CUESTIONARIO JAVA. RA 123467

0. Conversión de tipos ¿Cuál de los siguientes ejemplos muestra una conversión explícita de tipo?

- a) `int x = 10.5;`
- b) `double y = (double) 10;`
- c) `String s = "10";`
- d) `int z = Integer.parseInt("10");`

1. Operadores ¿Qué operador se utiliza para comparar dos valores en Java?

- a) `=`
- b) `==`
- c) `!=`
- d) `<=`

2. Instanciación de objetos ¿Cuál de las siguientes afirmaciones sobre la instanciación de objetos es correcta?

- a) Un objeto puede ser instanciado sin definir su clase.
- b) La palabra clave `new` es opcional al crear un objeto.
- c) Los objetos siempre se inicializan en `null`.
- d) Un objeto se crea utilizando el constructor de su clase.

3. Constructores ¿Qué ocurre si no se define un constructor en una clase?

- a) La clase no se podrá instanciar.
- b) El compilador genera un constructor por defecto.
- c) El constructor debe definirse en una subclase.
- d) Se genera un error de compilación.

4. Métodos estáticos ¿Qué característica tienen los métodos estáticos?

- a) Solo pueden ser invocados desde una instancia de la clase.
- b) No tienen acceso a las variables de instancia de la clase.
- c) Son exclusivos para clases abstractas.
- d) No pueden aceptar parámetros.

5. Estructuras de selección ¿Cuál es una estructura de selección válida en Java?

- a) `if-else`
- b) `switch-case`
- c) `for`
- d) Ambas a y b

6. Arrays multidimensionales ¿Cómo se accede al elemento en la segunda fila y tercera columna de un array bidimensional **`matriz`**?

- a) `matriz[2][3]`
- b) `matriz[1][2]`
- c) `matriz[3][2]`
- d) `matriz[2][1]`

7. Herencia ¿Qué palabra clave permite que una clase herede de otra?

- a) `implements`
- b) `extends`
- c) `inherits`
- d) `super`

8. Polimorfismo ¿Cuál es un ejemplo de polimorfismo?

- a) Sobrecarga de métodos
- b) Sobrescritura de métodos
- c) Instanciación de clases
- d) Ambas a y b

9. En Java, una variable declarada como `final` no puede cambiar su valor después de ser inicializada.

- a) Verdadero
- b) Falso

10. El operador `=` en Java se utiliza para comparar valores.

- a) Verdadero
- b) Falso

11. Un método declarado como estático no puede acceder a miembros no estáticos de la clase.

- a) Verdadero
- b) Falso

12. Un objeto puede ser instanciado sin utilizar la palabra clave `new`.

- a) Verdadero
- b) Falso

13. Una clase abstracta no puede ser instanciada directamente.

- a) Verdadero
- b) Falso

14. Los arrays en Java tienen un tamaño fijo una vez inicializados.

- a) Verdadero
- b) Falso

15. Los métodos sobrecargados deben tener el mismo nombre pero diferentes parámetros.

- a) Verdadero
- b) Falso

16. En una jerarquía de clases, un método de la superclase puede ser sobreescrito por una subclase.

- a) Verdadero
- b) Falso

17. Un método puede tener múltiples bloques `catch` para manejar diferentes tipos de excepciones.

- a) Verdadero
- b) Falso

18. En Java, el encapsulamiento permite que los atributos de una clase estén protegidos y sean accesibles únicamente a través de métodos específicos, lo que mejora la seguridad y control sobre el estado de los objetos.

- a) Verdadero
- b) Falso

19. Estos métodos ofrecen control sobre el acceso a los atributos privados, permitiendo la validación de los datos antes de ser asignados.

- a) Getters y Setters
- b) Ninguna es correcta
- a) Métodos protegidos
- b) Métodos privados

20. La palabra clave 'this' en Java se emplea para referirse a la variable de instancia actual de la clase

- a) Verdadero
- b) Falso

21. Literales ¿Qué tipo de literal se utiliza para representar un carácter en Java?

- a) "A"
- b) 'A'
- c) A
- d) Ninguna de las anteriores

22. Control de flujo ¿Qué hace la palabra clave **`continue`** dentro de un bucle?

- a) Termina el bucle inmediatamente.
- b) Salta la iteración actual y pasa a la siguiente.
- c) Detiene el programa con un error.
- d) Reinicia el bucle desde la primera iteración.

