

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

CUESTIONARIO



Comenzado el	Monday, 22 de April de 2024, 18:01
Estado	Finalizado
Finalizado en	Monday, 22 de April de 2024, 18:50
Tiempo empleado	48 minutos 41 segundos
Calificación	22,97 de 40,00 (57,42%)

Pregunta 1

Incorrecta

Se puntúa 0,00 sobre 1,00

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

- ☒ a. Las interfaces, a diferencia de las clases abstractas, pueden contener variables de instancia. ✗ Incorrecta, las interfaces no pueden contener variables de instancia. Todos los campos en una interfaz son automáticamente públicos, estáticos y finales, lo que significa que actúan como constantes. Las clases abstractas, en cambio, sí pueden tener variables de instancia.
- ☐ b. Solo las clases abstractas pueden contener métodos con implementaciones completas.
- ☐ c. Solo las interfaces pueden tener métodos públicos.
- ☐ d. Las interfaces permiten la herencia múltiple, mientras que las clases abstractas no lo permiten.

Respuesta incorrecta.

La respuesta correcta es:

Solo las clases abstractas pueden contener métodos con implementaciones completas.

Pregunta 2

Incorrecta

Se puntúa 0,00 sobre 1,00

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

- ☐ a. La subclase no puede ser utilizada en ningún caso.
- ☐ b. La subclase también debe ser declarada como abstracta.
- ☒ c. Java ✗ Incorrecta, no compilará a menos que todos los métodos abstractos sean sobrescritos, a menos que la subclase sea declarada como abstracta. sin errores.
- ☐ d. La subclase se convierte automáticamente en una interfaz.

Respuesta incorrecta.

La respuesta correcta es:

La subclase también debe ser declarada como abstracta.

Pregunta 3

Incorrecta

Se puntúa 0,00 sobre 1,00

¿Qué declaración sobre el tipo `var` en Java es correcta?

- ☐ a. `var` solo puede ser utilizado para declarar variables locales que sean inicializadas en su declaración.
- ☐ b. `var` puede ser utilizado para declarar tanto variables locales como globales dentro de un proyecto.
- ☒ c. `var` permite declarar variables ✗ Incorrecta, `var` no puede usarse sin especificar su tipo, lo cual puede usarse globalmente. globalmente, solo dentro de bloques de código o métodos.
- ☐ d. `var` hace que Java se convierta en un lenguaje de tipado débil.

Respuesta incorrecta.

La respuesta correcta es:

`var` solo puede ser utilizado para declarar variables locales que sean inicializadas en su declaración.

Pregunta 4

Parcialmente correcta

Se puntúa 0,67 sobre 1,00

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

Respuesta parcialmente correcta.

Ha seleccionado correctamente 2.

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 5

Incorrecta

Se puntúa 0,00 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. Otro
- ☒ b. Dos ✖ Incorrecta, porque aunque el caso 2 es el que coincide con el valor de `numero`, la falta de `break` en el caso 2 permite que la ejecución continúe al caso 3.
- ☐ c. DosTres
- ☐ d. UnoDosTres

Respuesta incorrecta.

La respuesta correcta es:

DosTres

Pregunta 6

Incorrecta

Se puntúa 0,00 sobre 1,00

¿Cuál es la función del operador ternario en Java?

- ☒ a. Comparar dos valores y retornar un booleano. ✗ Incorrecta, el operador ternario no es para comparaciones directas, aunque utiliza una condición para determinar qué valor asignar.
- ☐ b. Asignar un valor a una variable basado en una condición booleana.
- ☐ c. Iterar sobre un conjunto de valores y ejecutar una operación.
- ☐ d. Ejecutar un bloque de código repetidas veces basado en una condición.

Respuesta incorrecta.

La respuesta correcta es:

Asignar un valor a una variable basado en una condición booleana.

Pregunta 7

Correcta

Se puntúa 1,00 sobre 1,00

¿Qué sucede cuando una interfaz extiende otra interfaz?

