Academia Java

CAPITULO 4

Esthefanny De La Cruz Carbajal

VOSTRO 3400
15-1-2025

The answers to the chapter review questions can be found in the Appendix.

1. What is output by the following code? (Choose all that apply.)

```
1: public class Fish {
2:    public static void main(String[] args) {
3:       int numFish = 4;
4:       String fishType = "tuna";
5:       String anotherFish = numFish + 1;
6:       System.out.println(anotherFish + " " + fishType);
7:       System.out.println(numFish + " " + 1);
8: } }
```

A. `4 1`
B. `5`
C. `5 tuna`
D. `5tuna`
E. `51tuna`
F. The code does not compile.

F

En la línea 5, no se puede asignar un int a un string.

2. Which of these array declarations are not legal? (Choose all that apply.)

A. `int[][] scores = new int[5][];`
B. `Object[][][] cubbies = new Object[3][0][5];`
C. `String beans[] = new beans[6];`
D. `java.util.Date[] dates[] = new java.util.Date[2][];`
E. `int[][] types = new int[];`
F. `int[][] java = new int[][];`

Las declaraciones de arreglos que no son legales son las siguientes:

C. String beans[] = new beans[6];

- No es legal porque beans es una clase no definida. Si se quiso declarar un arreglo de tipo String, debería ser String beans[] = new String[6];.

E. int[][] types = new int[];

- No es legal porque al declarar un arreglo bidimensional (int[][]), se debe especificar al menos el tamaño del primer nivel. Por ejemplo, new int[5][] sería válido.

F. int[][] java = new int[][];

- No es legal porque aunque int[][] es una declaración válida, cuando se utiliza la inicialización new, se deben especificar los tamaños o proporcionar los elementos explícitamente. Por ejemplo, new int[5][5] o {{1, 2}, {3, 4}}.

Respuestas: C, E, F.

3. Note that March 13, 2022 is the weekend when we spring forward, and November 6, 2022 is when we fall back for daylight saving time. Which of the following can fill in the blank without the code throwing an exception? (Choose all that apply.)

```
var zone = ZoneId.of("US/Eastern");
var date = _____________________________;
var time = LocalTime.of(2, 15);
var z = ZonedDateTime.of(date, time, zone);
```

A. `LocalDate.of(2022, 3, 13)`
B. `LocalDate.of(2022, 3, 40)`
C. `LocalDate.of(2022, 11, 6)`

D. `LocalDate.of(2022, 11, 7)`
E. `LocalDate.of(2023, 2, 29)`
F. `LocalDate.of(2022, MonthEnum.MARCH, 13);`

A. LocalDate.of(2022, 3, 13)

- Válido. Aunque el 13 de marzo de 2022 es el día en que cambia el horario (spring forward) en la zona “US/Eastern”, Java maneja correctamente el salto de hora.

Resultado: No lanza excepción.

B. LocalDate.of(2022, 3, 40)

- Inválido. Marzo solo tiene 31 días. Este intento generará una excepción DateTimeException por ser una fecha no válida.

C. LocalDate.of(2022, 11, 6)

- Válido. El 6 de noviembre de 2022 es el día en que termina el horario de verano (fall back) en “US/Eastern”. Java maneja correctamente el ajuste de hora adicional.

Resultado: No lanza excepción.

D. LocalDate.of(2022, 11, 7)

- Válido. Es una fecha válida y no está afectada por cambios en el horario de verano.

Resultado: No lanza excepción.

E. LocalDate.of(2023, 2, 29)

- Inválido. El año 2023 no es bisiesto, por lo que febrero tiene solo 28 días. Intentar usar el día 29 generará una excepción DateTimeException.

F. LocalDate.of(2022, MonthEnum.MARCH, 13)

- Válido. Es una forma alternativa de declarar el 13 de marzo de 2022 utilizando MonthEnum.

Resultado: No lanza excepción.

Respuestas válidas: A, C, D, F.

En resumen java maneja bien los cambios de horario sin excepción, sin alguna instrucción extra.

4. Which of the following are output by this code? (Choose all that apply.)

```
3: var s = "Hello";
4: var t = new String(s);
5: if ("Hello".equals(s)) System.out.println("one");
6: if (t == s) System.out.println("two");
7: if (t.intern() == s) System.out.println("three");
8: if ("Hello" == s) System.out.println("four");
9: if ("Hello".intern() == t) System.out.println("five");
```

A. one
B. two
C. three
D. four
E. five
F. The code does not compile.
G. None of the above

POOL DE CADENAS

Análisis del código línea por línea:

3: var s = "Hello";

- Se declara s como un literal de cadena en el pool de cadenas de Java.

