

[Comisión 6 - Prog y Labo III](#) > Recuperatorio Primer parcial Programación III 2024

CUESTIONARIO

Comenzado el	Monday, 6 de May de 2024, 20:01
Estado	Finalizado
Finalizado en	Monday, 6 de May de 2024, 20:29
Tiempo empleado	28 minutos 15 segundos
Calificación	38,50 de 40,00 (96,25%)

Pregunta 1

Correcta

Se puntúa 1,00 sobre 1,00

¿Qué característica es verdadera respecto a las enumeraciones en Java?

- ☐ a. Cada constante en una enumeración puede implementar su propio método.
- ☐ b. Las enumeraciones pueden contener métodos abstractos que no requieren ser definidos.
- ☒ c. Las enumeraciones son esencialmente clases finales. ✔ Correcta, las enumeraciones son tratadas como clases finales por Java, lo que significa que no se pueden heredar.
- ☐ d. Las enumeraciones permiten la creación de subclases.

Respuesta correcta

La respuesta correcta es:


Las enumeraciones son esencialmente clases finales.

Pregunta 2

Correcta

Se puntúa 1,00 sobre 1,00

¿Cuál es una ventaja de utilizar clases abstractas sobre interfaces?

- ☐ a. Las clases abstractas permiten la herencia múltiple.
- ☐ b. Las clases abstractas proporcionan una implementación completa de todos los métodos.
- ☒ c. Las clases abstractas pueden definir tanto métodos concretos como abstractos.  Esta capacidad permite a las clases abstractas proporcionar un comportamiento predeterminado mientras definen requisitos para algunos métodos que las subclases deben implementar.
- ☐ d. Las clases abstractas eliminan la necesidad de herencia.

Respuesta correcta

La respuesta correcta es:

Las clases abstractas pueden definir tanto métodos concretos como abstractos.

Pregunta 3

Correcta

Se puntúa 1,00 sobre 1,00

Relaciona los siguientes conceptos clave de la programación orientada a objetos con sus descripciones correspondientes.

Son características o propiedades que definen el estado de un objeto, determinando sus cualidades o propiedades.



Atributos

Es un prototipo o plantilla para crear objetos. Define un tipo de datos por medio de elementos que describen datos y funciones.



Clase

Son instancias de una clase. Representan entidades concretas o abstractas del mundo real, cada uno con su propio estado y comportamiento.



Objeto

Son comportamientos o funciones que un objeto puede realizar, definidos en la clase y que permiten la interacción del objeto con otros o la realización de operaciones internas.



Funciones

Respuesta correcta

Definir

La respuesta correcta es:

Son características o propiedades que definen el estado de un objeto, determinando sus cualidades o propiedades. → Atributos,

Es un prototipo o plantilla para crear objetos. Define un tipo de datos por medio de elementos que describen datos y funciones. → Clase,

Son instancias de una clase. Representan entidades concretas o abstractas del mundo real, cada uno con su propio estado y comportamiento. → Objeto,

Son comportamientos o funciones que un objeto puede realizar, definidos en la clase y que permiten la interacción del objeto con otros o la realización de operaciones internas. → Funciones

Pregunta 4

Correcta

Se puntúa 1,00 sobre 1,00

¿Qué sucede cuando una interfaz extiende otra interfaz?

- ☐ a. Se crea una nueva implementación para todos los métodos antiguos.
 - ☐ b. Todas las interfaces deben implementar interfaces adicionales.
 - ☐ c. Debe redefinir todos los métodos de la interfaz padre.
 - ☒ d. Puede agregar nuevos métodos o constantes a la interfaz.
- ✓ Correcta, una interfaz puede extender otra interfaz y agregar nuevos métodos además de los heredados.

Respuesta correcta

La respuesta correcta es:

Puede agregar nuevos métodos o constantes a la interfaz.

Pregunta 5

Correcta

Se puntúa 1,00 sobre 1,00

¿Cuál es una posible desventaja de usar métodos estáticos en Java?