23. Clases ¿Qué sucede si una clase en Java no tiene ningún modificador de visibilidad explícito?

- a) Es privada por defecto.
- b) Es pública por defecto.
- c) Tiene visibilidad de paquete por defecto.
- d) No puede ser utilizada en ningún programa.

24. Métodos ¿Qué ocurre si un método en Java está declarado como estático?

- a) Solo puede ser llamado desde instancias de la clase.
- b) Puede ser llamado sin instanciar la clase.
- c) No puede tener parámetros.
- d) No puede devolver un valor.

25. Herencia ¿Cuál de las siguientes NO es una ventaja de la herencia?

- a) Reutilización de código.
- b) Mejora de la encapsulación.
- c) Polimorfismo.
- d) Sobreescritura

26. Polimorfismo ¿Qué técnica permite que un objeto de una clase hija sea tratado como si fuera de la clase padre?

- a) Sobrecarga
- b) Encapsulación
- c) Polimorfismo
- d) Visibilidad

27. Excepciones ¿Qué excepción se lanza cuando se intenta acceder a un índice fuera de los límites de un array?

- a) NullPointerException
- b) ArrayIndexOutOfBoundsException
- c) ArithmeticException
- d) IOException

28. Interfaces ¿Cuál es la diferencia entre una clase abstracta y una interfaz?

- a) Una clase abstracta no puede tener métodos abstractos.
- b) Una interfaz permite múltiples implementaciones.
- c) Una interfaz no puede ser implementada por otra clase.
- d) Una clase abstracta no puede contener métodos concretos.

29. Sobreescritura de métodos ¿Qué palabra clave se utiliza para sobrescribir un método en Java?

- a) override
- b) super
- c) this
- d) extends

30. Salida del programa

```
public class Main {
    public static void main(String[] args) {
        int a = 5;
        int b = 10;
        System.out.println(a + b);
    }
}
```

¿Cuál será la salida del programa?

- a) 15
- b) 510
- c) Error de compilación
- d) Ninguna de las anteriores

31. Corrección de errores

```
public class Main {
    public static void main(String[] args) {
        int num;
        System.out.println(num);
    }
}
```

El código genera un error de compilación. ¿Cómo se puede corregir?

- a) Declarando `num` como una variable estática.
- b) Inicializando `num` antes de usarlo.
- c) Cambiando `int` por `double`.
- d) Eliminando la declaración de `num`.

32. Análisis de bucles

```
public class Main {
    public static void main(String[] args) {
        for (int i = 0; i < 5; i++) {
            System.out.print(i + " ");
        }
    }
}
```

¿Cuál será la salida del programa?

- a) 0 1 2 3 4
- b) 1 2 3 4 5
- c) 0 1 2 3 4 5
- d) Error de compilación

33. Completar el código.

```
public class Main {
    public static void main(String[] args) {
        int[] nums = {1, 2, 3, 4, 5};
        int total = 0;
```

```
        // Añade aquí una estructura de control para sumar todos los elementos del
array
    }
}
```

¿Qué línea de código completa el programa para calcular la suma de los elementos del array?

- a) `for (int num : nums) total += num;`
- b) `for (int i = 1; i <= nums.length; i++) total += nums[i];`
- c) `while (nums.length > 0) total += nums[0];`
- d) Ninguna de las anteriores

34. Salida de métodos

```
public class Main {
    public static int multiply(int a, int b) {
        return a * b;
    }
    public static void main(String[] args) {
        System.out.println(multiply(4, 5));
    }
}
```

¿Cuál será la salida del programa?

- a) 9
- b) 20
- c) Error de compilación
- d) Ninguna de las anteriores

35. Orden de ejecución

```
public class Main {
    public static void main(String[] args) {
        int x = 5;
        int y = ++x;
        System.out.println(x + " " + y);
    }
}
```

¿Cuál será la salida del programa?

- a) 5 6
- b) 6 5
- c) 6 6
- d) Error de compilación

36. Salida condicional

```
public class Main {
    public static void main(String[] args) {
        int x = 10;
        if (x % 2 == 0) {
            System.out.println("Par");
        } else {
            System.out.println("Impar");
        }
    }
}
```

¿Qué imprime este programa?

- a) Par
- b) Impar
- c) Error de compilación
- d) Ninguna de las anteriores

37. Polimorfismo

```
class Animal {
    public void sound() {
        System.out.println("Cualquier sonido");
    }
}
class Dog extends Animal {
    public void sound() {
        System.out.println("Guau guau");
    }
}
public class Main {
    public static void main(String[] args) {
        Animal obj = new Dog();
        obj.sound();
    }
}
```

¿Qué salida genera este programa?