4: var t = new String(s);

- Se crea un nuevo objeto String con el valor "Hello" pero fuera del pool de cadenas, porque se usa el constructor explícito new String().

5: if ("Hello".equals(s)) System.out.println("one");

- Verdadero. El método equals() compara el contenido de las cadenas. "Hello" es igual a s.

→ Se imprime “one”.

6: if (t == s) System.out.println("two");

- Falso. El operador == compara referencias, no valores. t y s apuntan a objetos diferentes, aunque sus contenidos sean iguales.

→ No se imprime “two”.

7: if (t.intern() == s) System.out.println("three");

- Verdadero. El método intern() devuelve la referencia del literal "Hello" en el pool de cadenas. Como s ya apunta al mismo literal, ambas referencias son iguales.

→ Se imprime “three”.

8: if ("Hello" == s) System.out.println("four");

- Verdadero. "Hello" y s apuntan al mismo literal en el pool de cadenas, por lo que sus referencias son iguales.

→ Se imprime “four”.

9: if ("Hello".intern() == t) System.out.println("five");

- Falso. "Hello".intern() devuelve la referencia en el pool de cadenas, pero t apunta a un objeto distinto creado fuera del pool (por new String()).

→ No se imprime “five”.

Respuestas : A,C,D

5. What is the result of the following code?

```
7: var sb = new StringBuilder();
8: sb.append("aaa").insert(1, "bb").insert(4, "ccc");
9: System.out.println(sb);
```

A. abbaaccc
B. abbaccca
C. bbaaaccc
D. bbaaccca
E. An empty line
F. The code does not compile.

Tomando en cuenta que StringBuilder inicia en posición 0

Paso 1:

- sb.append("aaa") agrega "aaa" al objeto.
- Resultado: "aaa".

Paso 2:

- .insert(1, "bb") inserta "bb" en la posición 1 (después del primer carácter).
- Resultado: "abba".

Paso 3:

- .insert(4, "ccc") inserta "ccc" en la posición 4 (después de "abba").
- Resultado: "abbaccc".

B

6. How many of these lines contain a compiler error? (Choose all that apply.)

```
23: double one = Math.pow(1, 2);
24: int two = Math.round(1.0);
25: float three = Math.random();
26: var doubles = new double[] {one, two, three};
```

A. 0
B. 1

C. 2
D. 3
E. 4

24: int two = Math.round(1.0);

- Error. Math.round() devuelve un long, no un int. No se puede asignar directamente un long a una variable int sin conversión explícita.

Solución: Usar (int) Math.round(1.0) o cambiar two a long.

25: float three = Math.random();

- Error. Math.random() devuelve un double, y three es de tipo float. Se necesita una conversión explícita para asignar un double a un float.

Solución: Usar float three = (float) Math.random();.

26: La línea es correcta pero falla porque depende de las líneas 25 y 26

Respuesta C

7. Which of these statements is true of the two values? (Choose all that apply.)

```
2022-08-28T05:00 GMT-04:00
2022-08-28T09:00 GMT-06:00
```

A. The first date/time is earlier.
B. The second date/time is earlier.
C. Both date/times are the same.
D. The date/times are two hours apart.
E. The date/times are six hours apart.
F. The date/times are 10 hours apart.

A,E

1. 2022-08-28T05:00 GMT-04:00

- Este valor corresponde al 28 de agosto de 2022 a las 5:00 AM, en una zona horaria que está 4 horas detrás de GMT (hora estándar de la zona GMT-04:00).

2. 2022-08-28T09:00 GMT-06:00

- Este valor corresponde al 28 de agosto de 2022 a las 9:00 AM, en una zona horaria que está 6 horas detrás de GMT (hora estándar de la zona GMT-06:00).

Paso 1: Convertir ambas fechas a la hora GMT

- 2022-08-28T05:00 GMT-04:00

- Si esta hora es GMT-04:00, la hora en GMT será 2022-08-28T09:00 GMT (sumamos 4 horas).

- 2022-08-28T09:00 GMT-06:00

- Si esta hora es GMT-06:00, la hora en GMT será 2022-08-28T15:00 GMT (sumamos 6 horas).

Como podemos ver, 2022-08-28T09:00 GMT es 6 horas más temprano que 2022-08-28T15:00 GMT.