- ☐ a. Los métodos estáticos son menos eficientes que los métodos de instancia.
 - ☐ b. Los métodos estáticos pueden acceder a variables de instancia, lo que puede llevar a errores en tiempo de ejecución.
 - ☐ c. Los métodos estáticos no pueden ser parte de interfaces en Java.
 - ☒ d. Los métodos estáticos no permiten la sobrecarga y la sobreescritura en subclases.
- ✓ Correcta, debido a que los métodos estáticos pertenecen a la clase y no a la instancia, no se pueden sobrescribir en subclases, lo que puede limitar la flexibilidad y la reutilización del código.

Respuesta correcta

La respuesta correcta es:


Los métodos estáticos no permiten la sobrecarga y la sobreescritura en subclases.

Pregunta 6

Correcta

Se puntúa 1,00 sobre 1,00

Si una clase **Bicicleta** extiende a la clase **Vehículo**, ¿cuál de las siguientes afirmaciones es verdadera?

- ☐ a. **Bicicleta** puede acceder directamente a los miembros privados de **Vehículo**.
- ☐ b. **Bicicleta** no hereda ningún atributo de **Vehículo**.
- ☒ c. **Bicicleta** puede utilizar métodos públicos y protegidos de **Vehículo**.  La subclase puede acceder y utilizar métodos y atributos públicos o protegidos de su superclase.
- ☐ d. **Bicicleta** no puede sobrescribir métodos de **Vehículo**.

Respuesta correcta

La respuesta correcta es:

Bicicleta puede utilizar métodos públicos y protegidos de **Vehículo**.

Pregunta 7

Correcta

Se puntúa 1,00 sobre 1,00


Considera el siguiente fragmento de código en Java:

```
int numero = 2;
String resultado = "";

switch (numero) {
    case 1:
        resultado += "Uno";
    case 2:
        resultado += "Dos";
    case 3:
        resultado += "Tres";
        break;
    default:
        resultado += "Otro";
}

System.out.println(resultado);
```

¿Cuál será la salida impresa por este programa?

- ☐ a. UnoDosTres
- ☒ b. DosTres  Correcta, el flujo del programa entra en el caso 2 y continúa hasta el caso 3 debido a la ausencia de un **break** en el caso 2, concatenando "Dos" y "Tres".
- ☐ c. Otro
- ☐ d. Dos

Respuesta correcta

La respuesta correcta es:

DosTres

Pregunta 8

Correcta

Se puntúa 1,00 sobre 1,00

¿Cuál de las siguientes opciones describe correctamente el impacto de declarar una variable en Java con una especificación de tipo explícita?

- ☐ a. Hace que el código sea más susceptible a errores en tiempo de ejecución.
- ☒ b. Facilita la lectura y mantenimiento del código al clarificar el tipo de datos. ✔ Correcta, especificar el tipo de una variable explícitamente clarifica qué tipo de datos se espera manejar en esa variable, lo que hace que el código sea más legible y fácil de mantener.
- ☐ c. Permite que el código sea más flexible al permitir cambios de tipo en tiempo de ejecución.
- ☐ d. Aumenta la eficiencia en tiempo de ejecución de la aplicación.

Respuesta correcta

La respuesta correcta es:

Facilita la lectura y mantenimiento del código al clarificar el tipo de datos.

Pregunta 9

Correcta

Se puntúa 1,00 sobre 1,00

¿Cuál es una ventaja clave de usar interfaces en Java?

- ☒ a. Permiten la herencia múltiple de tipo. ✔ Correcta, las interfaces permiten que una clase implemente múltiples interfaces, proporcionando una forma de herencia múltiple de tipo.
- ☐ b. Reducen la necesidad de abstracción en el diseño de software.
- ☐ c. Facilitan la herencia múltiple de implementación.
- ☐ d. Aumentan la seguridad de la aplicación al restringir el acceso a métodos.

Respuesta correcta

La respuesta correcta es:

Permiten la herencia múltiple de tipo.

Pregunta 10

Correcta

Se puntúa 1,00 sobre 1,00

¿Qué característica diferencian principalmente a las clases abstractas de las interfaces en Java?

