1. ¿Qué es una clase en programación orientada a objetos?

- Un tipo de dato que almacena una sola variable
- Un modelo o plantilla para crear objetos ✓
- Una función que realiza una tarea específica
- Una colección de variables globales

2. ¿Qué es un objeto en programación orientada a objetos?

- Una instancia de una clase ✓
- Un tipo de dato abstracto
- Un archivo que contiene código fuente
- Un conjunto de funciones relacionadas

3. ¿Cuál de las siguientes afirmaciones sobre atributos es correcta?

- Los atributos son funciones que definen el comportamiento
- Los atributos son variables dentro de una clase ✓
- Los atributos son siempre públicos
- Los atributos no pueden ser modificados

4. ¿Qué es un método en programación orientada a objetos?

- Un procedimiento asociado a una clase ✓
- Una variable que almacena datos
- Una función global que no pertenece a ninguna clase
- Un tipo de dato que no puede ser modificado

5. ¿Qué sucede cuando se crea un objeto de una clase?

- Se define un nuevo tipo de dato
- Se ejecuta un método estático
- Se crea una instancia de esa clase ✓
- Se borra una instancia de esa clase

6. ¿Cómo se llama el proceso de ocultar los detalles internos de un objeto y exponer solo lo necesario?

- Herencia
- Encapsulamiento ✓
- Polimorfismo
- Abstracción

7. ¿Qué término describe mejor la capacidad de una clase de heredar atributos y métodos de otra clase?

- Polimorfismo
- Encapsulamiento
- Herencia ✓
- Abstracción

8. ¿Cuál es la diferencia principal entre un método y un atributo?

- Los métodos pueden ser públicos, los atributos no
- Los atributos almacenan datos, los métodos definen comportamiento ✓
- Los atributos son funciones, los métodos son variables
- No hay diferencia, son lo mismo

9. ¿Qué describe mejor el concepto de polimorfismo en programación orientada a objetos?

- La capacidad de una clase de derivar otra clase
- La capacidad de un objeto de cambiar su tipo dinámicamente
- La capacidad de un método de tener diferentes comportamientos ✓
- La capacidad de un objeto de ocultar sus detalles internos

10. ¿Cuál de las siguientes opciones describe mejor un constructor en una clase?

- Un método que se llama automáticamente al crear un objeto ✓
- Una función que debe llamarse explícitamente
- Una variable que inicializa un atributo
- Una clase que se utiliza para construir objetos

11. ¿Qué es un modificador de acceso en Java?

- Una palabra clave que define el alcance de una clase, método o variable ✓
- Un tipo de dato especial
- Una función que se ejecuta al inicio del programa
- Una interfaz que implementa múltiples métodos

12. ¿Cuál es el modificador de acceso más restrictivo?

- public
- protected
- default
- private ✓

13. ¿Para qué se utiliza la palabra reservada 'super' en Java?

- Para crear una nueva instancia de una clase
- Para referirse a la clase padre ✓
- Para finalizar la ejecución de un programa
- Para definir una constante

14. ¿Qué significa cuando una clase está marcada como 'final'?

- No se puede instanciar
- No se puede heredar ✓
- No se puede modificar
- No se puede eliminar

15. ¿Cuál de las siguientes es una variable de tipo primitivo en Java?

- String
- Integer
- boolean ✓
- ArrayList

16. ¿Qué es una variable local?

- Una variable declarada dentro de una clase
- Una variable declarada dentro de un método ✓
- Una variable que siempre es pública
- Una variable que solo puede ser leída, no modificada

17. ¿Cuál de los siguientes modificadores de acceso permite que un método sea accesible desde cualquier lugar?

- private
- public ✓
- protected
- default

18. ¿Qué es una variable de instancia?

- Una variable que se define dentro de un método
- Una variable que pertenece a una instancia específica de una clase ✓
- Una variable que está estática y compartida entre todas las instancias
- Una variable que se utiliza solo para constantes

19. ¿Para qué se utiliza la palabra reservada 'final' en una variable?

- Para permitir que su valor cambie
- Para prevenir que su valor cambie ✓
- Para declarar una variable estática
- Para definir una variable local

20. ¿Qué modificador de acceso se utiliza si no se especifica explícitamente en Java?

- private
- public
- protected
- default ✓

21. ¿Cuál de las siguientes afirmaciones es verdadera acerca de los tipos de datos primitivos en Java?

- Son objetos
- Pueden ser nulos
- No tienen métodos asociados
- Son siempre constantes ✓

22. ¿Qué es un parámetro en el contexto de un método en Java?

- Una variable local dentro de un método
- Un valor que se pasa a un método para influir en su comportamiento ✓
- Un tipo de retorno de un método
- Una clase que se utiliza en el método

23. ¿Qué es un constructor en Java?

- Un método que se llama automáticamente al crear un objeto ✓
- Un método que inicializa variables estáticas
- Una función que siempre retorna un valor
- Una variable que almacena datos

24. ¿Cuál es la diferencia principal entre un constructor y un método?

- Un constructor puede tener cualquier nombre
- Un método no puede sobrecargarse
- Un constructor no tiene tipo de retorno ✓
- Un método se llama automáticamente al crear un objeto

25. ¿Qué es la sobrecarga de métodos en Java?