8. Which of the following return 5 when run independently? (Choose all that apply.)

```
var string = "12345";
var builder = new StringBuilder("12345");
```

A. `builder.charAt(4)`
B. `builder.replace(2, 4, "6").charAt(3)`
C. `builder.replace(2, 5, "6").charAt(2)`
D. `string.charAt(5)`
E. `string.length`
F. `string.replace("123", "1").charAt(2)`
G. None of the above

A,B,F

A regresa 5 ya que busca la posición 4 del string y el indice comienza en 0.

B. Builder.replace(2,4,”6”) sustituye lo que hay de la posición 2 a la 4 de builder por 6 y luego con charAt pide la posición 3 , la cual es 5

E. Regresa 5 pero de tipo int

F. Caso similar que busca “123” en string y sustituye todo eso por 1, en este caso queda ”145”, luego solicita la posición 2 que es 5

9. Which of the following are true about arrays? (Choose all that apply.)

A. The first element is index 0.
B. The first element is index 1.
C. Arrays are fixed size.
D. Arrays are immutable.
E. Calling `equals()` on two different arrays containing the same primitive values always returns `true`.
F. Calling `equals()` on two different arrays containing the same primitive values always returns `false`.
G. Calling `equals()` on two different arrays containing the same primitive values can return `true` or `false`.

A,C,F

Siempre da falso ya que equals() compara la referencia de cada elemento del array

10. How many of these lines contain a compiler error? (Choose all that apply.)

```
23: int one = Math.min(5, 3);
24: long two = Math.round(5.5);
25: double three = Math.floor(6.6);
26: var doubles = new double[] {one, two, three};
```

A. 0
B. 1
C. 2
D. 3
E. 4

A

Todas las líneas son correctas

11. What is the output of the following code?

```
var date = LocalDate.of(2022, 4, 3);
date.plusDays(2);
date.plusHours(3);
System.out.println(date.getYear() + " " + date.getMonth()
    + " " + date.getDayOfMonth());
```

A. `2022 MARCH 4`
B. `2022 MARCH 6`
C. `2022 APRIL 3`
D. `2022 APRIL 5`
E. The code does not compile.
F. A runtime exception is thrown.

E.

A pesar de que se suman 2 días, esto no modifica la variable original.

El código no compila ya que plusHours no existe para LocalDate.of

12. What is output by the following code? (Choose all that apply.)

```
var numbers = "012345678".indent(1);
numbers = numbers.stripLeading();
System.out.println(numbers.substring(1, 3));
System.out.println(numbers.substring(7, 7));
System.out.print(numbers.substring(7));
```

A. 12
B. 123
C. 7
D. 78
E. A blank line
F. The code does not compile.
G. An exception is thrown.

---

var numbers = "012345678".indent(1); // Indenta la cadena, agregando un espacio al inicio.

numbers = numbers.stripLeading(); // Elimina el espacio del inicio de la cadena.

System.out.println(numbers.substring(1, 3)); // Extrae y muestra la subcadena de índices 1 a 2. Imprime 12

System.out.println(numbers.substring(7, 7)); // Extrae la subcadena desde el índice 7 (vacía). Imprime línea en blanco

System.out.print(numbers.substring(7)); // Muestra la subcadena desde el índice 7 hasta el final. Imprime 78

A,E,D

13. What is the result of the following code?

```
public class Lion {
   public void roar(String roar1, StringBuilder roar2) {
      roar1.concat("!!!");
      roar2.append("!!!");
   }
   public static void main(String[] args) {
      var roar1 = "roar";
      var roar2 = new StringBuilder("roar");
      new Lion().roar(roar1, roar2);
      System.out.println(roar1 + " " + roar2);
} }
```

A. roar roar
B. roar roar!!!
C. roar!!! roar
D. roar!!! roar!!!
E. An exception is thrown.
F. The code does not compile.

B

```
public class Lion {
    public void roar(String roar1, StringBuilder roar2) {
        roar1.concat("!!!");  // Concatena "!!!" a roar1, pero no lo guarda
        roar2.append("!!!");  // Modifica roar2 directamente, añadiendo "!!!"
    }

    public static void main(String[] args) {
        var roar1 = "roar";  // Declara y asigna el valor "roar" a roar1
        var roar2 = new StringBuilder("roar");  // Declara y asigna el valor "
        new Lion().roar(roar1, roar2);  // Llama al método roar
        System.out.println(roar1 + " " + roar2);  // Imprime roar1 y roar2
    }
}
```