- ☐ a. Todas las interfaces deben implementar interfaces adicionales.
- ☐ b. Debe redefinir todos los métodos de la interfaz padre.
- ☐ c. Se crea una nueva implementación para todos los métodos antiguos.
- ☒ 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 8

Incorrecta

Se puntúa 0,00 sobre 1,00

¿Cuándo es apropiado usar un método estático en Java?

- ☐ a. Cuando el método sólo se usa en una instancia de la clase.
- ☐ b. Cuando el método realiza una función que no depende del estado de instancia de cualquier objeto.
- ☐ c. Cuando el método necesita acceder y modificar el estado de un objeto específico.
- ☒ d. Cuando el método necesita ser invocado varias veces para diferentes instancias. ✖ Incorrecta, aunque un método estático puede ser invocado por diferentes instancias, esta no es la razón principal para hacer un método estático.

Respuesta incorrecta.

La respuesta correcta es:

Cuando el método realiza una función que no depende del estado de instancia de cualquier objeto.

Pregunta 9

Incorrecta

Se puntúa 0,00 sobre 1,00

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

- ☐ a. Aumentan la seguridad de la aplicación al restringir el acceso a métodos.
- ☐ b. Permiten la herencia múltiple de tipo.
- ☒ c. Reducen la necesidad de abstracción en el diseño de software. ✖ Incorrecta, las interfaces fomentan la abstracción al separar la declaración de métodos de su implementación.
- ☐ d. Facilitan la herencia múltiple de implementación.

Respuesta incorrecta.

La respuesta correcta es:

Permiten la herencia múltiple de tipo.

Pregunta 10

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. 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.
- ☐ b. Aumenta la eficiencia en tiempo de ejecución de la aplicación.
- ☐ c. Permite que el código sea más flexible al permitir cambios de tipo en tiempo de ejecución.
- ☐ d. Hace que el código sea más susceptible a errores en tiempo de ejecución.

Respuesta correcta

La respuesta correcta es:

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

Pregunta 11

Correcta

Se puntúa 1,00 sobre 1,00

Relaciona los siguientes principios SOLID con sus descripciones correspondientes.

Los objetos de una clase derivada deben poder sustituirse por objetos de su clase base sin alterar la correcta ejecución del programa.

Principio de Sustitución de Liskov (Liskov Substitution Principle)



Ningún cliente debería ser forzado a depender de métodos que no usa.

Principio de Segregación de Interfaces (Interface Segregation Principle)



Las entidades de software deben estar abiertas para la extensión, pero cerradas para la modificación.

Principio Abierto/Cerrado (Open/Closed Principle)



Los módulos de alto nivel no deberían depender de los módulos de bajo nivel.

Principio de Inversión de Dependencias (Dependency Inversion Principle)



Ambos deben depender de abstracciones.

Una clase debe tener una sola razón para cambiar, lo que implica que debe encapsular

Principio de Responsabilidad Única (Single Responsibility Principle)



una única funcionalidad o responsabilidad.

Respuesta correcta

La respuesta correcta es:

Los objetos de una clase derivada deben poder sustituirse por objetos de su

clase base sin alterar la correcta ejecución del programa. → Principio de Sustitución de Liskov (Liskov Substitution Principle),

Ningún cliente debería ser forzado a depender de métodos que no usa. → Principio de Segregación de Interfaces (Interface Segregation Principle),

Las entidades de software deben estar abiertas para la extensión, pero cerradas para la modificación. → Principio Abierto/Cerrado (Open/Closed Principle),

Los módulos de alto nivel no deberían depender de los módulos de bajo nivel. Ambos deben depender de abstracciones. → Principio de Inversión de Dependencias (Dependency Inversion Principle),


Una clase debe tener una sola razón para cambiar, lo que implica que debe encapsular una única funcionalidad o responsabilidad. → Principio de Responsabilidad Única (Single Responsibility Principle)

Pregunta 12

Correcta

Se puntúa 1,00 sobre 1,00

¿Qué implica declarar un método como final en una clase en Java?