- ☐ a. Solo las interfaces pueden tener métodos con implementaciones completas.
- ☐ b. Las clases abstractas no permiten implementar métodos.
- ☒ c. Las clases abstractas pueden contener estado (atributos con valores almacenados). ✔ A diferencia de las interfaces, las clases abstractas pueden tener atributos y métodos con implementaciones concretas, lo que permite almacenar estado.
- ☐ d. Las clases abstractas no pueden tener métodos concretos.

Respuesta correcta

La respuesta correcta es:

Las clases abstractas pueden contener estado (atributos con valores almacenados).

Pregunta 11

Correcta

Se puntúa 1,00 sobre 1,00

¿Qué efecto tiene el uso de super en un método de una subclase?

- ☐ a. Sobreescribe todos los métodos en la clase base.
- ☐ b. Llama a un método de una subclase hermana.
- ☒ c. Permite que la subclase ejecute un método específico definido en la clase base. ✔ Correcta, **super** se utiliza en el cuerpo de un método de subclase para invocar un método en la clase base, útil en casos de sobreescritura de métodos.
- ☐ d. Crea un nuevo método en la clase base.

Respuesta correcta

La respuesta correcta es:


Permite que la subclase ejecute un método específico definido en la clase base.

Pregunta 12

Correcta

Se puntúa 1,00 sobre 1,00

¿Cuál es la utilidad de los modificadores de acceso en la programación orientada a objetos?

- ☒ a. Ayudan a  Los modificadores de acceso (como `private`, `public`, y `protected`) son fundamentales para controlar el acceso a los componentes de una clase, ayudando a ocultar los detalles de implementación y a mantener la integridad de los datos.
- ☐ b. Permiten que el código se ejecute más rápido.
- ☐ c. Facilitan la interacción directa con la memoria del sistema.
- ☐ d. Automatizan el proceso de pruebas y depuración de clases.

Respuesta correcta

La respuesta correcta es:


Ayudan a proteger y encapsular los datos dentro de una clase.

Pregunta 13

Incorrecta

Se puntúa 0,00 sobre 1,00

Selecciona la afirmación verdadera sobre el uso de objetos como parámetros en métodos.

- ☐ a. Solo se pueden pasar objetos de tipos primitivos como parámetros.
- ☐ b. Pasar un objeto como parámetro siempre crea una nueva instancia del objeto.
- ☒ c. Los objetos pasados como parámetros son automáticamente clonados. 
- ☐ d. Pasar un objeto a un método permite modificar el objeto original.

Respuesta incorrecta.

La respuesta correcta es:


Pasar un objeto a un método permite modificar el objeto original.

Pregunta 14

Correcta

Se puntúa 1,00 sobre 1,00

¿Cuál de los siguientes no es un error común al trabajar con clases abstractas?

- ☐ a. Declarar todos los métodos de una clase abstracta como privados.
- ☒ b. Utilizar métodos abstractos para compartir código entre varias clases.  Es un uso común y apropiado de las clases abstractas compartir código a través de métodos concretos (no abstractos), mientras que los métodos abstractos definen lo que debe ser implementado por las subclasses, no comparten código directamente.
- ☐ c. Intentar instanciar directamente la clase abstracta.
- ☐ d. No implementar todos los métodos abstractos en una subclase concreta.

Respuesta correcta

La respuesta correcta es:

Utilizar métodos abstractos para compartir código entre varias clases.

Pregunta 15

Parcialmente correcta

Se puntúa 0,50 sobre 1,00

En la programación orientada a objetos, ✓ se refiere a la agrupación de datos y métodos en una unidad compacta con restricciones de acceso especificadas, lo cual protege la información sensible y permite un manejo más seguro y organizado de los datos. ✗ es un concepto que permite a los desarrolladores centrarse en las características esenciales de un objeto, omitiendo los detalles menos importantes que no son necesarios para la implementación actual.

Por otro lado, ✗ permite que una clase herede propiedades y métodos de otra, lo que facilita la reutilización de código y promueve una estructura de diseño más clara. ✓ es la capacidad de las clases derivadas de ser tratadas como su clase base, lo cual es crucial para implementar interfaces comunes con comportamientos específicos en las subclases.

Respuesta parcialmente correcta.

