ACADEMIA JAVA

CAPITULO 1

Esthefanny De La Cruz

Esthefanny De La Cruz
15-1-2025

1. Which of the following are legal entry point methods that can be run from the command line? (Choose all that apply.)

   A. `private static void main(String[] args)`
   B. `public static final main(String[] args)`
   C. `public void main(String[] args)`
   D. `public static final void main(String[] args)`
   E. `public static void main(String[] args)`
   F. `public static main(String[] args)`

E,D

D. public static final void main(String[] args)

Válido. El método incluye todos los modificadores necesarios: public, static, y void. Además, incluir final es permitido pero no obligatorio.

E. public static void main(String[] args)

Válido. Este es el método estándar y más comúnmente utilizado como punto de entrada en un programa Java.

2. Which answer options represent the order in which the following statements can be assembled into a program that will compile successfully? (Choose all that apply.)

```
X: class Rabbit {}
Y: import java.util.*;
Z: package animals;
```

   A. X, Y, Z
   B. Y, Z, X
   C. Z, Y, X
   D. Y, X
   E. Z, X
   F. X, Z
   G. None of the above

C,E

C. Válido. Este es el orden correcto:

Primero el paquete (package animals;).

Luego las importaciones (import java.util.*;).

Finalmente, la definición de la clase (class Rabbit {}).

E. Válido. Este orden es correcto si no hay importaciones necesarias:

Primero el paquete (package animals;).

Luego la definición de la clase (class Rabbit {}).

3. Which of the following are true? (Choose all that apply.)

```
public class Bunny {
   public static void main(String[] x) {
      Bunny bun = new Bunny();
} }
```

A. Bunny is a class.
B. bun is a class.
C. main is a class.
D. Bunny is a reference to an object.
E. bun is a reference to an object.
F. main is a reference to an object.
G. The main() method doesn't run because the parameter name is incorrect.

A,E

public class Bunny: Aquí estamos declarando una clase llamada Bunny.

public static void main(String[] x): Este es el punto de entrada al programa, y su parámetro es String[] x, que puede cambiar de nombre sin afectar el comportamiento (en este caso, x es solo un nombre de parámetro; podría haberse llamado args o cualquier otro nombre).

Bunny bun = new Bunny();: Aquí estamos creando una instancia de la clase Bunny y asignándola a una referencia llamada bun.

4. Which of the following are valid Java identifiers? (Choose all that apply.)

A. _
B. _helloWorld$
C. true
D. java.lang
E. Public
F. 1980_s
G. _Q2_

Respuesta: B, G

Un identificador puede comenzar con una letra (A-Z, a-z), un guion bajo (_) o un signo de dólar ($).

Un identificador puede contener letras, números, guiones bajos y signos de dólar después del primer carácter.

No puede ser una palabra reservada de Java, como true, if, public, etc. En este caso public es sensible a mayúsculas y no puede ser usada como identificador.

Los identificadores no pueden comenzar con un número.

5. Which statements about the following program are correct? (Choose all that apply.)

```
2:  public class Bear {
3:     private Bear pandaBear;
4:     private void roar(Bear b) {
5:        System.out.println("Roar!");
6:        pandaBear = b;
7:     }
8:     public static void main(String[] args) {
9:        Bear brownBear = new Bear();
10:       Bear polarBear = new Bear();
11:       brownBear.roar(polarBear);
12:       polarBear = null;
13:       brownBear = null;
14:       System.gc(); } }
```

A. The object created on line 9 is eligible for garbage collection after line 13.
B. The object created on line 9 is eligible for garbage collection after line 14.
C. The object created on line 10 is eligible for garbage collection after line 12.
D. The object created on line 10 is eligible for garbage collection after line 13.
E. Garbage collection is guaranteed to run.
F. Garbage collection might or might not run.
G. The code does not compile.

F,A,C

A. The object created on line 9 is eligible for garbage collection after line 13.

Correcto. En la línea 9, se crea el objeto brownBear. Luego, en la línea 13, se asigna brownBear = null;, lo que hace que el objeto previamente referenciado por brownBear ya no tenga ninguna referencia. Aún no es recogido por el recolector de basura, pero es elegible para la recolección de basura.