- ☐ a. El método no puede ser sobrecargado.
- ☐ b. El método no puede ser invocado.
- ☐ c. El método se ejecutará más rápido porque es **final**.
- ☒ d. El método no puede ser sobrescrito por ninguna subclase.  Correcta, declarar un método como **final** impide que las subclases lo sobrescriban, lo cual es útil para mantener el comportamiento definido en la clase base.

Respuesta correcta

La respuesta correcta es:


El método no puede ser sobrescrito por ninguna subclase.

Pregunta 13

Incorrecta

Se puntúa 0,00 sobre 1,00

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

- ☐ a. La clase se convierte automáticamente en una interfaz.
- ☐ b. Java automáticamente provee las implementaciones faltantes.
- ☒ c. La clase  Incorrecta, la clase no compilará a menos que declare todos los métodos de la interfaz o se declare a sí misma como abstracta.
compila sin errores.
- ☐ d. La clase debe ser declarada como abstracta.

Respuesta incorrecta.

La respuesta correcta es:

La clase debe ser declarada como abstracta.

Pregunta 14

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  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**.
- ☐ b. El polimorfismo no permite que diferentes clases hereden de una superclase común.
- ☐ c. El polimorfismo solo se aplica a clases que implementan interfaces, no a la herencia de clases.
- ☐ d. El polimorfismo obliga a todas las subclases a implementar los métodos de la clase base exactamente igual.

Respuesta correcta

La respuesta correcta es:

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

Pregunta 15

Correcta

Se puntúa 1,00 sobre 1,00

Relaciona los siguientes principios fundamentales de la programación orientada a objetos con sus descripciones correspondientes.

Facilita la concentración en características esenciales de un objeto, simplificando la complejidad al ocultar detalles técnicos y operativos innecesarios.



Abstracción

Tratamiento de objetos de clases derivadas como objetos de una clase base, permitiendo múltiples implementaciones de métodos.



Polimorfismo

Protección de datos y métodos dentro de una unidad, restringiendo el acceso a detalles de implementación.



Encapsulamiento

Capacidad de crear nuevas clases basadas en clases existentes.



Herencia

Respuesta correcta

La respuesta correcta es:

Facilita la concentración en características esenciales de un objeto, simplificando la complejidad al ocultar detalles técnicos y operativos innecesarios. → Abstracción,

Tratamiento de objetos de clases derivadas como objetos de una clase base, permitiendo múltiples implementaciones de métodos. → Polimorfismo,

Protección de datos y métodos dentro de una unidad, restringiendo el acceso a detalles de implementación. → Encapsulamiento,

Capacidad de crear nuevas clases basadas en clases existentes. → Herencia

Pregunta 16

Incorrecta

Se puntúa 0,00 sobre 1,00

¿Qué afirmación es correcta sobre los atributos estáticos en Java?

- ☐ a. Los atributos estáticos pueden ser modificados por instancias individuales de la clase.
- ☐ b. Los atributos estáticos se inicializan cada vez que se crea una nueva instancia de la clase.
- ☒ c. Los atributos estáticos ✗ son útiles para almacenar información que es única para cada instancia. Incorrecta, los atributos estáticos son ideales para datos que deben ser compartidos entre todas las instancias, no para información única para cada instancia.
- ☐ d. Los atributos estáticos son compartidos entre todas las instancias de una clase.

Respuesta incorrecta.

La respuesta correcta es:

Los atributos estáticos son compartidos entre todas las instancias de una clase.

Pregunta 17

Incorrecta

Se puntúa 0,00 sobre 1,00

¿Qué declaración es verdadera sobre los métodos en una interfaz?