---

14. Given the following, which can correctly fill in the blank? (Choose all that apply.)

```
var date = LocalDate.now();
var time = LocalTime.now();
var dateTime = LocalDateTime.now();
var zoneId = ZoneId.systemDefault();
var zonedDateTime = ZonedDateTime.of(dateTime, zoneId);
Instant instant = _____;
```

A. `Instant.now()`
B. `new Instant()`
C. `date.toInstant()`
D. `dateTime.toInstant()`
E. `time.toInstant()`
F. `zonedDateTime.toInstant()`

A,F

1. A. Instant.now()

   - Correcta.

Este método genera un Instant válido que representa el momento actual en UTC.

2. B. new Instant()

   - Incorrecta.

La clase Instant no tiene un constructor público; debe utilizarse uno de sus métodos estáticos como Instant.now().

3. C. date.toInstant()

   - Incorrecta.

La clase LocalDate no tiene un método toInstant(). Para convertir una fecha a un instante, es necesario proporcionar también una hora y una zona horaria.

4. D. dateTime.toInstant()

- Incorrecta.

LocalDateTime no tiene información de zona horaria, por lo que no se puede convertir directamente a un Instant. Se necesita usar un ZonedDateTime.

5. E. time.toInstant()

- Incorrecta.

LocalTime representa solo una hora sin fecha ni zona horaria, y no tiene un método toInstant().

6. F. zonedDateTime.toInstant()

- Correcta.

Un ZonedDateTime incluye información de fecha, hora y zona horaria, por lo que puede convertirse directamente en un Instant.

15. What is the output of the following? (Choose all that apply.)

```
var arr = new String[] { "PIG", "pig", "123"};
Arrays.sort(arr);
System.out.println(Arrays.toString(arr));
System.out.println(Arrays.binarySearch(arr, "Pippa"));
```


A. `[pig, PIG, 123]`
B. `[PIG, pig, 123]`
C. `[123, PIG, pig]`
D. `[123, pig, PIG]`
E. `-3`
F. `-2`
G. The results of `binarySearch()` are undefined in this example.

Respuesta : C,E

var arr = new String[] { "PIG", "pig", "123" }; // Declaración e inicialización del array

Arrays.sort(arr); // Ordena el array alfabéticamente como. 123 PIG Pig

System.out.println(Arrays.toString(arr)); // Imprime el array ordenado

System.out.println(Arrays.binarySearch(arr, "Pippa")); // Realiza una búsqueda binaria para "Pippa"

Arrays.binarySearch(arr, "Pippa") realiza una búsqueda binaria del valor "Pippa". La búsqueda binaria devuelve un valor negativo cuando el elemento no se encuentra en el array, y el valor negativo representa la posición en la que se debería insertar el elemento para mantener el orden del array.

El array ordenado es:

["123", "PIG", "pig"]

El valor "Pippa" no se encuentra en el array, y en la búsqueda binaria se determina que "Pippa" debería estar en la posición 2 (después de "PIG" pero antes de "pig"). La búsqueda binaria devuelve el valor negativo de esta posición menos 1, que es -3.

16. What is included in the output of the following code? (Choose all that apply.)

```
var base = "ewe\nsheep\\t";
int length = base.length();
int indent = base.indent(2).length();
int translate = base.translateEscapes().length();

var formatted = "%s %s %s".formatted(length, indent, translate);
System.out.format(formatted);
```

A. 10
B. 11
C. 12
D. 13
E. 14
F. 15
G. 16

Variables del código:

1. base:

El valor de base es:

"ewe\nsheep\\t"

- \n representa un salto de línea.
- \\t representa una barra invertida (\) seguida de una “t”.

Longitud de base:

- "ewe\nsheep\\t" contiene:
- 3 caracteres en ewe.
- 1 carácter por \n (es tratado como un único carácter de escape).
- 5 caracteres en sheep.
- 2 caracteres en \\t (la barra invertida y la “t”).
- Longitud total: 10

El método translateEscapes() convierte cualquier carácter de escape de texto en caracteres de escape reales, convirtiendo \\t en \t.

2. length:

int length = base.length();

- length será 11, como se explicó arriba.

3. indent:

int indent = base.indent(2).length();

- El método indent(2) agrega 2 espacios al principio de cada línea del texto.
- El texto ahora será:

" ewe

 sheep\t"

- Cálculo de longitud después de indent:
- 2 espacios agregados antes de “ewe”.
- El salto de línea \n permanece.
- 2 espacios agregados antes de “sheep\t”.