Ha seleccionado correctamente 2.

La respuesta correcta es:

En la programación orientada a objetos, [el encapsulamiento] se refiere a la agrupación de datos y métodos en una unidad compacta con restricciones de acceso especificadas, lo cual protege la información sensible y permite un manejo más seguro y organizado de los datos. [la abstracción] es un concepto que permite a los desarrolladores centrarse en las características esenciales de un objeto, omitiendo los detalles menos importantes que no son necesarios para la implementación actual.

Por otro lado, [la herencia] permite que una clase herede propiedades y métodos de otra, lo que facilita la reutilización de código y promueve una estructura de diseño más clara. [el polimorfismo] es la capacidad de las clases derivadas de ser tratadas como su clase base, lo cual es crucial para implementar interfaces comunes con comportamientos específicos en las subclases.

Pregunta 16

Correcta

Se puntúa 1,00 sobre 1,00

¿Qué característica de la POO permite crear nuevas clases a partir de clases existentes?

- ☐ a. Polimorfismo
- ☒ b. Herencia ✓ La herencia permite extender o modificar el comportamiento de las clases existentes, permitiendo la reutilización y extensión del código.
- ☐ c. Abstracción
- ☐ d. Encapsulamiento

Respuesta correcta

La respuesta correcta es:

Herencia

Pregunta 17

Correcta

Se puntúa 1,00 sobre 1,00

¿Qué problema evita Java al no permitir la herencia múltiple de clases?

- ☐ a. La sobrecarga de métodos estáticos
- ☒ b. El problema del diamante ✓ Java evita la herencia múltiple de clases para prevenir el problema del diamante, que ocurre cuando una clase hereda el mismo método de múltiples superclases.
- ☐ c. El incremento en el uso de la memoria
- ☐ d. El uso ineficiente de interfaces

Respuesta correcta

La respuesta correcta es:


El problema del diamante

Pregunta 18

Correcta

Se puntúa 1,00 sobre 1,00

Selecciona el escenario más adecuado para utilizar una clase abstracta.

- ☐ a. Necesitas crear múltiples instancias de una clase con variaciones menores en el comportamiento.
- ☒ b. Requieres una base común para varias clases con algunos métodos predeterminados y otros personalizables.  Las clases abstractas son ideales cuando se necesita una estructura común con cierta flexibilidad para personalizar comportamientos específicos en subclasses.
- ☐ c. Todas las subclasses deben ser completamente independientes en términos de comportamiento.
- ☐ d. Deseas definir un contrato estricto para métodos sin proporcionar ninguna implementación predeterminada.

Respuesta correcta

La respuesta correcta es:


Requieres una base común para varias clases con algunos métodos predeterminados y otros personalizables.

Pregunta 19

Correcta

Se puntúa 1,00 sobre 1,00

¿Qué debe hacer una subclase cuando extiende una clase abstracta que contiene métodos abstractos?

- ☐ a. Nada, ya que los métodos abstractos son opcionales.
- ☐ b. Declarar también la subclase como abstracta.
- ☒ c. Implementar  Una subclase que extiende una clase abstracta debe implementar todos los métodos abstractos, a menos que la subclase también sea declarada como abstracta.
- ☐ d. Ignorar los métodos abstractos si la subclase no los necesita.

Respuesta correcta

La respuesta correcta es:

Implementar todos los métodos abstractos.

Pregunta 20

Correcta

Se puntúa 1,00 sobre 1,00

En contraste con el paradigma ✓, que se centra en procedimientos y la ejecución secuencial de ✓, el Paradigma Orientado a Objetos (POO) se enfoca en ✓ y su interacción. Mientras que en la programación procedural se prioriza la secuencia del ✓, en POO la estructuración del código facilita la ✓ y reutilización mediante la definición de clases y objetos.

Respuesta correcta

La respuesta correcta es:

En contraste con el paradigma [procedural], que se centra en procedimientos y la ejecución secuencial de [instrucciones], el Paradigma Orientado a Objetos (POO) se enfoca en [objetos] y su interacción. Mientras que en la programación procedural se prioriza la secuencia del [código], en POO la estructuración del código facilita la [modularidad] y reutilización mediante la definición de clases y objetos.