- ☐ a. Los métodos en las interfaces pueden almacenar estado.
- ☒ b. Los métodos ✗ pueden ser privados o protegidos. Incorrecta, los métodos en las interfaces son por defecto públicos y no pueden ser privados o protegidos.
- ☐ c. Todas las interfaces deben contener al menos un método.
- ☐ d. Los métodos en una interfaz son por defecto abstractos y públicos.

Respuesta incorrecta.

La respuesta correcta es:

Los métodos en una interfaz son por defecto abstractos y públicos.

Pregunta 18

Incorrecta

Se puntúa 0,00 sobre 1,00

¿Cómo se puede utilizar una enumeración para controlar el comportamiento en una instrucción switch?

- ☐ a. Solo las constantes de enumeraciones pueden ser usadas en una instrucción switch.
 - ☐ b. Las enumeraciones requieren un método especial para ser utilizadas en instrucciones switch.
 - ☐ c. Las enumeraciones no son compatibles con las instrucciones switch en Java.
 - ☒ d. Cualquier método de la enumeración puede ser utilizado como una etiqueta de caso en switch.
- ✖ Incorrecta, los métodos de una enumeración no pueden ser usados como etiquetas de caso; solo las constantes mismas son válidas.

Respuesta incorrecta.

La respuesta correcta es:

Solo las constantes de enumeraciones pueden ser usadas en una instrucción switch.

Pregunta 19

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. Abstracción
- ☐ c. Herencia
- ☐ d. Polimorfismo

Respuesta correcta

La respuesta correcta es:
Encapsulamiento

Pregunta 20

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();
    }
}
```

En el contexto de este código, ¿qué sucede si se intenta instanciar directamente la clase `Animal`?

- ☐ a. Se creará una instancia de `Animal` pero no podrá ejecutar el método `hacerSonido`.
- ☐ b. Se creará una instancia, pero se comportará como el último objeto instanciado (`Gato`).
- ☐ c. La instancia se creará sin problemas y ejecutará el método `hacerSonido`.
- ☒ d. Se producirá un error de compilación porque `Animal` es una clase abstracta. ✔ Correcta, intentar instanciar una clase abstracta directamente como `Animal` resulta en un error de compilación, ya que las clases abstractas no pueden ser instanciadas.

Respuesta correcta

La respuesta correcta es:

Se producirá un error de compilación porque `Animal` es una clase abstracta.

Pregunta 21

Correcta

Se puntúa 1,00 sobre 1,00

```
class Animal {  
    public void emitirSonido() {  
        System.out.println("Este animal hace un sonido.");  
    }  
  
    public String obtenerNombre() {  
        return "Animal genérico";  
    }  
}  
  
class Perro extends Animal {  
    @Override
```

```
        public void emitirSonido() {  
            System.out.println("El perro ladra.");  
        }  
  
        public String obtenerNombre(String nombre) {  
            return nombre;  
        }  
}  
  
public class TestAnimales {  
    public static void main(String[] args) {  
        Animal miAnimal = new Animal();  
        miAnimal.emitirSonido();  
  
        Perro miPerro = new Perro();  
        miPerro.emitirSonido();  
  
        System.out.println("Nombre: " + miPerro.obtenerNombre()  
("Bobby"));  
    }  
}
```

Respuesta correcta

La respuesta correcta es:

```
class Animal {  
    public void emitirSonido() {  
        System.out.println("Este animal hace un sonido.");  
    }  
  
    public [String] obtenerNombre() {  
        return "Animal genérico";  
    }  
}
```

```
class Perro extends [Animal] {  
    [Override]
```

```
    public void emitirSonido() {  
        System.out.println("El perro ladra.");  
    }  
  
    public String obtenerNombre(String nombre) {  
        return nombre;  
    }  
}  
  
public class TestAnimales {  
    public static void main(String[] args) {  
        Animal miAnimal = new Animal();  
        miAnimal.emitirSonido();  
  