- a) Cualquier sonido
- b) Guau guau
- c) Error de compilación
- d) Ninguna de las anteriores

38. Método incompleto

```
public class Main {
    public static int factorial(int n) {
        if (n == 0) {
            return 1;
        } else {
            // Completa esta línea para calcular el factorial recursivo
        }
    }
    public static void main(String[] args) {
        System.out.println(factorial(5));
    }
}
```

¿Qué línea completa correctamente el método `factorial`?

- a) `return n;`
- b) `return n + factorial(n - 1);`
- c) `return n * factorial(n - 1);`
- d) Ninguna de las anteriores

39. Generación de salida (Bucle anidado)

```
public class Main {
    public static void main(String[] args) {
        for (int i = 1; i <= 3; i++) {
            for (int j = 1; j <= i; j++) {
                System.out.print(j + " ");
            }
            System.out.println();
        }
```

```
    }
}
```

¿Cuál será la salida del programa?

- a)

```
1
1 2
1 2 3
```

- b)

```
1 1
2 2
3 3
```

- c)

```
1 2 3
1 2 3
1 2 3
```

- d) Error de compilación

40. Identificación de errores

```
public class Main {
   public static void main(String[] args) {
       int[] arr = new int[5];
       arr[5] = 10;
       System.out.println("Done");
   }
}
```

¿Qué sucede al ejecutar este programa?

- a) Imprime `Done`.
- b) Genera un error en tiempo de compilación.
- c) Lanza una excepción `ArrayIndexOutOfBoundsException`.
- d) Se detiene sin imprimir nada.

41. Escritura de método recursivo. Escribe un método en Java que calcule la suma de todos los números de 1 a n utilizando recursión. Completa el siguiente código:

```
public static int suma(int n) {
   if (n == 1) {
       return 1;
   } else {
       // Completar aquí
   }
}
```

¿Cuál de las siguientes líneas completa correctamente el código?

- a) `return suma(n - 1);`
- b) `return n + suma(n - 1);`
- c) `return n * suma(n - 1);`
- d) Ninguna de las anteriores

42. Comportamiento de herencia.

```java
class Parent {
    public void vista() {
        System.out.println("Padre");
    }
}

class Child extends Parent {
    public void vista() {
        System.out.println("Hijo");
    }
}

public class Main {
    public static void main(String[] args) {
        Padre obj = new Hijo();
        obj.vista();
    }
}
```

¿Qué salida genera este programa?

- a) Padre
- b) Hijo
- c) Error de compilación
- d) Ninguna de las anteriores

43. Escritura de interfaces. Define una interfaz llamada **`Operable`** con un método **`operar`** que toma dos enteros como parámetros y devuelve un entero. Completa este código:

```java
interface Operable {
    // Escribe el metodo aquí
}
```

¿Qué línea completa correctamente la interfaz?

- a) `void operar();`
- b) `int operar(int a, int b);`
- c) `static int operar(int a, int b);`
- d) Ninguna de las anteriores

44. Orden de ejecución

```java
public class Main {
    public static void main(String[] args) {
        int a = 5, b = 10;
        System.out.println(++a + b--);
        System.out.println(a + b);
    }
}
```

¿Cuál será la salida del programa?

- a) 15, 15
- b) 16, 15
- c) 16, 14
- d) Error de compilación

45. Excepciones personalizadas. Completa el siguiente código para definir una excepción personalizada:

```java
class MyException extends Exception {
    public MyException(String message) {
```

```
        // Escribe aquí el cuerpo del constructor
    }
}
```

¿Qué línea completa correctamente el constructor?

- a) `super(message);`
- b) `this.message = message;`
- c) `System.out.println(message);`
- d) Ninguna de las anteriores

46. Uso de arrays multidimensionales.

```
public class Main {
    public static void main(String[] args) {
        int[][] matrix = {
                {1, 2, 3},
                {4, 5, 6},
                {7, 8, 9}
        };
        System.out.println(matrix[1][2]);
    }
}
```

¿Cuál será la salida del programa?

- a) 4
- b) 5
- c) 6
- d) Error de compilación

47. Búsqueda en una lista ¿Cuál de las siguientes expresiones verifica si un elemento está presente en una lista?

```
List<String> names = Arrays.asList("Alice", "Bob", "Charlie");
```

- a) `names.contains("Alice");`
- b) `names.indexOf("Alice") == -1;`
- c) `names.get("Alice");`
- d) `names.search("Alice");`

48. Operaciones con colecciones

```
public class Main {
    public static void main(String[] args) {
        List<String> list = new ArrayList<>();
        list.add("A");
        list.add("B");
        list.add("C");
        list.remove(1);
        System.out.println(list);
    }
}
```

¿Qué salida genera este programa?