- Declarar dos métodos con el mismo nombre y diferentes parámetros ✓
- Crear un método con el mismo nombre en dos clases diferentes
- Usar un método de una clase padre en una clase hija
- Crear un método sin ningún parámetro

26. ¿Cuál de los siguientes no es un modificador válido para un constructor en Java?

- public
- private
- static ✓
- protected

27. ¿Cuál es el propósito principal de un constructor en una clase?

- Instanciar los objetos de la clase ✓
- Destruir los objetos de la clase
- Definir variables estáticas
- Sobrescribir métodos de la clase padre

28. ¿Qué sucede cuando se declara un método como 'static'?

- Puede ser accedido sin crear una instancia de la clase ✓
- Solo puede ser accedido por objetos de la clase
- Se convierte en una constante
- No puede tener parámetros

29. ¿Qué es la sobrecarga de métodos en Java?

- Tener dos métodos con el mismo nombre pero diferentes parámetros ✓
- Tener dos métodos con el mismo nombre y los mismos parámetros
- Definir un método dentro de una clase abstracta
- Cambiar el tipo de retorno de un método

30. ¿Qué tipo de método no puede ser sobreescrito?

- Un método no estático
- Un método final ✓
- Un método público
- Un método protegido

31. ¿Qué diferencia a un constructor de un método regular en Java?

- Un constructor no puede tener parámetros
- Un constructor no tiene tipo de retorno ✓
- Un constructor debe ser estático
- Un constructor se llama manualmente

32. ¿Cómo se define un método constructor en una clase en Java?

- Con el mismo nombre de la clase y sin tipo de retorno ✓
- Con un nombre diferente al de la clase y un tipo de retorno
- Con el modificador 'static'
- Con el modificador 'final'

33. ¿Cuál es la ventaja de la sobrecarga de métodos en Java?

- Permite definir varios métodos con diferentes nombres
- Permite usar el mismo nombre de método para diferentes funcionalidades ✓
- Facilita el uso de herencia
- Restringe el uso de parámetros en métodos

34. ¿Qué pasa si no se define un constructor en una clase?

- La clase no podrá ser instanciada
- Se utilizará un constructor por defecto proporcionado por Java ✓
- Se lanzará un error de compilación
- Los objetos de la clase no podrán tener métodos

35. ¿Qué es la modularidad en el contexto de la programación orientada a objetos?

- Dividir el programa en partes independientes y reutilizables ✓
- Definir todas las variables como estáticas
- Usar exclusivamente variables globales
- Ejecutar todo el código en un solo método principal

36. ¿Qué se busca lograr con una alta cohesión y bajo acoplamiento en el diseño de software?

- Mayor dificultad en el mantenimiento del código
- Menor flexibilidad y mayor rigidez
- Mejor mantenimiento, flexibilidad y reutilización ✓
- Mayor complejidad en la estructura del código

37. ¿Qué es la herencia en programación orientada a objetos?

- La capacidad de una clase de derivar de otra clase ✓
- La capacidad de un objeto de cambiar su tipo
- La capacidad de una clase de tener múltiples constructores
- La capacidad de un método de ser llamado desde cualquier parte

38. ¿Cuál es el propósito principal del polimorfismo en programación orientada a objetos?

- Permitir que una clase herede de múltiples clases
- Permitir que un método se comporte de diferentes maneras según el objeto ✓
- Permitir la creación de variables sin tipo específico
- Permitir el uso de múltiples hilos en un programa

39. ¿Cuál es el propósito principal del polimorfismo en programación orientada a objetos?

- Permitir que una clase herede de múltiples clases
- Permitir que un método se comporte de diferentes maneras según el objeto ✓
- Permitir la creación de variables sin tipo específico
- Permitir el uso de múltiples hilos en un programa

40. ¿Qué es una clase abstracta en Java?

- Una clase que no puede ser instanciada ✓
- Una clase que no puede ser heredada
- Una clase que solo tiene métodos estáticos
- Una clase que tiene un único constructor

41. ¿Cómo se relacionan la herencia y el polimorfismo en Java?

- La herencia impide el uso del polimorfismo
- El polimorfismo requiere de la herencia ✓
- Son conceptos independientes
- No pueden ser utilizados juntos

42. ¿Qué palabra clave en Java se utiliza para declarar una clase que no puede ser instanciada?

- final
- static
- abstract ✓
- public

43. ¿Cuándo es adecuado extender una clase en Java?

- Cuando se necesita agregar funcionalidad a una clase existente ✓
- Cuando se quiere cambiar el nombre de una clase
- Cuando se necesita una clase con menos métodos
- Cuando se necesita evitar el uso de métodos finales

44. ¿Qué método de la clase Object se utiliza para comparar si dos objetos son iguales?

- clone()
- toString()
- equals() ✓
- hashCode()

45. ¿Qué palabra clave se usa para llamar al constructor de la superclase desde un subclase?

- this
- super ✓
- parent
- base

46. ¿Qué palabra clave se usa para prevenir la herencia de una clase?

