidente, lo que hace que la opción F sea incorrecta.

7. Assuming weather is a well-formed nonempty array, which code snippet, when inserted independently into the blank in the following code, prints all of the elements of weather? (Choose all that apply.)

```
private void print(int[] weather) {
  for(_____) {
    System.out.println(weather[i]);
  }
}
```

A. `int i=weather.length; i>0; i--`

B. `int i=0; i<=weather.length-1; ++i`

C. `var w : weather`

D. `int i=weather.length-1; i>=0; i--`

E. `int i=0, int j=3; i<weather.length; ++i`

F. `int i=0; ++i<10 && i<weather.length;`

G. None of the above

B, D

La opción A es incorrecta porque en la primera iteración, intenta acceder a weather[weather.length] de la matriz no vacía, lo que provoca que se lance una ArrayIndexOutOfBoundsException. La opción B es correcta e imprimirá los elementos en orden. La opción C no compila ya que i no está definido en weather[i]. Para que esto funcione, el cuerpo del bucle for-each también debería actualizarse. La opción D también es correcta y es una forma común de imprimir los elementos de una matriz en orden inverso. La opción E no compila y, por lo tanto, es incorrecta. Puede declarar varios elementos en un bucle for, pero el tipo de datos debe aparecer solo una vez, como en for(int i=0, j=3; ...). Finalmente, la opción F es incorrecta porque se omite el primer elemento de la matriz. Dado que la expresión condicional se verifica antes de que se ejecute el bucle por primera vez, el primer valor de i utilizado dentro del cuerpo del bucle será 1.

8. What is the output of calling printType(11)?

```
31: void printType(Object o) {
32:   if(o instanceof Integer bat) {
33:     System.out.print("int");
34:   } else if(o instanceof Integer bat && bat < 10) {
35:     System.out.print("small int");
36:   } else if(o instanceof Long bat || bat <= 20) {
37:     System.out.print("long");
38:   } default {
39:     System.out.print("unknown");
40:   }
41: }
```

A. int

B. small int

C. long

D. unknown

E. Nothing is printed.

F. The code contains one line that does not compile.

G. The code contains two lines that do not compile.

H. None of the above

G

Las dos primeras sentencias de coincidencia de patrones compilan sin problemas. Se permite volver a usar la variable bat, siempre que ya no esté en el ámbito. Sin embargo, la línea 36 no compila. Debido al alcance del flujo, si s no es un Long, entonces bat no está en el ámbito en la expresión bat <= 20. La línea 38 tampoco compila, ya que default no se puede usar como parte de una sentencia if/else. Por estas dos razones, la opción G es correcta.

9. Which statements, when inserted independently into the following blank, will cause the code to print 2 at runtime? (Choose all that apply.)

```
int count = 0;
BUNNY: for(int row = 1; row <=3; row++)
RABBIT: for(int col = 0; col <3 ; col++) {
  if((col + row) % 2 == 0)
    _____;
  count++;
}
System.out.println(count);
```

A. break BUNNY

B. break RABBIT

C. continue BUNNY

D. continue RABBIT

E. break

F. continue

G. None of the above, as the code contains a compiler error.

B, C, E

El código contiene un bucle anidado y una expresión condicional que se ejecuta si la suma de col + row es un número par; de lo contrario, count se incrementa. Ten en cuenta que las opciones E y F son equivalentes a las opciones B y D, respectivamente, ya que las sentencias sin etiqueta se aplican al bucle más interno. Estudiando los bucles, la primera vez que la condición es verdadera es en la segunda iteración del bucle interno, cuando row es 1 y col es 1. La opción A es incorrecta porque esto hace que el bucle salga inmediatamente con count solo establecido en 1. Las opciones B, C y E siguen la misma ruta. Primero, count se incrementa a 1 en el primer bucle interno, y luego se sale del bucle interno. En la siguiente iteración del bucle externo, row es 2 y col es 0, por lo que la ejecución sale del bucle interno inmediatamente. En la tercera iteración del bucle externo, row es 3 y col es 0, por lo que count se incrementa a 2. En la siguiente iteración del bucle interno, la suma es par, por lo que salimos y nuestro programa está completo, lo que hace que las opciones B, C y E sean correctas. Las opciones D y F son incorrectas, ya que hacen que los bucles interno y externo se ejecuten varias veces, con count teniendo un valor de 5 cuando termina. No necesitas rastrear todas las iteraciones; simplemente detente cuando el valor de count exceda 2.