        Perro miPerro = new [Perro()];  
        miPerro.emitirSonido();  
        System.out.println("Nombre: " + miPerro.[obtenerNombre]("Bobby"));  
    }  
}
```

Pregunta 22

Incorrecta

Se puntúa 0,00 sobre 1,00

En una subclase, ¿qué implica el uso incorrecto de la palabra reservada `super`?

- ☒ a. Puede llevar a un bucle infinito si `super` se llama dentro de un método que sobre escribe un método de la clase base. ✗ Incorrecta, aunque el uso indebido de `super` puede llevar a problemas de diseño, un bucle infinito no sería típicamente causado por el uso de `super` en sí.
- ☐ b. Se generará un error de compilación si `super` se usa para acceder a métodos privados en la clase base.
- ☐ c. `super` puede ser utilizado para acceder a cualquier método o variable sin importar su nivel de acceso.
- ☐ d. No tiene consecuencias; `super` siempre encuentra una manera de ejecutar el código.

Respuesta incorrecta.

La respuesta correcta es:

Se generará un error de compilación si `super` se usa para acceder a métodos privados en la clase base.

Pregunta 23

Correcta

Se puntúa 1,00 sobre 1,00

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

- ☐ a. Sobre escribe 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 24

Incorrecta

Se puntúa 0,00 sobre 1,00

¿Qué es necesario para que una enumeración pueda tener atributos y métodos?

- ☐ a. Definir un constructor que inicialice los atributos.
- ☒ b. Utilizar una interfaz que especifique los métodos y atributos. ✗ Incorrecta, las enumeraciones no necesitan implementar una interfaz para tener métodos y atributos; pueden definirlos directamente dentro de su declaración.
- ☐ c. Declarar la enumeración como abstracta.
- ☐ d. Extender la clase base java.lang.Enum.

Respuesta incorrecta.

La respuesta correcta es:

Definir un constructor que inicialice los atributos.

Pregunta 25

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 26

Correcta

Se puntúa 1,00 sobre 1,00

¿Qué caracteriza a una interfaz en Java?

- ☐ a. Permite la creación directa de objetos.
- ☐ b. Solo permite declarar métodos, no variables.
- ☒ c. Define un contrato que otras clases pueden implementar. ✔ Correcta, la principal función de una interfaz es actuar como un contrato que especifica qué métodos debe implementar una clase sin dictar cómo se deben implementar.
- ☐ d. Puede contener métodos con cuerpos definidos.

Respuesta correcta

La respuesta correcta es:

Define un contrato que otras clases pueden implementar.

Pregunta 27

Correcta

Se puntúa 1,00 sobre 1,00

¿Cuál es el propósito principal de usar la palabra reservada `super` en el contexto de un constructor?

- ☐ a. Asegurar que las subclases tengan el mismo constructor que la clase base.
- ☐ b. Prohibir la herencia del constructor de la clase base.
- ☒ c. Invocar el constructor de la clase base desde una subclase. ✔ Correcta, `super` se utiliza en el constructor de una subclase para llamar explícitamente al constructor de la clase base, asegurando que la inicialización adecuada de la clase base tenga lugar.
- ☐ d. Invocar el constructor de la subclase desde la clase base.

Respuesta correcta

La respuesta correcta es:

Invocar el constructor de la clase base desde una subclase.

Pregunta 28

Correcta

Se puntúa 1,00 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 correcta

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 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();
    }
}
```

¿Qué acción realiza el constructor en la clase `Vehiculo`?

- ☐ a. Inicializa el objeto `miCarro` con valores predeterminados.
- ☐ b. Declara tipos de datos para `marca` y `year`.
- ☐ c. Crea métodos para la clase `Vehiculo`.
- ☒ d. Asigna valores a los atributos `marca` y `year` basados en los argumentos proporcionados. ✔ Correcta, el propósito del constructor aquí es tomar los valores de `marca` y `year` proporcionados y asignarlos a los atributos del objeto.