C. The object created on line 10 is eligible for garbage collection after line 12.

Correcto. En la línea 10 se crea el objeto polarBear, y en la línea 12 se asigna polarBear = null;. Esto elimina la referencia al objeto creado en la línea 10, por lo que es elegible para la recolección de basura

F. Garbage collection might or might not run.

Correcto. La recolección de basura puede o no ejecutarse. La llamada a System.gc() es solo una sugerencia al recolector de basura. No hay garantía de que se ejecute inmediatamente o que elimine objetos en ese momento.

6. Assuming the following class compiles, how many variables defined in the class or method are in scope on the line marked on line 14?

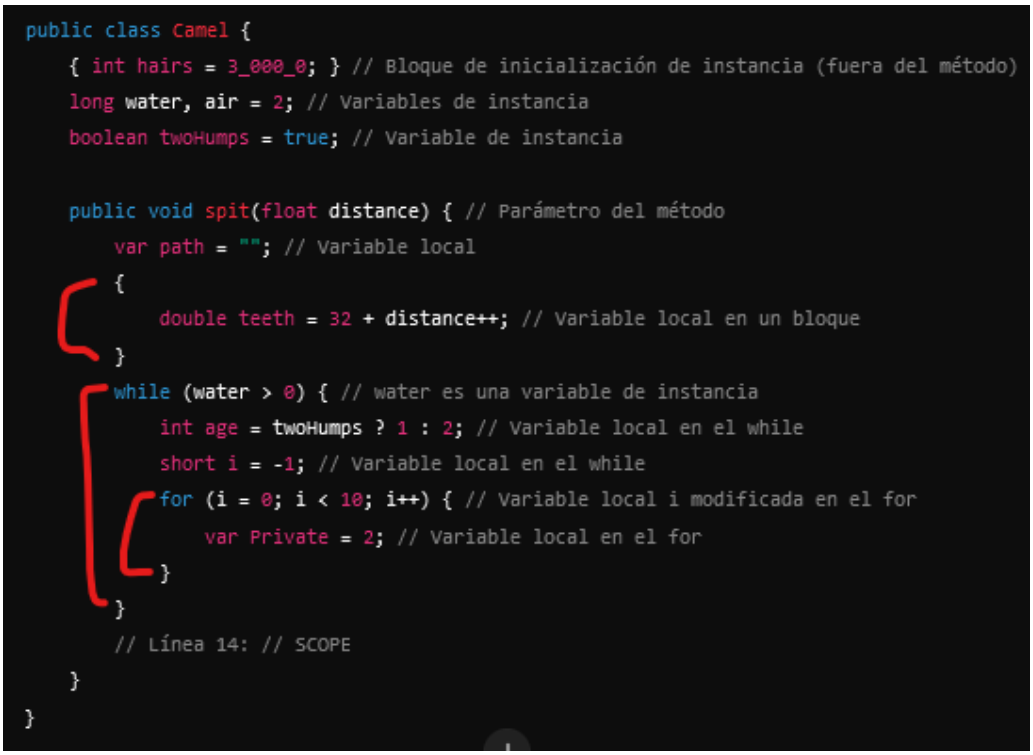
```
1:  public class Camel {
2:     { int hairs = 3_000_0; }
3:     long water, air=2;
4:     boolean twoHumps = true;
5:     public void spit(float distance) {
6:        var path = "";
```



```
7:        { double teeth = 32 + distance++; }
8:        while(water > 0) {
9:           int age = twoHumps ? 1 : 2;
10:          short i=-1;
11:          for(i=0; i<10; i++) {
12:             var Private = 2;
13:          }
14:          // SCOPE
15:       }
16:    }
17: }
```