10. Given the following method, how many lines contain compilation errors? (Choose all that apply.)

```
10: private DayOfWeek getWeekDay(int day, final int thursday) {
11:   int otherDay = day;
12:   int Sunday = 0;
13:   switch(otherDay) {
14:     default:
15:     case 1: continue;
16:     case thursday: return DayOfWeek.THURSDAY;
17:     case 2,10: break;
18:     case Sunday: return DayOfWeek.SUNDAY;
19:     case DayOfWeek.MONDAY: return DayOfWeek.MONDAY;
20:   }
21:   return DayOfWeek.FRIDAY;
22: }
```

A. None, the code compiles without issue.

B. 1

C. 2

D. 3

E. 4

F. 5

G. 6

H. The code compiles but may produce an error at runtime.

E

Este código contiene numerosos errores de compilación, lo que hace que las opciones A y H sean incorrectas. La línea 15 no compila, ya que continue no se puede usar dentro de una sentencia switch como esta. La línea 16 no es una constante en tiempo de compilación, ya que se puede pasar cualquier valor int como parámetro. Marcarlo como final no cambia esto, por lo que no compila. La línea 18 no compila porque Sunday no está marcado como final. Ser efectivamente final es insuficiente. Finalmente, la línea 19 no compila porque DayOfWeek.MONDAY no es un valor int. Si bien las sentencias switch admiten valores enum, cada sentencia case debe tener el mismo tipo de datos que la variable switch otherDay, que es int. El resto de las líneas compilan. Dado que exactamente cuatro líneas no compilan, la opción E es la respuesta correcta.

11. What is the output of calling printLocation(Animal.MAMMAL)?

```
10: class Zoo {
11:   enum Animal {BIRD, FISH, MAMMAL}
12:   void printLocation(Animal a) {
13:     long type = switch(a) {
14:       case BIRD -> 1;
15:       case FISH -> 2;
16:       case MAMMAL -> 3;
17:       default -> 4;
18:     };
19:     System.out.print(type);
20:   } }
```

A. 3

B. 4

C. 34

D. The code does not compile because of line 13.

E. The code does not compile because of line 17.

F. None of the above

A

El código compila y se ejecuta sin problemas, imprimiendo 3 en tiempo de ejecución y haciendo que la opción A sea correcta. La sentencia default en la línea 17 es opcional ya que todos los valores de enum se tienen en cuenta y se pueden eliminar sin cambiar el resultado.

12. What is the result of the following code snippet?

```
3: int sing = 8, squawk = 2, notes = 0;
4: while(sing> squawk) {
5:   sing--;
6:   squawk += 2;
7:   notes += sing + squawk;
8: }
9: System.out.println(notes);
```

A. 11

B. 13

C. 23

D. 33

E. 50

F. The code will not compile because of line 7.

C

Antes de la primera iteración, sing = 8, squawk = 2 y notes = 0. Después de la iteración del primer bucle, sing se actualiza a 7, squawk a 4 y notes a la suma de los nuevos valores para sing + squawk, 7 + 4 = 11. Después de la iteración del segundo bucle, sing se actualiza a 6, squawk a 6 y notes a la suma de sí mismo más los nuevos valores para sing + squawk, 11 + 6 + 6 = 23. En la tercera iteración del bucle, sing > squawk se evalúa como falso, ya que 6 > 6 es falso. El bucle termina y se genera el valor más reciente de notes, 23, por lo que la respuesta correcta es la opción C.

13. What is the output of the following code snippet?

```
2: boolean keepGoing = true;
3: int result = 15, meters = 10;
4: do {
5:   meters--;
6:   if(meters==8) keepGoing = false;
7:   result -= 2;
8: } while keepGoing;
9: System.out.println(result);
```

A. 7

B. 9

C. 10

D. 11

E. 15

F. The code will not compile because of line 6.

G. The code does not compile for a different reason.

G

Este ejemplo puede parecer complicado, pero el código no compila. La línea 8 no tiene los paréntesis requeridos alrededor de la expresión condicional booleana. Dado que el código no compila y no es debido a la línea 6, la opción G es la respuesta correcta. Si la línea 8 se corrigiera con paréntesis, entonces el bucle se ejecutaría dos veces y la salida sería 11.