Pregunta 21

Correcta

Se puntúa 1,00 sobre 1,00

¿Cuál es un beneficio principal de utilizar herencia en la programación?

- ☐ a. Permitir que cada clase opere de manera completamente independiente.
- ☐ b. Aumentar la duplicación de código para mejorar la claridad.
- ☒ c. Evitar la duplicidad y facilitar la reutilización de código. ✓ La herencia promueve la reutilización de código al permitir que las clases derivadas utilicen campos y métodos de la clase base.
- ☐ d. Restringir el acceso a métodos específicos a través de la encapsulación.

Respuesta correcta

La respuesta correcta es:

Evitar la duplicidad y facilitar la reutilización de código.

Pregunta 22

Correcta

Se puntúa 1,00 sobre 1,00

¿Qué sucede si una subclase no sobrescribe todos los métodos abstractos de su clase base abstracta?

- ☐ a. La subclase se convierte automáticamente en una interfaz.
- ☐ b. Java compilará la subclase sin errores.
- ☒ c. La subclase también debe ser declarada como abstracta. ✔ Correcta, si una subclase no implementa todos los métodos abstractos heredados, entonces debe ser declarada como abstracta.
- ☐ d. La subclase no puede ser utilizada en ningún caso.

Respuesta correcta

La respuesta correcta es:

La subclase también debe ser declarada como abstracta.

Pregunta 23

Correcta

Se puntúa 1,00 sobre 1,00

¿Qué sucede si dos métodos en una clase tienen el mismo nombre pero diferentes parámetros?

- ☒ a. Se realiza una sobrecarga de métodos. ✔ La sobrecarga de métodos ocurre cuando dos o más métodos en una clase tienen el mismo nombre pero diferentes listas de parámetros.
- ☐ b. Los métodos se ignoran.
- ☐ c. Se reemplaza el primer método por el segundo.
- ☐ d. Ocurre un error de compilación.

Respuesta correcta

La respuesta correcta es:


Se realiza una sobrecarga de métodos.

Pregunta 24

Correcta

Se puntúa 1,00 sobre 1,00

Al definir una clase en POO, ¿qué representa el "estado" de un objeto?

- ☒ a. La combinación de valores de sus atributos en un momento dado.  El estado de un objeto se refiere al conjunto de valores de sus atributos, que pueden cambiar a lo largo del tiempo.
- ☐ b. Los métodos y la funcionalidad que el objeto ofrece.
- ☐ c. Los parámetros que se pasan a los métodos del objeto.
- ☐ d. El código fuente de la clase a la que pertenece el objeto.

Respuesta correcta

La respuesta correcta es:


La combinación de valores de sus atributos en un momento dado.

Pregunta 25

Correcta

Se puntúa 1,00 sobre 1,00

¿Qué principio de la POO es esencial para ocultar los detalles de implementación y proteger el estado de un objeto?

- ☐ a. Polimorfismo
- ☐ b. Abstracción
- ☒ c. Encapsulamiento  El encapsulamiento agrupa los datos (atributos) y los métodos en una unidad compacta que esconde los detalles de la implementación de los objetos.
- ☐ d. Herencia

Respuesta correcta

La respuesta correcta es:


Encapsulamiento

Pregunta 26

Correcta

Se puntúa 1,00 sobre 1,00

¿Cuál es el resultado de aplicar la palabra clave `final` a un método en Java?

- ☐ a. El método se elimina del compilado final.
- ☐ b. El método automáticamente se vuelve estático.
- ☒ c. El método no puede ser sobrescrito por ninguna subclase.  La palabra clave `final` cuando se aplica a un método previene que sea sobrescrito en las subclases.
- ☐ d. El método puede ser sobrescrito en cualquier subclase.

Respuesta correcta

La respuesta correcta es:


El método no puede ser sobrescrito por ninguna subclase.

Pregunta 27

Correcta

Se puntúa 1,00 sobre 1,00

¿Qué sucede si una clase implementa una interfaz pero no proporciona implementaciones para todos sus métodos?

