

Academia Java

CAPITULO 3

Esthefanny De La Cruz Carbajal

Esthefanny De La Cruz
15-1-2025

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

Int, enum, string, char, byte

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.

Análisis paso a paso:

Inicialización de variables:

temperature = 4 (de tipo int).

humidity = -temperature + temperature \* 3:

humidity = -(4) + 4 \* 3 = -4 + 12 = 8

Por lo tanto, humidity = 8 (de tipo long).

Primera condición if:

if (temperature >= 4):

Como temperature = 4, esta condición es verdadera.

Segunda condición if:

if (humidity < 6):

Como humidity = 8, esta condición es falsa. Por lo tanto, el programa ejecuta el bloque else correspondiente a este if:

```
System.out.println("Just Right");
```

El else de la línea 8:

El else en la línea 8 está asociado con el primer if en la línea 5.

Sin embargo, como la primera condición (temperature >= 4) es verdadera, este bloque else nunca se ejecuta.

**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

**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

No hacer

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 código no compila en la línea del `continue`, ya que todo lo demás de código que está escrito después de esta línea no está alcanzable.

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.

**Respuestas: C,E,D**

C. La expresión condicional de un bucle `for` se evalúa antes de la primera ejecución del cuerpo del bucle.

En un bucle for, la condición se evalúa antes de cada iteración, incluida la primera. Si la condición es false, el cuerpo del bucle no se ejecuta.

E. El cuerpo de un bucle do/while está garantizado que se ejecute al menos una vez.

Un bucle do/while evalúa la condición después de ejecutar el cuerpo del bucle, lo que garantiza que se ejecute al menos una vez.

D. Aunque es una buena práctica incluir una rama default en un switch para manejar casos no previstos, no es obligatorio. El código compilará y ejecutará sin un default si todos los posibles valores del String están cubiertos en las ramas.

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

**Respuesta B,D**

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

Explicación: Este es un bucle for clásico para recorrer un arreglo desde el índice 0 hasta el último índice (que es `weather.length - 1`). La condición `i <= weather.length - 1` asegura que el índice esté dentro de los límites del arreglo.

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

Explicación: Este bucle recorre el arreglo en orden inverso, comenzando desde el último índice (`weather.length - 1`) y disminuyendo hasta 0. A pesar de ser inverso, sigue imprimiendo todos los elementos del arreglo.

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: }
```

Review Que:

- 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

Este no.

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,E,C**

El código de RABBIT solo se ejecutaría una vez , en la primera iteración de l for al entrar a if, es válido el break con y sin RABBIT.

La opción C (continue BUNNY) es correcta, porque al usar continue BUNNY, el ciclo exterior salta a la siguiente fila, evitando incrementar count innecesariamente en las iteraciones posteriores de cada fila. Esto hace que count sea igual a 2.

continue: Esto haría que el bucle interno salte la iteración y continúe con la siguiente, sin incrementar count cuando no se cumple la condición. Sin embargo, no evitaría que count se incremente correctamente en las iteraciones donde la condición se cumple.

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. Hay 4 errores.

Línea 15 (case 1: continue;)

El uso de continue dentro de un switch es incorrecto. continue solo puede usarse dentro de bucles (for, while, do-while), no dentro de un switch. Esto generará un error de compilación.

Línea 17 (case 2,10: break;)

En un switch, no se pueden tener múltiples valores en una sola etiqueta de case, como case 2, 10. Cada valor debe estar separado y tener su propia instrucción case, o puede usarse una lista de valores en una sola etiqueta solo si están en un enum o tipo compatible. Esto también genera un error de compilación.

Línea 18 (case Sunday: return DayOfWeek.SUNDAY;)



La variable Sunday es una variable de tipo int y no un valor literal, y por tanto no se puede usar como una etiqueta case en el switch. Las etiquetas de case deben ser constantes literales o valores que son determinables en tiempo de compilación. Usar una variable como etiqueta en un case genera un error.

Si int estuviera marcado como final entonces compilaría

Línea 19

No compila porque DayOfWeek.MONDAY no es un valor int. Aunque las sentencias switch admiten valores enum, cada sentencia case debe tener el mismo tipo de datos que la variable switch