14. Which statements about the following code snippet are correct? (Choose all that apply.)

```
for(var penguin : new int[2])
  System.out.println(penguin);
  var ostrich = new Character[3];
  for(var emu : ostrich)
    System.out.println(emu);

  List<Integer> parrots = new ArrayList<Integer>();
  for(var macaw  : parrots)
    System.out.println(macaw);
```

A. The data type of penguin is Integer.

B. The data type of penguin is int.

C. The data type of emu is undefined.

D. The data type of emu is Character.

E. The data type of macaw is List.

F. The data type of macaw is Integer.

G. None of the above, as the code does not compile.

B, D, F

El código compila, lo que hace que la opción G sea incorrecta. En el primer bucle for-each, el lado derecho del bucle for-each tiene un tipo de int[], por lo que cada elemento penguin tiene un tipo de int, lo que hace que la opción B sea correcta. En el segundo bucle for-each, ostrich tiene un tipo de Character[], por lo que emu tiene un tipo de datos de Character, lo que hace que la opción D sea correcta. En el último bucle for-each, parrots tiene un tipo de datos de List<Integer>. Dado que el tipo genérico de Integer se usa en la Lista, macaw tendrá un tipo de datos de Integer, lo que hace que la opción F sea correcta.

15. What is the result of the following code snippet?

```
final char a = 'A', e = 'E';
char grade = 'B';
switch (grade) {
default:
case a:
case 'B': 'C': System.out.print("great ");
case 'D': System.out.print("good "); break;
case e:
case 'F': System.out.print("not good ");
}
```

A. great

B. great good

C. good

D. not good

E. The code does not compile because the data type of one or more case statements does not match the data type of the switch variable.

F. None of the above

F

El código no compila, aunque no por la razón especificada en la opción E. La segunda sentencia case contiene una sintaxis inválida. Cada sentencia case debe tener la palabra clave case; en otras palabras, no puedes encadenarlas con dos puntos (:). Por esta razón, la opción F es la respuesta correcta. Esta línea podría haberse corregido para decir case 'B', 'C' o agregando la palabra clave case antes de 'C'; entonces el resto del código habría compilado e impreso great good en tiempo de ejecución.

16. Given the following array, which code snippets print the elements in reverse order from how they are declared? (Choose all that apply.)

```
char[] wolf = {'W', 'e', 'b', 'b', 'y'};
```

A.
```
int q = wolf.length;
for( ; ; ) {
  System.out.print(wolf[--q]);
  if(q==0) break;
}
```

B.
```
for(int m=wolf.length-1; m>=0; --m)
  System.out.print(wolf[m]);
```

C.
```
for(int z=0; z<wolf.length; z++)
  System.out.print(wolf[wolf.length-z]);
```

D.
```
int x = wolf.length-1;
for(int j=0; x>=0 && j==0; x--)
  System.out.print(wolf[x]);
```

E.
```
final int r = wolf.length;
for(int w = r-1; r>-1; w = r-1)
  System.out.print(wolf[w]);
```

F.
```
for(int i=wolf.length; i>0; --i)
  System.out.print(wolf[i]);
```

G. None of the above

A, B, D

Para imprimir elementos en la matriz wolf en orden inverso, el código necesita comenzar con wolf[wolf.length-1] y terminar con wolf[0]. La opción A logra esto y es la primera respuesta correcta. La opción B también es correcta y es una de las formas más comunes en que se escribe un bucle inverso. La condición de terminación es a menudo m>=0 o m>-1, y ambas son correctas. Las opciones C y F causan una ArrayIndexOutOfBoundsException en tiempo de ejecución, ya que ambas leen primero desde wolf[wolf.length], con un índice que pasa la longitud de la matriz basada en 0 wolf. La forma de la opción C sería exitosa si el valor se cambiara a wolf[wolf.length-z-1]. La opción D también es correcta, ya que la j es extraña y se puede ignorar en este ejemplo. Finalmente, la opción E es incorrecta y produce un bucle infinito, ya que w se establece repetidamente en r-1, en este caso 4, en cada iteración del bucle. Dado que la sentencia de actualización no tiene ningún efecto después de la primera iteración, la condición nunca se cumple y el bucle nunca termina.