A. 2
B. 3
C. 4
D. 5
E. 6
F. 7
G. None of the above

Respuesta: D



```java
public class Camel {
    { int hairs = 3_000_0; } // Bloque de inicialización de instancia (fuera del método)
    long water, air = 2; // Variables de instancia
    boolean twoHumps = true; // Variable de instancia

    public void spit(float distance) { // Parámetro del método
        var path = ""; // Variable local
        {
            double teeth = 32 + distance++; // Variable local en un bloque
        }
        while (water > 0) { // water es una variable de instancia
            int age = twoHumps ? 1 : 2; // Variable local en el while
            short i = -1; // Variable local en el while
            for (i = 0; i < 10; i++) { // Variable local i modificada en el for
                var Private = 2; // Variable local en el for
            }
        }
        // Línea 14: // SCOPE
    }
}
```

El alcance se refiere a las variables que están visibles y accesibles en ese punto específico del programa.

Variables en consideración y su alcance:

hairs (línea 2): Declarada en un bloque de inicialización de instancia. No está en alcance en la línea 14 porque su bloque ya terminó.

water (línea 3): Variable de instancia, está en alcance porque las variables de instancia son accesibles en cualquier parte de la clase no estática.

air (línea 3): Variable de instancia, también está en alcance por la misma razón que water.

twoHumps (línea 4): Variable de instancia, está en alcance.

distance (línea 5): Parámetro del método spit(), está en alcance.

path (línea 6): Variable local declarada en el método spit(), está en alcance.

teeth (línea 7): Variable local declarada dentro de un bloque. No está en alcance en la línea 14 porque su bloque ya terminó.

age (línea 9): Variable local declarada dentro del while. No está en alcance en la línea 14 porque su alcance es solo dentro del while.

i (línea 10): Variable local declarada dentro del while. No está en alcance en la línea 14, por la misma razón que age.

Private (línea 12): Variable local declarada dentro del for. No está en alcance en la línea 14 porque el for ya terminó.

7. Which are true about this code? (Choose all that apply.)

```
public class KitchenSink {
    private int numForks;

    public static void main(String[] args) {
       int numKnives;
       System.out.print("""
          "# forks = " + numForks +
           " # knives = " + numKnives +
          # cups = 0""");
    }
}
```

A. The output includes: `# forks = 0`.
B. The output includes: `# knives = 0`.
C. The output includes: `# cups = 0`.
D. The output includes a blank line.
E. The output includes one or more lines that begin with whitespace.
F. The code does not compile.

Respuesta F.

Errores en el código:

- Uso de numForks en un contexto estático:

numForks es una variable de instancia, pero el método main es estático. No puedes acceder directamente a variables de instancia desde un contexto estático sin crear un objeto de la clase.

Esto provoca un error de compilación.

- Variable local numKnives no inicializada:

Las variables locales deben ser inicializadas explícitamente antes de ser utilizadas. En este caso, numKnives no tiene un valor asignado antes de intentar ser utilizado en el System.out.print.

Esto también provoca un error de compilación.

- Errores en la sintaxis del texto multilínea (Text Block):

En la línea del System.out.print, el """ que inicia el bloque de texto está mal usado porque dentro del bloque de texto se incluyen expresiones concatenadas (+ numForks +). Esto no está permitido en los bloques de texto de Java (text blocks), ya que estos solo aceptan texto literal.

Conclusión:

El código no compila debido a los errores mencionados.

8. Which of the following code snippets about `var` compile without issue when used in a method? (Choose all that apply.)

A. `var spring = null;`
B. `var fall = "leaves";`
C. `var evening = 2; evening = null;`
D. `var night = Integer.valueOf(3);`
E. `var day = 1/0;`
F. `var winter = 12, cold;`
G. `var fall = 2, autumn = 2;`
H. `var morning = ""; morning = null;`

Respuesta: BDEH

A. var spring = null;

Error de compilación: El compilador no puede inferir el tipo de null, ya que no tiene suficiente información.

Regla violada: var requiere un tipo claro durante la inicialización.

B. var fall = "leaves";

Correcto: La cadena "leaves" permite al compilador inferir que el tipo de fall es String.