- static
- final ✓
- abstract
- protected

47. ¿Cómo se declara una clase que no debe ser extendida?

- class ClaseFinal
- final class ClaseFinal ✓
- abstract class ClaseFinal
- protected class ClaseFinal

48. ¿Cuál es el objetivo de usar la palabra clave 'super' en un método de una subclase?

- Llamar a un método de la misma clase
- Llamar a un método de la superclase ✓
- Crear una instancia de la superclase
- Declarar un método como estático

49. ¿Qué sucede si una clase no sobrescribe el método equals()?

- La clase no podrá compararse

- La clase usará la implementación de equals() de la clase Object ✓
- La clase no podrá instanciarse
- La clase no podrá tener métodos propios

50. ¿Qué método de la clase Object se debe sobreescribir para permitir el uso en colecciones que usan hashing?

- toString()
- finalize()
- clone()
- hashCode() ✓

51. ¿Qué diferencia a un constructor de un método regular en Java?

- Un constructor no puede tener parámetros
- Un constructor no tiene tipo de retorno ✓
- Un constructor debe ser estático
- Un constructor se llama manualmente

52. ¿Cómo se define un método constructor en una clase en Java?

- Con el mismo nombre de la clase y sin tipo de retorno ✓
- Con un nombre diferente al de la clase y un tipo de retorno
- Con el modificador 'static'
- Con el modificador 'final'

53. ¿Cuándo es adecuado crear una clase abstracta en Java?

- Cuando se necesita una clase que no puede ser instanciada directamente ✓
- Cuando se necesita una clase con métodos privados
- Cuando se quiere evitar la herencia
- Cuando se necesita una clase con métodos finales

54. ¿Qué es un método abstracto en Java?

- Un método que no tiene implementación ✓
- Un método que solo puede ser llamado desde la misma clase
- Un método que no puede ser sobreescrito
- Un método que solo puede ser llamado desde subclases

55. ¿Qué es un método abstracto en Java?

- Un método que no tiene implementación ✓

- Un método que solo puede ser llamado desde la misma clase
- Un método que no puede ser sobreescrito
- Un método que solo puede ser llamado desde subclases

56. ¿Cuál es el propósito de una clase abstracta?

- Proveer una implementación completa de métodos
- Proveer una base para que otras clases puedan derivar e implementar métodos abstractos ✓
- Proveer solo métodos privados
- Proveer solo métodos públicos

57. ¿Cómo se declara una clase abstracta en Java?

- class NombreClase abstract
- final abstract class NombreClase
- abstract class NombreClase ✓
- static class NombreClase

58. ¿Qué es una interfaz en Java?

- Una clase que puede tener múltiples constructores
- Una referencia a un objeto que puede cambiar
- Un contrato que define métodos que una clase debe implementar ✓
- Una clase que no puede ser heredada

59. ¿Cuál es la principal diferencia entre la herencia múltiple y la herencia simple?

- La herencia múltiple permite una clase heredar de múltiples clases, mientras que la herencia simple permite heredar de una sola clase ✓
- La herencia simple permite heredar métodos estáticos, mientras que la herencia múltiple permite heredar de una sola clase
- La herencia múltiple permite heredar métodos estáticos, mientras que la herencia simple no lo permite
- La herencia simple permite heredar métodos estáticos, mientras que la herencia múltiple no lo permite

60. ¿Cuál es una ventaja de usar interfaces en lugar de herencia múltiple en Java?

- Las interfaces permiten la herencia de implementación
- Las interfaces permiten que una clase implemente múltiples interfaces ✓
- Las interfaces permiten definir constructores

- Las interfaces permiten tener variables de instancia

61. ¿Qué es una enumeración en Java?

- Una clase que puede ser extendida
- Una clase que solo contiene métodos estáticos
- Un tipo especial de clase que define un conjunto de constantes ✓
- Una clase que no puede tener métodos

62. ¿Cuándo es apropiado utilizar una enumeración en Java?

- Cuando se necesita definir un grupo de valores constantes relacionados ✓
- Cuando se necesita definir métodos estáticos
- Cuando se necesita implementar una interfaz
- Cuando se necesita definir una clase abstracta

63. ¿Qué es una Collection en Java?

- Una clase que implementa la interfaz Map
- Una interfaz que representa un grupo de objetos ✓
- Una clase abstracta para manejar datos
- Una interfaz para manejar conexiones de red

64. ¿Cuál es la principal característica de la interfaz List en Java?

- No permite elementos duplicados
- Mantiene el orden de inserción ✓
- No permite elementos nulos
- Utiliza una tabla hash para almacenar elementos

65. ¿Qué interfaz se debe usar para manejar una colección que requiere acceso por clave-valor?

- List
- Queue
- Set
- Map ✓

66. ¿Cuál es una característica distintiva de la interfaz Set en Java?

- No permite elementos duplicados ✓
- Mantiene el orden de inserción
- Permite el acceso a elementos por índice

- Permite elementos duplicados

68. ¿Qué implementación de la interfaz List es más adecuada si necesitas acceso rápido por índice?

- LinkedList
- ArrayList ✓
- HashSet
- TreeSet