17. What distinct numbers are printed when the following method is executed? (Choose all that apply.)

```
private void countAttendees() {
 int participants = 4, animals = 2, performers = -1;

 while((participants = participants+1) < 10) {}
 do {} while (animals++ <= 1);
 for( ; performers<2; performers+=2) {}

 System.out.println(participants);
 System.out.println(animals);
 System.out.println(performers);
}
```

A. 6

B. 3

C. 4

D. 5

E. 10

F. 9

G. The code does not compile.

H. None of the above

B, E

El código compila sin problemas e imprime dos números distintos en tiempo de ejecución, por lo que las opciones G y H son incorrectas. El primer bucle se ejecuta un total de cinco veces, y el bucle termina cuando participants tiene un valor de 10. Por esta razón, la opción E es correcta. En el segundo bucle, animals comienza sin ser menor o igual que 1, pero como es un bucle do/while, se ejecuta al menos una vez. De esta manera, animals toma un valor de 3 y el bucle termina, lo que hace que la opción B sea correcta. Finalmente, el último bucle se ejecuta un total de dos veces, con performers comenzando con -1, yendo a 1 al final del primer bucle y luego terminando con un valor de 3 después del segundo bucle, lo que rompe el bucle. Esto hace que la opción B sea una respuesta correcta dos veces.

18. Which statements about pattern matching and flow scoping are correct? (Choose all that apply.)

A. Pattern matching with an if statement is implemented using the instance operator.

B. Pattern matching with an if statement is implemented using the instanceon operator.

C. Pattern matching with an if statement is implemented using the instanceof operator.

D. The pattern variable cannot be accessed after the if statement in which it is declared.

E. Flow scoping means a pattern variable is only accessible if the compiler can discern its type.

F. Pattern matching can be used to declare a variable with an else statement.

C, E

La coincidencia de patrones con una sentencia if se implementa utilizando el operador instanceof, lo que hace que la opción C sea correcta y las opciones A y B incorrectas. La opción D es incorrecta, ya que es posible acceder a una variable de patrón fuera de la sentencia if en la que está definida. La opción E es una afirmación correcta sobre el alcance del flujo. La opción F es incorrecta. La coincidencia de patrones no admite la declaración de variables en sentencias else, ya que las sentencias else no tienen una expresión booleana.

19. What is the output of the following code snippet?

```
2: double iguana = 0;
3: do {
4:   int snake = 1;
5:   System.out.print(snake++ + " ");
6:   iguana--;
7: } while (snake <= 5);
8: System.out.println(iguana);
```

A. 1 2 3 4 -4.0

B. 1 2 3 4 -5.0

C. 1 2 3 4 5 -4.0

D. 0 1 2 3 4 5 -5.0

E. The code does not compile.

F. The code compiles but produces an infinite loop at runtime.

G. None of the above

E

La variable snake se declara dentro del cuerpo de la sentencia do/while, por lo que está fuera de alcance en la línea 7. Por esta razón, la opción E es la respuesta correcta. Si snake se declarara antes de la línea 3 con un valor de 1, entonces la salida habría sido 1 2 3 4 5 -5.0, y la opción G habría sido la respuesta correcta.

20. Which statements, when inserted into the following blanks, allow the code to compile and run without entering an infinite loop? (Choose all that apply.)

```
4: int height = 1;
5: L1: while(height++ <10) {
6:   long humidity = 12;
7:   L2: do {
8:     if(humidity-- % 12 == 0) _____;
9:     int temperature = 30;
10:    L3: for( ; ; ) {
11:      temperature++;
12:      if(temperature>50) _____;
13:    }
14:  } while (humidity > 4);
15: }
```

A. break L2 on line 8; continue L2 on line 12

B. continue on line 8; continue on line 12

C. break L3 on line 8; break L1 on line 12

D. continue L2 on line 8; continue L3 on line 12

E. continue L2 on line 8; continue L2 on line 12

F. None of the above, as the code contains a compiler error

A, E

Lo más importante que hay que notar al leer este código es que el bucle más interno es un bucle infinito. Por lo tanto, se buscan soluciones que omitan el bucle más interno por completo o que salgan de ese bucle. La opción A es correcta, ya que break L2 en la línea 8 hace que el segundo bucle interno salga cada vez que se ingresa, omitiendo el bucle más interno por completo.

Para la opción B, el primer continue en la línea 8 hace que la ejecución omita el bucle más interno en la primera iteración del segundo bucle, pero no en la segunda iteración del segundo bucle. Se ejecuta el bucle más interno y, con continue en la línea 12, produce un bucle infinito en tiempo de ejecución, lo que hace que la opción B sea incorrecta. La opción C es incorrecta porque contiene un error de compilador. La etiqueta L3 no es visible fuera de su bucle. La opción D es incorrecta, ya que es equivalente a la opción B, ya que el break y el continue sin etiqueta se aplican al bucle más cercano y, por lo tanto, producen un bucle infinito en tiempo de ejecución.