```
1 package com.curso.v0;
2
3 import java.time.DayOfWeek;
4
5 public class Question10 {
6
7 private DayOfWeek getWeekDay(int day, final int thursday) {
8
9 int otherDay = day;
10 int Sunday = 0;
11 final int jueves = 2;
12
13 switch(otherDay) {
14 default:
15 case 1: return DayOfWeek.MONDAY; //1
16 case thursdayjue:
17 }
18
19 }
20 }
21 }
```

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

Este no

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

El while se ejecuta dos veces.

Antes de la ejecución los valores son:

antes de ejecutar el cuerpo del bucle:

sing = 8

squawk = 2

primera vez Dentro del bucle:

sing--: sing = 7

squawk += 2: squawk = 4

notes += sing + squawk:

notes = 0 + (7 + 4) = 11

antes de ejecutar el cuerpo del bucle:

sing = 7

squawk = 4

Dentro del bucle:

sing--: sing = 6

squawk += 2: squawk = 6

notes += sing + squawk:

notes = 11 + (6 + 6) = 23

Iteracion 3:

Sing=6

Squak= 6

Condición del while: falso no se ejecuta por 3ra vez

Notes se queda con el valor de 23

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**

Faltan paréntesis en la línea 8, para que la sintaxis sea correcta.

---

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

`new int[2]` es un arreglo de enteros primitivos (`int[]`). Aquí, `penguin` es de tipo `int` Y SIGUIENDO la misma pauta para las demás opciones tenemos que:

`ostrich` es un arreglo de tipo `Character[]`.

`emu` es de tipo `Character`.

`parrots` es una lista genérica de tipo `List<Integer>`

`macaw` es de tipo `Integer`

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 pero porque la sintaxis en `case 'B' ...` está mal

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]);
```

Review Questions

149

**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

A

Aunque ese ciclo for es infinito, break hace que salga del ciclo cuando q=0

B

Es la forma común para recorrer un array en orden inverso

D

x controla al ciclo for y va en orden inverso.

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

E,B,

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

 // WHILE LOOP
 while ((participants = participants + 1) < 10) {}

 // DO-WHILE LOOP
 do {} while (animals++ <= 1);

 // FOR LOOP
 for (; performers < 2; performers += 2) {}

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

El primer while corre 6 veces y al final participants tiene un valor de 10.

el do while es falso desde la primera iteración pero deja a animals con valor de 3

el for itera 2 veces y en la 3ra se vuelve falso pero imprime 3

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.

No se resuelve

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

La respuesta E.

El código no compila porque la variable `snake` debería vivir fuera del `do while`.

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: }
```

Review Questions

1

- 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

Respuesta: A,E . MARCADA PARA REPASAR.

Para que el código compile y se ejecute sin entrar en un bucle infinito, es necesario insertar declaraciones que permitan salir de los bucles internos en los momentos adecuados. Analicemos las opciones proporcionadas:

A. break L2 en la línea 8; continue L2 en la línea 12

- break L2 en la línea 8: Esta instrucción termina el bucle do-while etiquetado como L2, lo que evita un posible bucle infinito si se cumplen ciertas condiciones.
- continue L2 en la línea 12: Esta instrucción salta a la siguiente iteración del bucle do-while etiquetado como L2, permitiendo que el bucle continúe correctamente.

Conclusión: Esta combinación permite que el código compile y se ejecute sin entrar en un bucle infinito.

E. continue L2 en la línea 8; continue L2 en la línea 12

- continue L2 en la línea 8: Esta instrucción salta a la siguiente iteración del bucle do-while etiquetado como L2, asegurando que el bucle continúe correctamente.
- continue L2 en la línea 12: Al igual que la anterior, esta instrucción salta a la siguiente iteración del bucle do-while etiquetado como L2, manteniendo el flujo del bucle.

Conclusión: Esta combinación también permite que el código compile y se ejecute sin entrar en un bucle infinito.

Otras opciones:

- B. continue en la línea 8; continue en la línea 12: Estas instrucciones afectan solo al bucle más interno en el que se encuentran. Sin etiquetas, no controlan adecuadamente los bucles externos, lo que puede llevar a un bucle infinito.
- C. break L3 en la línea 8; break L1 en la línea 12: Estas instrucciones terminan los bucles etiquetados como L3 y L1 respectivamente. Sin embargo, dependiendo de la lógica del programa, esto podría interrumpir el flujo esperado del código.
- D. continue L2 en la línea 8; continue L3 en la línea 12: La instrucción continue L3 intenta continuar un bucle for infinito (L3), lo que no es posible y resultaría en un error de compilación.

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

No se hace

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.

- 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

C

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

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

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

```

3 public class Question22 {
4     public static void main(String[] args) {
5         int tailFeathers = 3;
6         final int one = 1;
7
8         switch (tailFeathers) {
9             case one: //1
10                 System.out.print(3 + " ");
11                 break;
12             default:
13                 case 3:
14                     System.out.print(5 + " ");
15                 }
16
17         while (tailFeathers > 1) {
18             System.out.print(--tailFeathers + " ");
19         }
20     }
21 }

```

el código compila sin errores,

primero entra al caso 3 e imprime 5

luego continua al while , en este tailfeathers se vuelve 2 en la primera iteración.

en la segunda iteración se vuelve 1 y ya no hay tercera iteración

quedando impresos 5 2 1

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

Este ejercicio no tiene la estructura correcta de if, ifelse, if, por lo cual la respuesta correcta es D

Problemas en el código:

Línea 17 (if (older = true)):

Aquí ocurre un error lógico que también es un error de compilación. El operador = es de asignación, no de comparación. La intención parece ser comparar si older es true, lo cual debería hacerse con == (es decir, if (older == true) o simplemente if (older)). Sin embargo, el

uso de = en esta línea provoca un error de compilación, ya que no es válido utilizar una asignación dentro de una condición de un if en Java.

Línea 19:

La línea 19 tiene otro error. Un bloque else no puede ir seguido directamente de un else if.

La estructura correcta sería colocar un bloque if, seguido de else if y, opcionalmente, un else. Esto también provoca un error de compilación.

Pero como el código se ejecuta en orden, se termina en la línea 17.

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

A,D,E

Estos tipos de datos permiten iterable .

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.