- Nueva longitud: 15 (11 + 4 espacios añadidos).

4. translate:

int translate = base.translateEscapes().length();

- El método translateEscapes() interpreta los caracteres de escape en el texto:
- \n se convierte en un salto de línea real.
- \\t se convierte en un carácter de tabulación real (\t).
- El texto traducido será:

"ewe

sheep<TAB>"

- Cálculo de longitud después de translateEscapes():
- "ewe": 3 caracteres.
- \n: no cuenta como carácter visible, pero separa líneas.
- "sheep": 5 caracteres.
- \t: es un carácter de tabulación (1 carácter).
- Longitud total: 9.

5. formatted:

var formatted = "%s %s %s".formatted(length, indent, translate);

- Sustituye los valores de length, indent, y translate en el formato.
- Resultado: "10 15 9".

17. Which of these statements are true? (Choose all that apply.)

```
var letters = new StringBuilder("abcdefg");
```

A. `letters.substring(1, 2)` returns a single-character `String`.
B. `letters.substring(2, 2)` returns a single-character `String`.
C. `letters.substring(6, 5)` returns a single-character `String`.
D. `letters.substring(6, 6)` returns a single-character `String`.
E. `letters.substring(1, 2)` throws an exception.
F. `letters.substring(2, 2)` throws an exception.
G. `letters.substring(6, 5)` throws an exception.
H. `letters.substring(6, 6)` throws an exception.

A, G

Explicación:

1. letters.substring(1, 2)

- Devuelve una subcadena que comienza en el índice 1 (incluido) y termina en el índice 2 (excluido).
- Esto es válido y retorna una cadena de un solo carácter: "b".
- Respuesta correcta: A.

2. letters.substring(2, 2)

- Cuando los índices inicial (beginIndex) y final (endIndex) son iguales, la subcadena es vacía.
- No genera una excepción.
- El índice inicial (beginIndex) no puede ser mayor que el índice final (endIndex)
- Esto lanza una StringIndexOutOfBoundsException.

4. letters.substring(6, 6)

- Aunque el índice inicial y final son iguales, no hay problema. Devuelve una cadena vacía ("").

18. What is the result of the following code? (Choose all that apply.)

```
13: String s1 = """
14:     purr""";
```



```
15: String s2 = "";
16:
17: s1.toUpperCase();
18: s1.trim();
19: s1.substring(1, 3);
20: s1 += "two";
21:
22: s2 += 2;
23: s2 += 'c';
24: s2 += false;
25:
26: if ( s2 == "2cfalse") System.out.println("==");
27: if ( s2.equals("2cfalse")) System.out.println("equals");
28: System.out.println(s1.length());
```

A. `2`
B. `4`
C. `7`
D. `10`
E. `==`
F. `equals`
G. An exception is thrown.
H. The code does not compile.

C,F

Después de estas operaciones, s1 pasa de " purr" a " purrtwo".

La longitud de s1 es 8 (" purrtwo").

2. s2 Modificaciones:

String s2 = "";

s2 += 2; // Concatena "2" → s2 = "2".

s2 += 'c'; // Concatena 'c' → s2 = "2c".

s2 += false; // Concatena "false" → s2 = "2cfalse".

Ahora, s2 es "2cfalse".

3. Comparaciones:

if (s2 == "2cfalse") System.out.println("==");

- Esto no imprime nada, porque el operador == compara referencias, y s2 es un nuevo objeto creado en tiempo de ejecución (no apunta al mismo objeto en el string pool).

if (s2.equals("2cfalse")) System.out.println("equals");

- Esto imprime "equals", porque el método .equals() compara valores de cadenas.

4. Impresión de la longitud de s1:

System.out.println(s1.length());

- La longitud de s1 es 8.

19. Which of the following fill in the blank to print a positive integer? (Choose all that apply.)

```
String[] s1 = { "Camel", "Peacock", "Llama"};
String[] s2 = { "Camel", "Llama", "Peacock"};
String[] s3 = { "Camel"};
String[] s4 = { "Camel", null};
System.out.println(Arrays._____);
```

A. `compare(s1, s2)`
B. `mismatch(s1, s2)`
C. `compare(s3, s4)`
D. `mismatch (s3, s4)`
E. `compare(s4, s4)`
F. `mismatch (s4, s4)`

A, B,D

1. compare(s1, s2)

- Compara lexicográficamente s1 y s2.
- En el índice 1: "Peacock" (en s1) es mayor que "Llama" (en s2).
- Resultado: positivo.