- a) [A, B]
- b) [A, C]
- c) [B, C]
- d) Error de compilación

49. Herencia y jerarquía de clases. Dado el siguiente código:

```
class Animal {
    public void hacerSonido() {
        System.out.println("Cualquier sonido");
    }
}

class Cat extends Animal {
    public void hacerSonido() {
        System.out.println("Meow");
    }
}

public class Main {
    public static void main(String[] args) {
        Animal animal = new Cat();
        animal.hacerSonido();
    }
}
```

¿Qué salida genera este programa?

- a) Cualquier sonido
- b) Meow
- c) Error de compilación
- d) Ninguna de las anteriores

50. Polimorfismo ¿Qué principio de la programación orientada a objetos permite que una subclase pueda sobreescribir un método de su superclase?

- a) Encapsulación
- b) Abstracción
- c) Herencia
- d) Polimorfismo

51. Clases abstractas ¿Cuál de las siguientes afirmaciones sobre las clases abstractas en Java es correcta?

- a) Una clase abstracta no puede tener métodos concretos.
- b) Una clase abstracta debe ser final.
- c) Una clase abstracta puede tener métodos abstractos y concretos.
- d) Una clase abstracta no puede extender otra clase abstracta.

52. Sobrescritura de métodos ¿Qué palabra clave debe usarse para indicar que un método de una subclase sobrescribe un método de la superclase?

- a) `super`
- b) `@Override`
- c) `@Overload`
- d) `this`

53. Uso de super ¿Qué hace la palabra clave **`super`** en Java?

- a) Permite acceder a los miembros de la subclase desde la superclase.
- b) Permite acceder a los miembros de la superclase desde la subclase.
- c) Evita la sobrescritura de métodos y atributos.
- d) Proporciona sobreescritura de métodos y atributos.

54. Implementación de interfaces. Completa el siguiente código para implementar correctamente la interfaz **`Calculable`**:

```
interface Calculable {
```

```
    int calculate(int a, int b);
}

class Calculator implements Calculable {
    // Completar el métodо
}
```

¿Qué línea completa correctamente la implementación de la interfaz?

- a) `public int calculate() { return 0; }`
- b) `public int calculate(int x, int y) { return x + y; }`
- c) `public void calculate(int a, int b) { return a + b; }`
- d) Ninguna de las anteriores

55. Jerarquía de clases. Dado el siguiente código:

```
class Vehicle {
    public void start() {
        System.out.println("Vehicle started");
    }
}

class Car extends Vehicle {
    public void start() {
        System.out.println("Car started");
    }
}

public class Main {
    public static void main(String[] args) {
        Vehicle v = new Car();
        v.start();
    }
}
```

¿Qué salida genera este programa?

- a) Vehicle started
- b) Car started
- c) Error de compilación
- d) Ninguna de las anteriores

56. Polimorfismo y métodos ¿Verdadero o falso? Un método estático puede ser sobreescrito por una subclase.

- a) Verdadero
- b) Falso

57. Herencia múltiple mediante interfaces. Dado el siguiente código:

```
interface A {
    void haceAlgo();
}

interface B {
    void haceAlgo();
}

class C implements A, B {
    public void haceAlgo() {
        System.out.println("Hago algo");
    }
}
```

¿Qué ocurre si la clase `C` implementa ambas interfaces?

- a) Genera un error de compilación porque hay ambigüedad.
- b) Genera un error de compilación porque Java no permite herencia múltiple.
- c) Funciona correctamente, siempre que `haceAlgo()` esté implementado.
- d) No es posible implementar más de una interfaz en Java.

58. Modificadores de acceso. Dado el siguiente código:

```
class Parent {
   private void display() {
       System.out.println("Parent");
   }
}

class Child extends Parent {
   public void display() {
       System.out.println("Child");
   }
}

public class Main {
   public static void main(String[] args) {
       Parent obj = new Child();
       obj.display();
   }
}
```

¿Qué salida genera este programa?

- a) Parent
- b) Child
- c) Error de compilación
- d) Ninguna de las anteriores

59. Variables y constantes ¿Cuál de las siguientes afirmaciones sobre variables y constantes es correcta?

- a) Las constantes pueden ser modificadas durante la ejecución.
- b) Las variables deben declararse y definirse en líneas separadas.
- c) Las constantes se declaran usando **`final`**.
- d) Las variables siempre requieren un valor inicial.