Al igual que la opción A, el continue L2 en la línea 8 permite que el bucle más interno se ejecute la segunda vez que se llama al segundo bucle. Sin embargo, el continue L2 en la línea 12 sale del bucle infinito, lo que hace que el control regrese al segundo bucle. Dado que el primer y segundo bucle terminan, el código termina y la opción E es una respuesta correcta.

21. A minimum of how many lines need to be corrected before the following method will compile?

```
21: void findZookeeper(Long id) {
22:   System.out.print(switch(id) {
23:     case 10 -> {"Jane"}
24:     case 20 -> {yield "Lisa";};
25:     case 30 -> "Kelly";
26:     case 30 -> "Sarah";
27:     default -> "Unassigned";
28:   });
29: }
```

A. Zero

B. One

C. Two

D. Three

E. Four

F. Five

E

La línea 22 no compila porque Long no es un tipo compatible para una sentencia o expresión switch. La línea 23 no compila porque le falta un punto y coma después de "Jane" y una sentencia yield. La línea 24 no compila porque contiene un punto y coma adicional al final. Finalmente, las líneas 25 y 26 no compilan porque usan el mismo valor de caso. Al menos uno de ellos debería cambiarse para que el código compile. Dado que es necesario corregir cuatro líneas, la opción E es correcta.

22. What is the output of the following code snippet? (Choose all that apply.)

```
2: var tailFeathers = 3;
3: final var one = 1;
4: switch (tailFeathers) {
5:   case one: System.out.print(3 + " ");
6:   default: case 3: System.out.print(5 + " ");
7: }
8: while (tailFeathers> 1) {
9:   System.out.print(--tailFeathers + " "); }
```

A. 3

B. 5 1

C. 5 2

D. 3 5 1

E. 5 2 1

F. The code will not compile because of lines 3–5.

G. The code will not compile because of line 6.

E

El código compila sin problemas, lo que hace que las opciones F y G sean incorrectas. Recuerda que var es compatible tanto en bucles switch como while, siempre que el compilador determine que el tipo es compatible con estas sentencias. Además, la variable one está permitida en una sentencia case porque es una variable local final, lo que la convierte en una constante en tiempo de compilación. El valor de tailFeathers es 3, que coincide con la segunda sentencia case, lo que hace que 5 sea la primera salida. El bucle while se ejecuta dos veces, con el operador de preincremento (--) modificando el valor de tailFeathers de 3 a 2 y luego a 1 en el segundo bucle. Por esta razón, la salida final es 5 2 1, lo que hace que la opción E sea la respuesta correcta.

23. What is the output of the following code snippet?

```
15: int penguin = 50, turtle = 75;
16: boolean older = penguin>= turtle;
17: if (older = true) System.out.println("Success");
18: else System.out.println("Failure");
19: else if(penguin != 50) System.out.println("Other");
```

A. Success

B. Failure

C. Other

D. The code will not compile because of line 17.

E. The code compiles but throws an exception at runtime.

F. None of the above

F

La línea 19 comienza con una sentencia else, pero no hay una sentencia if precedente con la que coincida. Por esta razón, la línea 19 no compila, lo que hace que la opción F sea la respuesta correcta. Si se eliminara la palabra clave else de la línea 19, entonces el fragmento de código imprimiría Success.

24. Which of the following are possible data types for friends that would allow the code to compile? (Choose all that apply.)

```
for(var friend in friends) {
  System.out.println(friend);
}
```

A. Set

B. Map

C. String

D. int[]

E. Collection

F. StringBuilder

G. None of the above

G

La sentencia no es un bucle for-each válido (o un bucle for tradicional) ya que usa una palabra clave in inexistente. Por esta razón, el código no compila y la opción G es correcta. Si la palabra clave in se cambiara a dos puntos (:), entonces Set, int[] y Collection serían correctos.