C. var evening = 2; evening = null;

Error de compilación: Inicialmente, evening se infiere como int debido al valor 2. Sin embargo, más adelante intentas asignar null, lo cual no es válido para tipos primitivos.

D. var night = Integer.valueOf(3);

Correcto: El método Integer.valueOf(3) devuelve un objeto de tipo Integer, por lo que el compilador infiere correctamente el tipo.

E. var day = 1/0;

Correcto: Aunque 1/0 genera una excepción en tiempo de ejecución (ArithmeticException), el compilador lo considera una operación válida y asigna el tipo int a day.

Resultado: Compila sin problemas. (Falla en tiempo de ejecución).

F. var winter = 12, cold;

Error de compilación: No se permite declarar varias variables en una sola línea usando var.

G. var fall = 2, autumn = 2;

Regla violada: Declaración múltiple no está permitida con var.

H. var morning = ""; morning = null;

Correcto: Inicialmente, morning se infiere como String. Más adelante, es válido asignarle null, ya que String es un tipo de referencia.

9. Which of the following are correct? (Choose all that apply.)

- **A.** An instance variable of type `float` defaults to `0`.
- **B.** An instance variable of type `char` defaults to `null`.
- **C.** A local variable of type `double` defaults to `0.0`.
- **D.** A local variable of type `int` defaults to `null`.
- **E.** A class variable of type `String` defaults to `null`.
- **F.** A class variable of type `String` defaults to the empty string `""`.
- **G.** None of the above.

A. An instance variable of type float defaults to 0.

✓ incorrecto. Las variables de instancia de tipo float se inicializan automáticamente con el valor 0.0f. no solo 0

B. An instance variable of type char defaults to null.

✗ Incorrecto. El valor por defecto de un char es '\u0000' (no null).

C. A local variable of type double defaults to 0.0.

✗ Incorrecto. Las variables locales no se inicializan automáticamente y deben ser inicializadas explícitamente antes de usarse.

D. A local variable of type int defaults to null.

✗ Incorrecto. Las variables locales de tipo int no se inicializan automáticamente y no pueden tener null como valor.

E. A class variable of type String defaults to null.

✓ Correcto. Las variables de clase (estáticas) de tipo String se inicializan automáticamente con el valor null.

F. A class variable of type String defaults to the empty string "".

✗ Incorrecto. El valor por defecto para referencias (como String) es null, no una cadena vacía.

G. None of the above.

✗ Incorrecto. Hay opciones correctas (E).

10. Which of the following expressions, when inserted independently into the blank line, allow the code to compile? (Choose all that apply.)

```
public void printMagicData() {
   var magic = _____________;
   System.out.println(magic);
}
```

A. `3_1`
B. `1_329_.0`
C. `3_13.0_`
D. `5_291._2`
E. `2_234.0_0`
F. `9___6`
G. `_1_3_5_0`

A,F

A. 3_1 Es un literal entero válido. El guión bajo está en una posición permitida y no está al principio ni al final.

F. 9___6 Es un literal entero válido con múltiples guiones bajos consecutivos en posiciones permitidas.

11. Given the following two class files, what is the maximum number of imports that can be removed and have the code still compile?

```
// Water.java
package aquarium;
public class Water { }
```



```
// Tank.java
package aquarium;
import java.lang.*;
import java.lang.System;
import aquarium.Water;
import aquarium.*;
public class Tank {
   public void print(Water water) {
   System.out.println(water); } }
```

A. 0
B. 1
C. 2
D. 3
E. 4
F. Does not compile

Respuesta E

import java.lang.*;

→ Se puede eliminar, ya que es redundante (está implícito).

import java.lang.System;

→ Se puede eliminar, ya que System está incluido en java.lang.*, que es implícito.

import aquarium.Water;

→ Se puede eliminar, porque el package aquarium.* ya lo incluye.

import aquarium.*;

→ se puede eliminar, ya que la clase Water está en el paquete aquarium

12. Which statements about the following class are correct? (Choose all that apply.)

```
1: public class ClownFish {
2:    int gills = 0, double weight=2;
3:    { int fins = gills; }
4:    void print(int length = 3) {
5:       System.out.println(gills);
6:       System.out.println(weight);
7:       System.out.println(fins);
8:       System.out.println(length);
9: } }
```

A. Line 2 generates a compiler error.
B. Line 3 generates a compiler error.
C. Line 4 generates a compiler error.
D. Line 7 generates a compiler error.
E. The code prints `0`.
F. The code prints `2.0`.
G. The code prints `2`.
H. The code prints `3`.

