1. ¿Cuál es el modificador de acceso más restrictivo que permite el acceso dentro del mismo paquete?
   a) public
   b) Protected
   c) Private **X**
   d) (sin modificador)

2. ¿Qué método se utiliza para comparar el contenido de dos objetos en Java?
   a) equals() **X**
   b) compare()
   c) match()
   d) ==

3. ¿Cuál de las siguientes opciones es verdadera sobre herencia en Java?
   a) Java permite la herencia múltiple de clases
   b) Todas las clases heredan directamente de la clase Object **X**
   c) No se puede sobrescribir un método en una clase hija
   d) La herencia solo funciona son clases abstractas

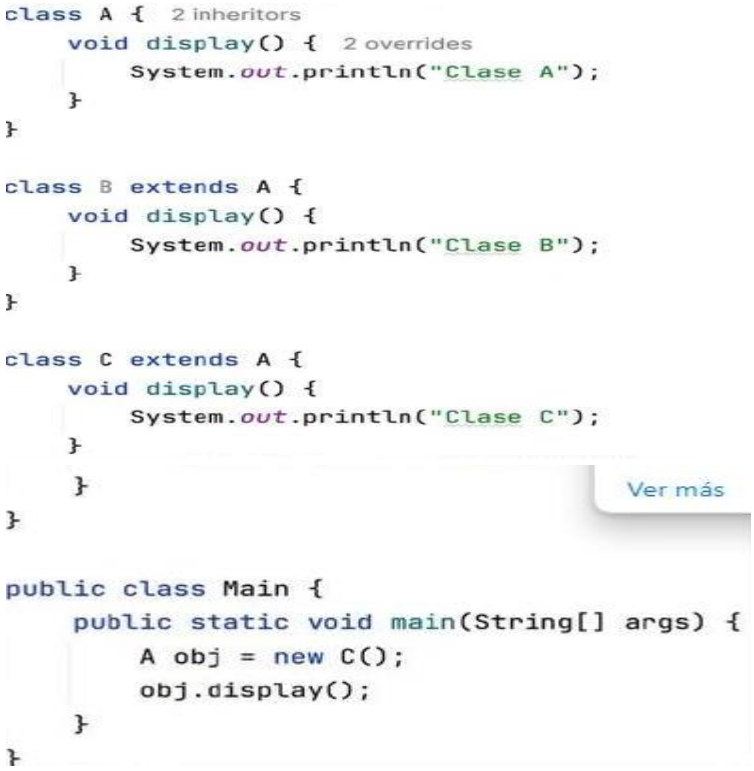
4. ¿Cuál de estas afirmaciones sobre métodos sobrescritos es correcta?
   a) Deben tener el mismo nombre y parámetros que en la clase padre **X**
   b) Pueden tener cualquier tipo de modificador de acceso
   c) Deben tener el modificador final
   d) Pueden tener un número diferente de parámetros

5. ¿Qué palabra clave se utiliza para acceder a un método o constructor de la clase padre?
   a) super **X**
   b) parent
   c) this
   d) override

6. ¿Qué método de la clase Object puede sobrescribirse para definir la representación textual de un objeto?

   a) toString() **X**
   b) hashCode()
   c) equals()
   d) getClass()

7. ¿Qué ocurre al compilar y ejecutar el siguiente código?



a) Imprime A
b) Imprime B
c) Imprime C **X**
d) Error de compilación

8. ¿Qué sucede si se define un método sobrecargado con la misma firma, pero tipos de parámetros diferentes en una clase?
   a) Se genera un error de compilación
   b) El compilador determina cuál método llamar en tiempo de ejecución
   c) El compilador determina cuál método llamar en tiempo de compilación **X**
   d) El método con más parámetros será ignorado.

