1.- ¿Qué palabra clave se utiliza para acceder a un método o constructor de la clase padre?
A. super
B. parent
C. this
D. override

2.- ¿Qué sucede si se llama a un método sobrescrito usando una referencia de la clase padre?
A. Se ejecuta el método de la clase padre.
B. Se ejecuta el método de la clase hija.
C. Se genera un error de compilación.
D. Se genera un error en tiempo de ejecución.

**3.-** ¿Qué es el casting en Java?
A. Convertir un objeto de un tipo a otro.
B. Crear una nueva instancia de una clase.
C. Heredar de múltiples clases.
D. Crear un método estático.

4.- ¿Qué sucede si una subclase omite la anotación @Override al sobrescribir un método?
A. Se genera un error de compilación.
B. El método no será sobrescrito correctamente si no coincide la firma.
C. El método será automáticamente considerado sobrescrito.
D. El método original será eliminado.

**5.-** ¿Cuál de las siguientes relaciones es válida entre dos clases?
A. Una relación "es un".
B. Una relación "tiene un".
C. Ambas A y B.
D. Ninguna de las anteriores.

**6.- ¿Cuál es el resultado del siguiente fragmento de código?**

```
class Parent {
  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
B. Child: null
C. Error de compilación
D. Ninguna de las anteriores

**7.- Dado el siguiente código, ¿qué sucede al ejecutarlo?**

```
class A {
  private void display() {
    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();
    obj.display();
  }
}
```
A. Clase A
B. Clase B
C. Error de compilación
D. Error en tiempo de ejecución

**8.- ¿Qué ocurrirá al compilar y ejecutar este código?**

```
class A {
  void display() {
    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
C. Error de compilación
D. Error en tiempo de ejecución

**9.- ¿Qué operador se utiliza para comprobar si un objeto pertenece a una clase específica?**

A. instanceof
B. super
C. this
D. cast

**10.- ¿Qué ocurre al compilar y ejecutar el siguiente código?**

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

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

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

A. Parent
B. Child
C. Error en tiempo de ejecución
D. Error de compilación

**11.-¿Qué ocurre al llamar a un método sobrecargado con parámetros incorrectos?**

A. El compilador generará un error.
B. El programa se compilará, pero fallará en tiempo de ejecución.
C. El compilador seleccionará automáticamente un método con los parámetros más cercanos.
D. El compilador ignorará el método.

**12.- En el siguiente código, ¿qué será impreso al ejecutarlo?**

```
class A {
  void display(int num) {
    System.out.println("Clase A con un parámetro: " + num);
  }
}

class B extends A {
  void display(int num1, int num2) {
    System.out.println("Clase B con dos parámetros: " + num1 + ", " + num2);
  }
}

public class Main {
  public static void main(String[] args) {
    A obj = new B();
    obj.display(10, 20);
  }
}
```

A. Clase A con un parámetro: 10
B. Clase B con dos parámetros: 10, 20
C. Error de compilación
D. Error en tiempo de ejecución

**13.- ¿Qué es lo que imprime el siguiente código?**

```
class A {
  int a = 10;
}

class B extends A {
  int a = 20;
}

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

```
    System.out.println(obj.a);
  }
}
```
A. 10
B. 20
C. Error de compilación
D. NullPointerException

**14.- ¿Qué pasa si intentas acceder a un atributo private en una clase heredada?**
A. El atributo puede ser accedido directamente.
B. El atributo puede ser accedido usando super.
C. El atributo no es accesible, incluso usando super.
D. El atributo se convierte en público.

**15.-¿Cuál es el resultado de este código?**

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

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

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

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

D. Error de compilación

**16.- ¿Cuál es el resultado de este código si se usa el casting?**

```
class A {
  void display() {
    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 objB = (B) obj;
    objB.display();
  }
}
```

A. Clase A
B. Clase B
C. Error de compilación
D. NullPointerException

**17.- ¿Qué se imprime al ejecutar este código?**

java
Copiar código

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

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

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

```
}

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

A. Clase A, Clase B
B. Clase B, Clase C
C. Clase A, Clase C
D. Error de compilación

**18.- ¿Qué pasa si haces un upcasting en este caso?**

```
class A {
  void display() {
    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) {
    B obj = new B();
    A objA = obj;  // Upcasting
    objA.display();
  }
}
```

A. Clase A
B. Clase B
C. Error de compilación
D. NullPointerException

**19.- ¿Qué imprime este código?**

```
class A {
  void display() {
    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 A();
    obj.display();
    obj = new B();
    obj.display();
  }
}
```

A. Clase A, Clase A
B. Clase B, Clase A
C. Clase A, Clase B
D. Clase B, Clase B

**20.- ¿Cuál es la salida de este código?**

```
class Animal {
  void sound() {
    System.out.println("Animal hace un sonido");
  }
}

class Dog extends Animal {
  @Override
  void sound() {
    System.out.println("El perro ladra");
  }
}

public class Main {
  public static void main(String[] args) {
    Animal obj = new Dog();
    obj.sound();
  }
}
```

A. Animal hace un sonido
B. El perro ladra
C. Error de compilación
D. No se imprime nada