2. mismatch(s1, s2)

- Busca el primer índice donde s1 y s2 difieren.

- En el índice 1: "Peacock" ≠ "Llama".
- Resultado: 1 (positivo).

3. compare(s3, s4)
   - Compara lexicográficamente s3 y s4.
   - En el índice 1, s3 no tiene más elementos, pero s4 tiene null.
   - Resultado: no positivo.
4. mismatch(s3, s4)
   - Busca el primer índice donde s3 y s4 difieren.
   - En el índice 1, s3 termina y s4 tiene null.
   - Resultado: 1 (positivo).
5. compare(s4, s4)
   - Compara s4 consigo mismo.
   - Como son iguales, el resultado es 0.
   - Resultado: no positivo.
6. mismatch(s4, s4)
   - Busca diferencias entre s4 y s4.
   - Como son iguales, no hay diferencias.
   - Resultado: -1 (no positivo).

20. Note that March 13, 2022 is the weekend that clocks spring ahead for daylight saving time. What is the output of the following? (Choose all that apply.)

```
var date = LocalDate.of(2022, Month.MARCH, 13);
var time = LocalTime.of(1, 30);
var zone = ZoneId.of("US/Eastern");
var dateTime1 = ZonedDateTime.of(date, time, zone);
var dateTime2 = dateTime1.plus(1, ChronoUnit.HOURS);

long diff = ChronoUnit.HOURS.between(dateTime1, dateTime2);
int hour = dateTime2.getHour();
boolean offset = dateTime1.getOffset()
   == dateTime2.getOffset();
System.out.println("diff = " + diff);
System.out.println("hour = " + hour);
System.out.println("offset = " + offset);
```

A. `diff = 1`
B. `diff = 2`
C. `hour = 2`
D. `hour = 3`
E. `offset = true`
F. The code does not compile.
G. A runtime exception is thrown.

B,D

- La fecha es el 13 de marzo de 2022.
- La hora inicial es 1:30 AM (antes de que cambien los relojes por el horario de verano en la zona horaria US/Eastern).
- Después de agregar 1 hora, la hora pasa de 1:30 AM a 3:30 AM debido al cambio al horario de verano (de -05:00 a -04:00).
- Línea 6 (ChronoUnit.HOURS.between):

long diff = ChronoUnit.HOURS.between(dateTime1, dateTime2);

- Se calcula la diferencia en horas entre dateTime1 (1:30 AM) y dateTime2 (3:30 AM). Aunque parece que solo se ha sumado 1 hora de reloj, el cambio al horario de verano significa que 2 horas cronológicas han transcurrido (pasando de 1:30 AM a 3:30 AM).
- Resultado: diff = 2
- Línea 7 (getHour):

int hour = dateTime2.getHour();

- La hora de dateTime2 es 3 (porque pasa de 1:30 AM a 3:30 AM).

- Resultado: hour = 3
- Línea 8 (getOffset):

boolean offset = dateTime1.getOffset() == dateTime2.getOffset();

- dateTime1 tiene el offset -05:00 (horario estándar de la zona horaria US/Eastern).
- dateTime2 tiene el offset -04:00 (horario de verano).
- Resultado: offset = false

Salida esperada:

diff = 2

hour = 3

offset = false

21. Which of the following can fill in the blank to print avaJ? (Choose all that apply.)

```
3: var puzzle = new StringBuilder("Java");
4: puzzle._____;
5: System.out.println(puzzle);
```

A. `reverse()`
B. `append("vaJ$").substring(0, 4)`
C. `append("vaJ$").delete(0, 3).deleteCharAt(puzzle.length() - 1)`
D. `append("vaJ$").delete(0, 3).deleteCharAt(puzzle.length())`
E. None of the above

22. What is the output of the following code?

```
var date = LocalDate.of(2022, Month.APRIL, 30);
date.plusDays(2);
date.plusYears(3);
System.out.println(date.getYear() + " " + date.getMonth()
   + " " + date.getDayOfMonth());
```


A. `2022 APRIL 30`
B. `2022 MAY 2`
C. `2025 APRIL 2`
D. `2025 APRIL 30`
E. `2025 MAY 2`
F. The code does not compile.
G. A runtime exception is thrown.

A

Date es inmutable por lo cual se queda con el mismo valor