Respuesta: A, D, C

El código no imprime algo debido a los diversos errores de compilación

Línea 2:int gills = 0, double weight = 2;

Problema: No se pueden declarar variables de diferentes tipos en la misma línea utilizando esta sintaxis. Aquí se intenta declarar gills como int y weight como double en una sola línea, lo cual genera un error de compilación.

Línea 3:

{ int fins = gills; }

Problema: Aunque esta línea intenta inicializar fins con el valor de gills, el bloque de inicialización no tiene errores en este contexto, ya que gills se ha declarado previamente como un atributo de la clase.

Línea 4:

void print(int length = 3) {

Problema: Java no permite valores predeterminados para los parámetros de métodos. La asignación int length = 3 no es válida en este contexto. Si se desea asignar un valor predeterminado, debe hacerse dentro del cuerpo del método.

Línea 7:

System.out.println(fins);

Problema: La variable fins está declarada dentro del bloque de inicialización (línea 3), lo que la hace una variable de alcance local al bloque. No es accesible fuera de dicho bloque, por lo que intentar imprimirla aquí genera un error de compilación

13. Given the following classes, which of the following snippets can independently be inserted in place of `INSERT IMPORTS HERE` and have the code compile? (Choose all that apply.)

```
package aquarium;
public class Water {
   boolean salty = false;
}

package aquarium.jellies;
public class Water {
   boolean salty = true;
}

package employee;
INSERT IMPORTS HERE
public class WaterFiller {
   Water water;
}
```

A. `import aquarium.*;`

B.
```
import aquarium.Water;
import aquarium.jellies.*;
```

C.
```
import aquarium.*;
import aquarium.jellies.Water;
```

D.
```
import aquarium.*;
import aquarium.jellies.*;
```

E.
```
import aquarium.Water;
import aquarium.jellies.Water;
```

F. None of these imports can make the code compile.

Respuesta: A, B,C MARCADA PARA REVISION

Si la pregunta solo se refiere a qué importaciones permiten que el código compile, sin profundizar en posibles confusiones o ambigüedades en tiempo de ejecución, la respuesta debe enfocarse en que el código realmente compila.

Revisión de las opciones:

Opción A: import aquarium.*; Esta opción sí compila. Aunque hay ambigüedad entre aquarium.Water y aquarium.jellies.Water, Java elige la clase Water de aquarium debido a que no se importan explícitamente ambas clases de forma conflictiva en el código (la ambigüedad no se resuelve hasta el uso de la clase). El código compila porque no se hace referencia a ambas clases al mismo tiempo.

Opción B: import aquarium.Water; import aquarium.jellies.*; Esta opción también compila. Aquí se hace una importación explícita de Water desde aquarium y luego se importan todas las clases de aquarium.jellies. No hay conflicto porque la clase Water ya está definida de forma explícita en el código.

Opción C: import aquarium.*; import aquarium.jellies.Water; Esta opción también compila. Similar a la opción B, pero en este caso se importa explícitamente Water desde aquarium.jellies. Aunque hay una importación de todas las clases de aquarium y podría generar confusión, no afecta el compilado porque la clase Water de aquarium.jellies se está importando explícitamente.

Opción D: import aquarium.*; import aquarium.jellies.*; Esta opción no compila. El uso de los comodines de ambos paquetes genera un conflicto de ambigüedad porque Java no sabría cuál de las dos clases Water utilizar, y esto hace que el código no compile.