Respuesta correcta

La respuesta correcta es:

Asigna valores a los atributos `marca` y `year` basados en los argumentos proporcionados.

Pregunta 30

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. Marca: Honda, Año: 2020
- ☐ c. Marca: Toyota, Año: 2021
- ☐ d. No se puede determinar sin ejecutar el código.

Respuesta correcta

La respuesta correcta es:

Marca: Honda, Año: 2021

Pregunta 31

Parcialmente correcta

Se puntúa 0,50 sobre 1,00

Relaciona los siguientes modificadores de acceso con sus descripciones correspondientes.

Permite la visibilidad solo dentro de la misma clase, protegiendo los datos del acceso directo desde fuera de la clase.

private



Otorga acceso a cualquier clase dentro del mismo paquete y a todas las subclases, incluso si están en diferentes paquetes, protegiendo los datos mientras se permite una herencia razonable.

default (sin modificador)



Hace que el miembro sea accesible desde cualquier otra clase ubicada en cualquier paquete, facilitando la máxima exposición y reutilización.

public



Restringe el acceso al mismo paquete, ofreciendo un nivel de protección moderado pero sin restringir el acceso a otras clases del mismo paquete.

protected



Respuesta parcialmente correcta.

Ha seleccionado correctamente 2.

La respuesta correcta es:

Permite la visibilidad solo dentro de la misma clase, protegiendo los datos del acceso directo desde fuera de la clase. → private,

Otorga acceso a cualquier clase dentro del mismo paquete y a todas las subclases, incluso si están en diferentes paquetes, protegiendo los datos mientras se permite una herencia razonable. → protected,

Hace que el miembro sea accesible desde cualquier otra clase ubicada en cualquier paquete, facilitando la máxima exposición y reutilización. → public,

Restringe el acceso al mismo paquete, ofreciendo un nivel de protección moderado pero sin restringir el acceso a otras clases del mismo paquete. → default (sin modificador)

Pregunta 32

Incorrecta

Se puntúa 0,00 sobre 1,00

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

- ☐ a. Las enumeraciones permiten la creación de subclases.
- ☐ b. Las enumeraciones son esencialmente clases finales.
- ☒ c. Cada constante ✗ Incorrecta, aunque es posible que cada en una constante tenga comportamientos específicos a enumeración través de clases anónimas internas, no pueden puede tener sus propios métodos independientes. implementar su propio método.
- ☐ d. Las enumeraciones pueden contener métodos abstractos que no requieren ser definidos.

Respuesta incorrecta.

La respuesta correcta es:

Las enumeraciones son esencialmente clases finales.

Pregunta 33

Incorrecta

Se puntúa 0,00 sobre 1,00

¿Cuál es el resultado de intentar utilizar una variable local en Java que ha sido declarada pero no inicializada dentro de un método?

- ☐ a. La variable tiene un valor predeterminado de null.
- ☐ b. El compilador genera un error porque la variable local debe ser inicializada antes de su uso.
- ☒ c. El código ✗ Incorrecta, aunque las variables de instancia compila sin errores y la pueden tener valores predeterminados como cero o variable tiene false, las variables locales no son inicializadas automáticamente y generarán un error si se usan un valor de sin ser inicializadas. cero o false.
- ☐ d. La variable automáticamente toma un valor predeterminado según su tipo.

Respuesta incorrecta.

La respuesta correcta es:

El compilador genera un error porque la variable local debe ser inicializada antes de su uso.

Pregunta 34

Correcta

Se puntúa 1,00 sobre 1,00

¿Cuál es una característica clave de las clases abstractas en Java?

- ☐ a. Pueden ser instanciadas utilizando un constructor especial.
- ☒ b. Deben ser siempre extendidas para ser utilizadas. ✓ Correcta, las clases abstractas necesitan ser extendidas por otras clases para que sus métodos abstractos sean implementados, permitiendo la creación de objetos.
- ☐ c. No pueden contener métodos con implementación completa.
- ☐ d. Sólo pueden contener métodos abstractos.