60. Estructura de un programa ¿Cuál de las siguientes opciones NO es un bloque fundamental en la estructura de un programa?

- a) Variables
- b) Constantes
- c) Servidores
- d) Operadores

61. Variables ¿Cuál de las siguientes afirmaciones es correcta sobre las variables?

- a) Las variables no cambian su valor durante la ejecución de un programa.
- b) Las variables almacenan datos que pueden cambiar durante la ejecución de un programa.
- c) Las variables siempre deben inicializarse con cero.
- d) Las variables son opcionales en un programa.

62. Tipos de datos ¿Cuál de los siguientes NO es un tipo de dato primitivo?

- a) Entero
- b) Booleano
- c) Clase
- d) Caracter

63. Operadores ¿Cuál de las siguientes opciones describe un operador aritmético?

- a) Todas son correctas
- b) OR
- c) +
- d) !=

64. Conversión de tipos ¿Qué tipo de conversión se realiza automáticamente por el compilador?

- a) Implícita
- b) Explícita
- c) Forzada
- d) Mixta

65. Programación orientada a objetos (POO) ¿Verdadero o falso? En programación orientada a objetos, un objeto es una instancia de una clase.

- a) Verdadero
- b) Falso

66. Clases ¿Qué define una clase en POO?

- a) La estructura de un tipo de dato primitivo
- b) El diseño y comportamiento de un objeto
- c) Un conjunto de operaciones matemáticas
- d) Un espacio de memoria reservado

67. Métodos ¿Cuál de las siguientes NO es una característica de los métodos?

- a) Pueden tener parámetros.
- b) Pueden devolver un valor.
- c) Siempre son públicos.
- d) Pueden ser estáticos o no estáticos.

68. Constructores ¿Qué ocurre si una clase no tiene un constructor explícito definido?

- a) No se puede crear un objeto de esa clase.
- b) Se genera automáticamente un constructor por defecto.
- c) La clase se convierte en abstracta.
- d) No se pueden inicializar variables de la clase.

69. Estructuras de control ¿Cuál es el propósito de una estructura de selección?

- a) Repetir un bloque de código varias veces.
- b) Decidir qué bloque de código ejecutar según una condición.
- c) Saltar líneas de código específicas.
- d) Manejar errores en tiempo de ejecución.

70. Estructuras de repetición ¿Cuál de las siguientes estructuras repite un bloque de código mientras una condición es verdadera?

- a) if-else
- b) switch
- c) while
- d) break

71. Excepciones ¿Qué palabra clave se utiliza en Java para manejar excepciones?

- a) throw
- b) catch
- c) try
- d) Todas las anteriores

72. Visibilidad ¿Qué modificador de visibilidad permite que un miembro de la clase sea accesible sólo dentro de su paquete?

- a) public
- b) private
- c) protected
- d) Sin modificador (default)

73. Herencia ¿Verdadero o falso? La herencia permite que una clase hija reutilice el código de su clase padre.

- a) Verdadero
- b) Falso

74. Sobrecarga de métodos ¿Cuál de las siguientes opciones describe correctamente la sobrecarga de métodos?

- a) Usar el mismo nombre de método pero con diferentes firmas.
- b) Usar el mismo nombre de método en diferentes clases sin herencia.
- c) Cambiar el nombre de un método heredado.
- d) Crear un método que no devuelva nada.

75. Flujos de entrada/salida ¿Verdadero o falso? Los flujos de entrada/salida permiten la comunicación entre un programa y el entorno externo.

- a) Verdadero
- b) Falso

76. Estructuras de datos ¿Qué tipo de estructura de datos utiliza posiciones fijas en memoria?

- a) Listas
- b) Arrays
- c) Conjuntos
- d) Diccionarios

77. Colecciones ¿Qué característica tienen los ArrayList frente a los arrays?

- a) Tamaño fijo
- b) Capacidad de crecer y decrecer
- c) Sólo almacenan números
- d) Requieren definición de tamaño al inicio

78. Tipos de datos ¿Cuál de los siguientes tipos de datos permite almacenar números con decimales?

- a) int
- b) double
- c) char
- d) boolean

79. Variables ¿Qué sucede si se intenta usar una variable local sin inicializar?

- a) Genera un error en tiempo de ejecución.
- b) Se inicializa automáticamente a cero.
- c) Genera un error en tiempo de compilación.
- d) La variable toma un valor aleatorio.