25. What is the output of the following code snippet?

```
6:  String instrument = "violin";
7:  final String CELLO = "cello";
8:  String viola = "viola";
9:  int p = -1;
10: switch(instrument) {
11:    case "bass" : break;
12:    case CELLO : p++;
13:    default: p++;
14:    case "VIOLIN": p++;
15:    case "viola" : ++p; break;
16: }
17: System.out.print(p);
```

A. -1

B. 0

C. 1

D. 2

E. 3

F. The code does not compile.

D

El código compila sin problemas, por lo que la opción F es incorrecta. La variable viola creada en la línea 8 nunca se usa y se puede ignorar. Si se hubiera usado como el valor de caso en la línea 15, habría causado un error de compilación ya que no está marcada como final. Dado que "violin" y "VIOLIN" no coinciden exactamente, la rama predeterminada de la sentencia switch se ejecuta en tiempo de ejecución. Esta ruta de ejecución incrementa p un total de tres veces, lo que lleva el valor final de p a 2 y hace que la opción D sea la respuesta correcta.

26. What is the output of the following code snippet? (Choose all that apply.)

```
9:  int w = 0, r = 1;
10: String name = "";
11: while(w < 2) {
12:   name += "A";
13:   do {
14:     name += "B";
15:     if(name.length()>0) name += "C";
16:     else break;
17:   } while (r <=1);
18:   r++; w++; }
19: System.out.println(name);
```

A. ABC

B. ABCABC

C. ABCABCABC

D. Line 15 contains a compilation error.

E. Line 18 contains a compilation error.

F. The code compiles but never terminates at runtime.

G. The code compiles but throws a NullPointerException at runtime.

F

El fragmento de código no contiene ningún error de compilación, por lo que las opciones D y E son incorrectas. Sin embargo, hay un problema con este fragmento de código. Aunque puede parecer complicado, la clave es notar que la variable r se actualiza fuera del bucle do/while. Esto está permitido desde el punto de vista de la compilación, ya que está definido antes del bucle, pero significa que el bucle más interno nunca rompe la condición de terminación r <= 1. En tiempo de ejecución, esto producirá un bucle infinito la primera vez que se ingrese al bucle más interno, lo que hace que la opción F sea la respuesta correcta.

27. What is printed by the following code snippet?

```
23: byte amphibian = 1;
24: String name = "Frog";
25: String color = switch(amphibian) {
26:   case 1 -> { yield "Red"; }
27:   case 2 -> { if(name.equals("Frog")) yield "Green"; }
28:   case 3 -> { yield "Purple"; }
29:   default -> throw new RuntimeException();
30: };
31: System.out.print(color);
```

A. Red

B. Green

C. Purple

D. RedPurple

E. An exception is thrown at runtime.

F. The code does not compile.

F

La línea 27 no compila porque el bloque case no produce un valor si name no es igual a Frog. Por esta razón, la opción F es correcta. Cada ruta dentro de un bloque case debe producir un valor si se espera que la expresión switch devuelva un valor.

28. What is the output of calling getFish("goldie")?

```
40: void getFish(Object fish) {
41:   if (!(fish instanceof String guppy))
42:     System.out.print("Eat!");
43:   else if (!(fish instanceof String guppy)) {
44:     throw new RuntimeException();
45:   }
46:   System.out.print("Swim!");
47: }
```

A. Eat!

B. Swim!

C. Eat! followed by an exception.

D. Eat!Swim!

E. An exception is printed.

F. None of the above

F

Basado en el alcance del flujo, guppy está en el ámbito después de las líneas 41-42 si el tipo no es una cadena. En este caso, la línea 43 declara una variable guppy que es un duplicado de la variable local definida previamente en la línea 41. Por esta razón, el código no compila y la opción F es correcta. Si se usara un nombre de variable diferente en la línea 43, entonces el código compilaría e imprimiría ¡Nada! en tiempo de ejecución con la entrada especificada.

29. What is the result of the following code?

```
1: public class PrintIntegers {
2:   public static void main(String[] args) {
3:     int y = -2;
4:     do System.out.print(++y + " ");
5:     while(y <= 5);
6: } }
```

A. -2 -1 0 1 2 3 4 5

B. -2 -1 0 1 2 3 4

C. -1 0 1 2 3 4 5 6

D. -1 0 1 2 3 4 5

E. The code will not compile because of line 5.

F. The code contains an infinite loop and does not terminate.

C

Dado que se utilizó el operador de preincremento, el primer valor que se mostrará es -1, por lo que las opciones A y B son incorrectas. En la penúltima iteración del bucle, y se incrementará a 5, y el bucle mostrará 5. El bucle continuará ya que 5 <= 5 es verdadero, y en la última iteración, se mostrará 6. Al final de esta última iteración, la expresión booleana 6 <= 5 se evaluará como falsa y el bucle terminará. Dado que 6 fue el último valor mostrado por el bucle, la respuesta es la opción C.