Respuesta correcta

La respuesta correcta es:

Deben ser siempre extendidas para ser utilizadas.

Pregunta 35

Correcta

Se puntúa 1,00 sobre 1,00

¿Qué efecto tiene la palabra reservada final cuando se aplica a una variable en Java?

- ☐ a. La variable puede ser inicializada una vez y luego su valor puede ser modificado.
- ☐ b. La variable es automáticamente inicializada con un valor predeterminado.
- ☒ c. La variable actúa como una constante, no puede ser reasignada después de su inicialización. ✓ Correcta, el uso principal de `final` para una variable es asegurar que actúe como una constante, es decir, que su valor no pueda cambiar una vez establecido.
- ☐ d. La variable debe ser inicializada en su declaración.

Respuesta correcta

La respuesta correcta es:

La variable actúa como una constante, no puede ser reasignada después de su inicialización.

Pregunta 36

Correcta

Se puntúa 1,00 sobre 1,00

Relaciona los siguientes términos relacionados con la gestión de memoria en Java con sus descripciones adecuadas.

Mecanismo automático de Java que se encarga de liberar memoria desalojando objetos que ya no están en uso o accesibles, ayudando a prevenir fugas de memoria.



Área de memoria donde se almacenan los objetos creados por tu programa. Es gestionada por el recolector de basura, que libera memoria eliminando objetos que ya no son accesibles.



Parte de la memoria que almacena variables locales y registros de activación de métodos. Se caracteriza por su rápido acceso y gestión automática que sigue el principio de "último en entrar, primero en salir".

**Respuesta correcta**

La respuesta correcta es:

Mecanismo automático de Java que se encarga de liberar memoria desalojando objetos que ya no están en uso o accesibles, ayudando a prevenir fugas de memoria. → Garbage Collector,

Área de memoria donde se almacenan los objetos creados por tu programa. Es gestionada por el recolector de basura, que libera memoria eliminando objetos que ya no son accesibles. → Heap,

Parte de la memoria que almacena variables locales y registros de activación de métodos. Se caracteriza por su rápido acceso y gestión automática que sigue el principio de "último en entrar, primero en salir". → Stack

Pregunta 37

Correcta

Se puntúa 1,00 sobre 1,00

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

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

Clase



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

Atributos



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



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

Objeto

**Respuesta correcta****Definir**

La respuesta correcta es:

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

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

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,


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

Pregunta 38

Correcta

Se puntúa 1,00 sobre 1,00

¿Qué significa declarar una clase como final en Java?

- ☒ a. Ninguna otra clase puede heredar de esta clase.  Correcta, el propósito de hacer una clase **final** es prevenir la herencia, asegurando que la clase no pueda ser extendida.
- ☐ b. La clase no puede tener métodos o variables.
- ☐ c. Todos los métodos en la clase son automáticamente **final**.
- ☐ d. La clase puede ser extendida por múltiples subclases.

Respuesta correcta

La respuesta correcta es:


Ninguna otra clase puede heredar de esta clase.

Pregunta 39

Incorrecta

Se puntúa 0,00 sobre 1,00

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

- ☐ a. Los métodos estáticos no permiten la sobrecarga y la sobreescritura en subclases.
- ☐ b. Los métodos estáticos son menos eficientes que los métodos de instancia.
- ☒ c. Los métodos estáticos pueden acceder a variables de instancia, lo que puede llevar a errores en tiempo de ejecución.  Incorrecta, los métodos estáticos no pueden acceder directamente a las variables de instancia, lo que evita este tipo de error.
- ☐ d. Los métodos estáticos no pueden ser parte de interfaces en Java.

Respuesta incorrecta.

La respuesta correcta es:

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

Pregunta 40

Parcialmente correcta

Se puntúa 0,80 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 parcialmente correcta.

Ha seleccionado correctamente 4.

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.