- ☐ a. La clase compila sin errores.
- ☒ b. La clase debe ser declarada como abstracta.  Correcta, si una clase implementa una interfaz pero no implementa todos los métodos, debe ser declarada como abstracta.
- ☐ c. La clase se convierte automáticamente en una interfaz.
- ☐ d. Java automáticamente provee las implementaciones faltantes.

Respuesta correcta

La respuesta correcta es:

La clase debe ser declarada como abstracta.

Pregunta 28

Correcta

Se puntúa 1,00 sobre 1,00

```
public class Vehiculo {
    private String marca;
    private int year;

    public Vehiculo(String marca, int year) {
        this.marca = marca;
        this.year = year;
    }

    public String getMarca() {
        return marca;
    }

    public void setMarca(String marca) {
        this.marca = marca;
    }


    public int getYear() {
        return year;
    }

    public void setYear(int year) {
        this.year = year;
    }

    public void mostrarInformacion() {
        System.out.println("Marca: " + marca + ", Año: " + year);
    }
}

public class Main {
    public static void main(String[] args) {
        Vehiculo miCarro = new Vehiculo("Toyota", 2021);
        miCarro.mostrarInformacion();
        miCarro.setMarca("Honda");
        miCarro.mostrarInformacion();
    }
}
```

¿Qué concepto del paradigma orientado a objetos demuestra el método `setMarca` en la clase `Vehiculo`?

- ☒ a. Encapsulamiento  Correcta, el encapsulamiento se muestra mediante el uso de métodos `set` para controlar el acceso y la modificación de los atributos privados de la clase.
- ☐ b. Herencia
- ☐ c. Polimorfismo
- ☐ d. Abstracción

Respuesta correcta

La respuesta correcta es:
Encapsulamiento

Pregunta 29

Correcta

Se puntúa 1,00 sobre 1,00

```
public class Vehiculo {
    private String marca;
    private int year;

    public Vehiculo(String marca, int year) {
        this.marca = marca;
        this.year = year;
    }

    public String getMarca() {
        return marca;
    }

    public void setMarca(String marca) {
        this.marca = marca;
    }


    public int getYear() {
        return year;
    }

    public void setYear(int year) {
        this.year = year;
    }

    public void mostrarInformacion() {
        System.out.println("Marca: " + marca + ", Año: " + year);
    }
}

public class Main {
    public static void main(String[] args) {
        Vehiculo miCarro = new Vehiculo("Toyota", 2021);
        miCarro.mostrarInformacion();
        miCarro.setMarca("Honda");
        miCarro.mostrarInformacion();
    }
}
```

Al final del **main**, ¿cuál es el estado del objeto **miCarro**?

- ☒ a. Marca:  Correcta, muestra el estado actualizado del objeto Honda, **miCarro** después de cambiar la marca y mantener el Año: 2021 año.
- ☐ b. No se puede determinar sin ejecutar el código.
- ☐ c. Marca: Toyota, Año: 2021
- ☐ d. Marca: Honda, Año: 2020

Respuesta correcta

La respuesta correcta es:

Marca: Honda, Año: 2021

Pregunta 30

Correcta

Se puntúa 1,00 sobre 1,00

¿En qué situación es más adecuado utilizar un bucle `do-while` en lugar de un bucle `while`?

- ☐ a. Cuando se necesita iterar sobre un conjunto de elementos en una colección.
 - ☐ b. Cuando se desconoce el número de veces que debe ejecutarse el bucle.
 - ☐ c. Cuando la condición que controla el bucle es extremadamente compleja.
 - ☒ d. Cuando el bucle debe ejecutarse al menos una vez independientemente de la condición inicial.
- ✓ Correcta, el bucle `do-while` garantiza que el cuerpo del bucle se ejecute al menos una vez antes de evaluar la condición por primera vez, lo que es útil cuando se necesita que el bloque de código se ejecute sin importar la condición inicial.

Respuesta correcta

La respuesta correcta es:

Cuando el bucle debe ejecutarse al menos una vez independientemente de la condición inicial.

Pregunta 31

Correcta

Se puntúa 1,00 sobre 1,00

¿Qué es necesario para que una clase derivada en Java pueda sobrescribir un método de su clase base?