```
String viola = "viola";
int p = -1;

switch (instrument) { //violin
case "bass":
    break;
case CELLO:
    p++;
default:
    p++; //0
    //break;
case "VIOLIN":
    p++; //1
case "viola":
    ++p; //2
    break;
}
System.out.print(p);
}
```

Ya que después de `p++` en el caso default no tiene `break` va a iniciar a ejecutar , las instrucciones siguientes, entonces se asignan valores de 0, 1, 2.

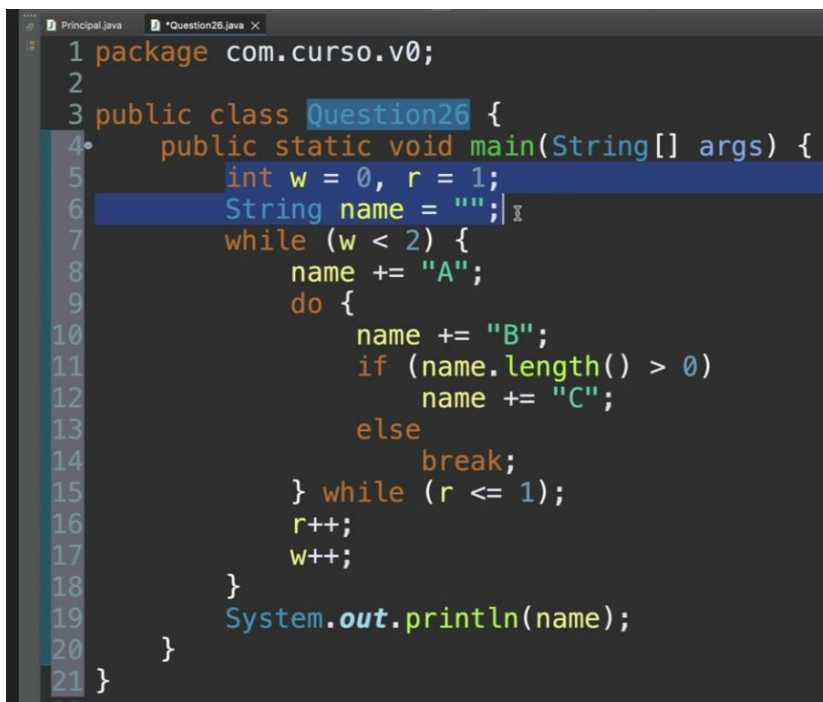
Respuesta: D

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.

Respuesta F



```
1 package com.curso.v0;
2
3 public class Question26 {
4     public static void main(String[] args) {
5         int w = 0, r = 1;
6         String name = "";
7         while (w < 2) {
8             name += "A";
9             do {
10                 name += "B";
11                 if (name.length() > 0)
12                     name += "C";
13                 else
14                     break;
15             } while (r <= 1);
16             r++;
17             w++;
18         }
19         System.out.println(name);
20     }
21 }
```

Inicialización:

w = 0, r = 1, name = "".

Primera iteración del while (línea 11):

Condición: $w < 2$ (es verdadera porque $w = 0$).

Línea 12: `name += "A"` → `name = "A"`.

Línea 13: Entra en el do-while.

Línea 14: `name += "B"` → `name = "AB"`.

Línea 15: `name.length() > 0` (es verdadera) → `name += "C"` → `name = "ABC"`.

Línea 17: Verifica $r \leq 1$ (es verdadera porque $r = 1$), así que el bucle se repite.

Línea 14: `name += "B"` → `name = "ABCB"`.

Línea 15: `name.length() > 0` → `name += "C"` → `name = "ABCBBC"`.

Línea 17: Verifica $r \leq 1$ (es verdadera). El bucle do-while nunca sale porque la variable r no se modifica dentro del bucle y siempre cumple la condición $r \leq 1$.

Problema identificado:

El bucle do-while (líneas 13-17) es infinito porque r no cambia y siempre cumple $r \leq 1$.

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);
```

154 Chapter 3 • Making Decisions

- A.** Red
- B.** Green
- C.** Purple
- D.** RedPurple
- E.** An exception is thrown at runtime.
- F.** The code does not compile.

Este no

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

El código tiene un error de compilación. Esto ocurre porque dentro de un bloque if con un patrón de coincidencia (instanceof), la variable declarada (en este caso, guppy) no puede redefinirse dentro del mismo bloque o en un bloque relacionado, como otro if o else if.

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.

D

el código compil y se ejecuta correctamente, luego ++y=-1 y de ahí itera hasta 5