Opción E: import aquarium.Water; import aquarium.jellies.Water; Esta opción no compila. Java no permite la importación de dos clases con el mismo nombre, incluso si están en paquetes diferentes, por lo que esta combinación generaría un error de compilación

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

```
3: short numPets = 5L;
4: int numGrains = 2.0;
5: String name = "Scruffy";
6: int d = numPets.length();
7: int e = numGrains.length;
8: int f = name.length();
```


A. Line 3 generates a compiler error.
B. Line 4 generates a compiler error.
C. Line 5 generates a compiler error.
D. Line 6 generates a compiler error.
E. Line 7 generates a compiler error.
F. Line 8 generates a compiler error.

Respuestas: A,B,D,E

Línea 3: short numPets = 5L;

En Java, 5L es un long, y no se puede asignar un long a un short sin realizar una conversión explícita. Esto generará un error de compilación.

Línea 4: int numGrains = 2.0;

2.0 es un valor double, y no se puede asignar un double a un int sin hacer una conversión explícita. Esto también genera un error de compilación.

Línea 6: int d = numPets.length();

numPets es un short, y el tipo short no tiene un método length().Línea 7: int e = numGrains.length;

línea 7 : numGrains es un int, y el tipo int no tiene una propiedad length.

15. Which of the following statements about garbage collection are correct? (Choose all that apply.)

- **A.** Calling `System.gc()` is guaranteed to free up memory by destroying objects eligible for garbage collection.
- **B.** Garbage collection runs on a set schedule.
- **C.** Garbage collection allows the JVM to reclaim memory for other objects.
- **D.** Garbage collection runs when your program has used up half the available memory.
- **E.** An object may be eligible for garbage collection but never removed from the heap.
- **F.** An object is eligible for garbage collection once no references to it are accessible in the program.
- **G.** Marking a variable `final` means its associated object will never be garbage collected.

Respuestas: C,F,E

G. Marcar una variable como final significa que su objeto asociado nunca será recolectado por la basura.

Incorrecto: La palabra clave final garantiza que la referencia de la variable no se puede cambiar, pero no impide que el objeto al que apunta la variable sea recolectado por la basura si ya no tiene otras referencias. Un objeto puede ser recogido por la basura si no hay otras referencias activas a él, incluso si está marcado como final

16. Which are true about this code? (Choose all that apply.)

```
var blocky = """
   squirrel \s
   pigeon   \
   termite""";
System.out.print(blocky);
```

A. It outputs two lines.
B. It outputs three lines.
C. It outputs four lines.
D. There is one line with trailing whitespace.
E. There are two lines with trailing whitespace.
F. If we indented each line five characters, it would change the output.

*Este ejercicio es a partir de java 13

B,D

B. It outputs three lines.

Correcto: La cadena tiene 3 líneas separadas Y “”” permite tener un string en varias líneas. Aunque el uso de \ hace que pigeon \ continúe en la misma línea, los saltos de línea aún están presentes al final de cada línea.

D. There is one line with trailing whitespace.

Correcto: La primera línea (squirrel \s) tiene un espacio al final debido al \s.

F. If we indented each line five characters, it would change the output.

Incorrecto: El texto en la cadena mantendría los saltos de línea, pero la indentación de cada línea no afecta la salida cuando usas comillas triples, ya que los saltos de línea y los espacios dentro de las comillas se mantienen tal cual están.

17. What lines are printed by the following program? (Choose all that apply.)

```
1:  public class WaterBottle {
2:     private String brand;
3:     private boolean empty;
4:     public static float code;
5:     public static void main(String[] args) {
6:        WaterBottle wb = new WaterBottle();
```



```
7:        System.out.println("Empty = " + wb.empty);
8:        System.out.println("Brand = " + wb.brand);
9:        System.out.println("Code = " + code);
10:    } }
```

A. Line 8 generates a compiler error.
B. Line 9 generates a compiler error.
C. `Empty =`
D. `Empty = false`
E. `Brand =`
F. `Brand = null`
G. `Code = 0.0`
H. `Code = 0f`

Línea 7: System.out.println("Empty = " + wb.empty);

Para un boolean, el valor predeterminado es false.