- ☐ a. El método debe ser estático.
 - ☐ b. Todos los métodos en la clase base deben ser privados.
 - ☒ c. El método en la clase derivada debe tener el mismo nombre y parámetros.
 - ☐ d. La clase base debe ser declarada como `final`.
- ✓ Para sobrescribir un método, la subclase debe definir un método con la misma firma que el método en la superclase.

Respuesta correcta

La respuesta correcta es:

El método en la clase derivada debe tener el mismo nombre y parámetros.

Pregunta 32

Correcta

Se puntúa 1,00 sobre 1,00

```
public class Main {  
    public static void main(String[] args) {  
        Saludador saludador = new Saludador () {  
            @Override  
            public void saludar () {  
                System.out.println("Hola, mundo!");  
            }  
        };  
  
        realizarAccion(saludador);  
    }  
  
    public static void realizarAccion(Saludador  
saludador) {  
        saludador.saludar();  
    }  
  
    interface Saludador {  
        void saludar();  
    }  
}
```

Respuesta correcta

La respuesta correcta es:

```
public class Main {  
    public static void main(String[] args) {  
        Saludador saludador = [new Saludador]() {  
            @Override  
            public void [saludar]() {  
                System.out.println("Hola, mundo!");  
            }  
        };  
  
        realizarAccion(saludador);  
    }  
  
    public static void realizarAccion([Saludador] saludador) {  
        saludador.saludar();  
    }  
  
    interface Saludador {  
        void saludar();  
    }  
}
```

Pregunta 33

Correcta

Se puntúa 1,00 sobre 1,00

¿Cuál es la diferencia principal entre una interfaz y una clase abstracta en Java?

- ☐ a. Las interfaces permiten la herencia múltiple, mientras que las clases abstractas no lo permiten.
- ☒ b. Solo las clases abstractas pueden contener métodos con implementaciones completas. ✔ Correcta, antes de Java 8, las interfaces solo podían contener declaraciones de métodos sin ninguna implementación (todos los métodos son implícitamente abstractos), mientras que las clases abstractas podían contener tanto métodos con implementaciones completas como métodos sin implementar (abstractos).
- ☐ c. Solo las interfaces pueden tener métodos públicos.
- ☐ d. Las interfaces, a diferencia de las clases abstractas, pueden contener variables de instancia.

Respuesta correcta

La respuesta correcta es:

Solo las clases abstractas pueden contener métodos con implementaciones completas.

Pregunta 34

Correcta

Se puntúa 1,00 sobre 1,00

¿Cuál es el propósito de los métodos `get` y `set` en POO?

- ☒ a. Controlar cómo se accede y se actualiza el estado de un objeto. ✔ Los métodos `get` y `set` permiten leer y modificar los atributos de un objeto de manera controlada, respetando el principio de encapsulamiento.
- ☐ b. Definir los parámetros que los objetos deben aceptar.
- ☐ c. Determinar el tamaño de la clase.
- ☐ d. Inicializar los objetos cuando se crea una instancia de una clase.

Respuesta correcta

La respuesta correcta es:


Controlar cómo se accede y se actualiza el estado de un objeto.

Pregunta 35

Correcta

Se puntúa 1,00 sobre 1,00

¿Cuál de los siguientes NO es un beneficio directo del polimorfismo?

- ☐ a. Facilidad de mantenimiento y extensión de código
- ☐ b. Flexibilidad en la interacción entre tipos
- ☒ c. Mejora en la seguridad de acceso a datos  El polimorfismo se refiere a la capacidad de tratar objetos de clases derivadas como objetos de una clase base, lo que no tiene relación directa con la seguridad del acceso a datos.
- ☐ d. Reutilización de código

Respuesta correcta

La respuesta correcta es:


Mejora en la seguridad de acceso a datos

Pregunta 36

Correcta

Se puntúa 1,00 sobre 1,00

En el contexto de clases abstractas, ¿qué significa que un método sea abstracto?