9. Una clase A es ascendiente de otra clase B
   a) Cuando la relación entre A y B es directa
   b) Cuando la clase A está por encima de la clase B en la jerarquía **X**
   c) Cuando la clase B está por encima de la clase A en la jerarquía
   d) Ninguna de las anteriores es correcta

10. ¿Qué es lo que imprime el siguiente código?

```
class Parent {  1 inheritor
    void display() {
        System.out.println("Parent");
    }
}

class Child extends Parent {
    void display(String msg) {
        System.out.println("Child: " + msg);
    }
}

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

a) Parent **X**
b) Child: null
c) Error de compilación
d) NullPointerException

11. ¿Qué pasa si intentas asignar un objeto de la subclase a una referencia de la clase padre sin usar casting?
    a) El programa fallará en tiempo de compilación
    b) El programa funciona, pero los métodos de la subclase no son accesibles **X**
    c) El programa genera una excepción en tiempo de ejecución
    d) No ocurre nada, ya que Java permite automáticamente la conversión.

12. ¿Qué se entiende por polimorfismo en Java?
    a) La capacidad de una clase para tener múltiples métodos con el mismo nombre pero diferentes
    b) La capacidad de un objeto para tomar múltiples formas dependiendo del contexto **X**
    c) La capacidad de una clase para heredar de múltiples clases
    d) La capacidad de un objeto para crear nuevas instancias de otras clases

13. ¿Qué es instanceof en Java?
    a) Un operador utilizado para verificar el tipo de una clase **X**
    b) Un tipo de variable que puede contener instancias de diferentes clases
    c) Un método para crear una instancia de una clase
    d) Un tipo de casting que se usa en tiempo de ejecución

14. ¿Cuál es la diferencia entre sobrecarga y sobreescritura de métodos en Java?
    a) La sobrecarga implica tener múltiples métodos con el mismo nombre pero diferentes parámetros, mientras que la sobreescritura implica redefinir un método de una clase padre en una subclase **X**
    b) La sobreescritura implica tener múltiples métodos con el mismo nombre pero diferentes parámetros, mientras que la sobrecarga implica redefinir un método de una clase padre en una subclase
    c) La sobrecarga ocurre solo con métodos estáticos
    d) La sobreescritura ocurre solo con métodos estáticos

15. ¿Qué imprime este código?

```
class Animal {  1 inheritor
    void hacerSonido() {  1 override
        System.out.println("Sonido de animal");
    }
}

class Perro extends Animal {
    void hacerSonido() {
        System.out.println("El perro ladra");
    }
}

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

    a) El perro ladra **X**
    b) Sonido de animal
    c) Error en tiempo de ejecución
    d) NullPointerException

16. ¿Qué significa instancias una clase?
    a) Duplicar una clase
    b) Eliminar una clase
    c) Crear un objeto a partir de la clase **X**
    d) Conectar dos clases entre sí

17. ¿Qué ocurre cuando se invoca el método super () en un constructor de una clase hija?
    a) Invoca el constructor de la clase hija
    b) Invoca el constructor de la clase padre **X**
    c) No se puede invocar super() en un constructor
    d) El compilador elige automáticamente el constructor adecuado

18. ¿Qué clase en Java se encuentra en la raíz de la jerarquía de herencia?
    a) Object **X**
    b) Class
    c) Root
    d) JavaLang

19. ¿Cuál es el resultado del siguiente fragmento de código?

```
class Parent {  1 inheritor
    void display() {  1 override
        System.out.println("Parent");
    }
}

class Child extends Parent {
    void display(String msg) {
        System.out.println("Child: " + msg);
    }
}

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

    a) Parent **X**
    b) Child: null
    c) Error de compilación
    d) Ninguna de las anteriores

20. ¿Qué ocurrirá al compilar y ejecutar este código?

```java
class A {  1 inheritor
    void display() {  1 override
        System.out.println("Clase A");
    }
}

class B extends A {
    void display() {
        System.out.println("Clase B");
    }
}

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

a) Clase A
b) Clase B **X**
c) Error de compilación
d) Error en tiempo de ejecución