Línea 8: System.out.println("Brand = " + wb.brand);

Las variables de tipo objeto en Java se inicializan con null si no se les asigna un valor explícito.

Línea 9: System.out.println("Code = " + code);

En este caso, el valor predeterminado para un float es 0.0

18. Which of the following statements about `var` are true? (Choose all that apply.)

A. A `var` can be used as a constructor parameter.
B. The type of a `var` is known at compile time.
C. A `var` cannot be used as an instance variable.
D. A `var` can be used in a multiple variable assignment statement.
E. The value of a `var` cannot change at runtime.
F. The type of a `var` cannot change at runtime.
G. The word `var` is a reserved word in Java.

B,C,F

B. The type of a var is known at compile time.

VERDADERO.

El tipo de una variable declarada con var es inferido por el compilador en el momento de la compilación basándose en el valor asignado a la variable. Esto significa que el tipo de var no es dinámico, sino estáticamente determinado.

C. A var cannot be used as an instance variable.

VERDADERO.

var solo puede usarse para declarar variables locales.

F. The type of a var cannot change at runtime.

VERDADERO.

El tipo de una variable declarada con var está definido en tiempo de compilación y no puede cambiar en tiempo de ejecución. Por ejemplo, si el compilador infiere que var es un int, no puedes asignarle un String más adelante.

19. Which are true about the following code? (Choose all that apply.)

```
var num1 = Long.parseLong("100");
var num2 = Long.valueOf("100");
System.out.println(Long.max(num1, num2));
```

A. The output is `100`.
B. The output is `200`.
C. The code does not compile.
D. `num1` is a primitive.
E. `num2` is a primitive.

A,D

1.
var num1 = Long.parseLong("100");
El método Long.parseLong(String s) devuelve un valor primitivo long.
Por lo tanto, num1 es de tipo primitivo long.

var num2 = Long.valueOf("100");
El método Long.valueOf(String s) devuelve un objeto wrapper de tipo Long (no un primitivo).
Por lo tanto, num2 es un objeto de la clase Long.

Uso del método Long.max:

Long.max(long a, long b) es un método estático que recibe valores primitivos de tipo long como parámetros.

En este caso,:

num1 ya es un primitivo long.

num2 se desempaqueta automáticamente (auto-unboxing) al tipo primitivo long para que el método funcione.

El método calcula el valor máximo entre los dos números, que es 100.

imprime 100, ya que ambos valores son iguales.

20. Which statements about the following class are correct? (Choose all that apply.)

```
1:  public class PoliceBox {
2:     String color;
3:     long age;
4:     public void PoliceBox() {
5:        color = "blue";
6:        age = 1200;
```



```
7:     }
8:     public static void main(String []time) {
9:        var p = new PoliceBox();
10:       var q = new PoliceBox();
11:       p.color = "green";
12:       p.age = 1400;
13:       p = q;
14:       System.out.println("Q1="+q.color);
15:       System.out.println("Q2="+q.age);
16:       System.out.println("P1="+p.color);
17:       System.out.println("P2="+p.age);
18: } }
```

A. It prints `Q1=blue`.
B. It prints `Q2=1200`.
C. It prints `P1=null`.
D. It prints `P2=1400`.
E. Line 4 does not compile.
F. Line 12 does not compile.
G. Line 13 does not compile.
H. None of the above.

La respuesta es E , ya que en condiciones normales la línea 4 no compila, pero en caso de que lo haga imprimiría C

La línea 4 define un método llamado `PoliceBox` con tipo de retorno `void`. **Esto no es un constructor válido** porque un constructor no puede tener un tipo de retorno.

public void PoliceBox() no es un constructor, sino un método con nombre igual al de la clase. Esto hace que el compilador genere un constructor predeterminado que no inicializa las variables color y age. Por lo tanto, al ejecutar el programa:

q.color será null (valor por defecto de String).

q.age será 0 (valor por defecto de long).