- ☐ a. El método es opcional y las subclasses pueden decidir si implementarlo.
- ☐ b. El método no puede ser usado en la clase abstracta.
- ☐ c. El método proporciona una implementación predeterminada que las subclasses pueden sobrescribir.
- ☒ d. El método debe ser implementado por cualquier subclase no abstracta.  Los métodos abstractos en una clase abstracta deben ser implementados por las subclasses concretas; esto asegura que ciertos comportamientos esenciales sean definidos en la jerarquía de la clase.

Respuesta correcta

La respuesta correcta es:

El método debe ser implementado por cualquier subclase no abstracta.

Pregunta 37

Correcta

Se puntúa 1,00 sobre 1,00

¿Qué implica la "modularidad" en la programación orientada a objetos?

- ☐ a. Las clases pueden ser compiladas en diferentes módulos del sistema.
- ☐ b. Los objetos pueden pasar información unos a otros a través de parámetros.
- ☐ c. Los objetos se pueden modificar sin afectar a otros objetos del sistema.
- ☒ d. El código se organiza en módulos independientes que se pueden reutilizar. ✔ La modularidad en POO facilita la organización del código en componentes separados que son más fáciles de gestionar, entender y reutilizar.

Respuesta correcta

La respuesta correcta es:

El código se organiza en módulos independientes que se pueden reutilizar.

Pregunta 38

Correcta

Se puntúa 1,00 sobre 1,00

Cuando una subclase en Java necesita llamar a un constructor específico de la superclase, ¿qué debe hacer?

- ☐ a. Usar `extends` con los argumentos deseados.
- ☐ b. No es necesario llamar a un constructor específico, Java lo maneja automáticamente.
- ☐ c. Usar `this()` para llamar al constructor deseado de la superclase.
- ☒ d. Usar `super()` con los argumentos apropiados para llamar al constructor de la superclase. ✔ `super()` se utiliza para llamar a un constructor específico de la superclase, y debe coincidir con uno de los constructores definidos en la superclase.

Respuesta correcta

La respuesta correcta es:


Usar `super()` con los argumentos apropiados para llamar al constructor de la superclase.

Pregunta 39

Correcta

Se puntúa 1,00 sobre 1,00

¿Cuál es el propósito principal de utilizar clases abstractas en programación orientada a objetos?

- ☐ a. Permitir la creación directa de objetos.
- ☐ b. Reducir la necesidad de usar interfaces.
- ☒ c. Proporcionar una plantilla para otras clases que se deben extender.  Las clases abstractas no se pueden instanciar y están diseñadas para servir como una base o plantilla para otras clases.
- ☐ d. Incrementar la eficiencia de ejecución del programa.

Respuesta correcta

La respuesta correcta es:

Proporcionar una plantilla para otras clases que se deben extender.

Pregunta 40

Correcta

Se puntúa 1,00 sobre 1,00

```
abstract class Animal {
    abstract void hacerSonido();
}


class Perro extends Animal {
    @Override
    void hacerSonido() {
        System.out.println("Guau Guau!");
    }
}

class Gato extends Animal {
    @Override
    void hacerSonido() {
        System.out.println("Miau Miau!");
    }
}

public class Main {
    public static void main(String[] args) {
        Animal miAnimal = new Perro();
        Animal otroAnimal = new Gato();

        miAnimal.hacerSonido();
        otroAnimal.hacerSonido();
    }
}
```

¿Qué demuestra este código sobre el polimorfismo?

- ☐ a. El polimorfismo solo se aplica a clases que implementan interfaces, no a la herencia de clases.
- ☐ b. El polimorfismo no permite que diferentes clases hereden de una superclase común.
- ☐ c. El polimorfismo obliga a todas las subclasses a implementar los métodos de la clase base exactamente igual.
- ☒ d. El polimorfismo  Correcta, el código demuestra que objetos de diferentes clases (**Perro** y **Gato**) pueden ser tratados como instancias de su clase base (**Animal**), y el comportamiento específico de cada uno se manifiesta a través del método **hacerSonido**.
permite que diferentes objetos sean tratados como instancias de su clase base.

Respuesta correcta

La respuesta correcta es:

El polimorfismo permite que diferentes objetos sean tratados como instancias de su clase base.