En las líneas 9 y 10, se crean dos objetos PoliceBox utilizando el constructor predeterminado, el cual no inicializa las variables color y age.

Por lo tanto, después de la creación:

p.color == null, p.age == 0

q.color == null, q.age == 0

Modificaciones en p:

Línea 11: p.color se asigna a "green".

Línea 12: p.age se asigna a 1400.

Asignación p = q:

Línea 13: Ahora, tanto p como q apuntan al mismo objeto (el creado en la línea 10). Esto provoca que el objeto original referenciado por p sea elegible para recolección de basura (GC).

Entonces queda:

p.color == null, p.age == 0

q.color == null, q.age == 0

21. What is the output of executing the following class?

```
1:  public class Salmon {
2:     int count;
3:     { System.out.print(count+"-"); }
4:     { count++; }
5:     public Salmon() {
6:        count = 4;
7:        System.out.print(2+"-");
8:     }
9:     public static void main(String[] args) {
10:       System.out.print(7+"-");
11:       var s = new Salmon();
12:       System.out.print(s.count+"-"); } }
```


A. `7-0-2-1-`
B. `7-0-1-`
C. `0-7-2-1-`
D. `7-0-2-4-`
E. `0-7-1-`
F. The class does not compile because of line 3.
G. The class does not compile because of line 4.
H. None of the above.

Respuesta D

Ejecución paso a paso

Inicio del programa (línea 10):

Se ejecuta:

System.out.print(7+"-");

Salida acumulada: 7-.

Creación de la instancia Salmon (línea 11):

Al crear el objeto, se ejecutan los bloques de inicialización en el orden en que están declarados:

Bloque 1 (línea 3):

Imprime el valor actual de count (0) seguido de un guion.

Salida acumulada: 7-0-.

Bloque 2 (línea 4):

Incrementa el valor de count en 1. Ahora, count = 1.

Luego se ejecuta el constructor (líneas 5-8):

Asigna el valor 4 a count.

Imprime 2-.

Salida acumulada: 7-0-2-.

Impresión final del valor de count (línea 12):

El valor actual de count es 4 (establecido por el constructor).

Salida acumulada: 7-0-2-4-

22. Given the following class, which of the following lines of code can independently replace `INSERT CODE HERE` to make the code compile? (Choose all that apply.)

```
public class Price {
   public void admission() {
      INSERT CODE HERE
      System.out.print(amount);
      } }
```

A. `int Amount = 0b11;`
B. `int amount = 9L;`
C. `int amount = 0xE;`
D. `int amount = 1_2.0;`
E. `double amount = 1_0_.0;`
F. `int amount = 0b101;`
G. `double amount = 9_2.1_2;`
H. `double amount = 1_2_.0_0;`

A: Válido (literal binario).

C: Válido (literal hexadecimal).

F: Válido (literal binario)

Java, una variable de tipo int puede almacenar números representados en diferentes formatos numéricos, como binarios, hexadecimales, octales y decimales

23. Which statements about the following class are true? (Choose all that apply.)

```
1:  public class River {
2:     int Depth = 1;
3:     float temp = 50.0;
4:     public void flow() {
5:        for (int i = 0; i < 1; i++) {
6:           int depth = 2;
7:           depth++;
8:           temp--;
9:        }
```



```
10:       System.out.println(depth);
11:       System.out.println(temp); }
12:    public static void main(String... s) {
13:       new River().flow();
14: } }
```

A. Line 3 generates a compiler error.
B. Line 6 generates a compiler error.
C. Line 7 generates a compiler error.
D. Line 10 generates a compiler error.
E. The program prints 3 on line 10.
F. The program prints 4 on line 10.
G. The program prints `50.0` on line 11.
H. The program prints `49.0` on line 11.

A,D

Un literal float debe terminar con una f o F. Sin ella, se asigna un literal double, causando un error de compilación.

Correcto: float temp = 50.0f;

Incorrecto: float temp = 50.0;

No se puede acceder a una variable declarada dentro de un bloque (como un bucle for) fuera de ese bloque.