EXAMEN TEÓRICO PROGRAMACIÓN 1ªPARTE

1.- Son lo mismo una clase y un objeto en java? Justifica tu respuesta.

No, no son lo mismo. Una clase es una plantilla o molde que define el comportamiento y estado de los objetos que se crearán a partir de ella. Un objeto, en cambio, es una instancia de una clase, es decir, una realización concreta de esa plantilla en la memoria del programa.

2.- Diferencias y semejanzas entre una clase abstracta y una interfaz.

Diferencias:

- Una clase abstracta puede contener métodos concretos (implementados), mientras que en una interfaz, todos los métodos son abstractos por defecto.

- Una clase abstracta puede tener atributos con distintos modificadores de acceso, mientras que en una interfaz, todos los atributos son public static final

- Una clase puede heredar de una única clase abstracta, mientras que puede implementar múltiples interfaces

Semejanzas:

- Ambas sirven para definir un comportamiento mínimo que deben seguir otras clases.

- No se pueden instanciar directamente.

- Se utilizan para implementar polimorfismo.

3.- Tiene sentido que una clase final declare algún método que sea abstracto? Justifica tu respuesta

No, no tiene sentido. Una clase final no puede ser heredada, mientras que un método abstracto requiere ser sobrescrito en una subclase. Dado que una clase final no puede tener subclases, no se podría proporcionar una implementación para el método abstracto, lo que haría que la declaración fuera incoherente.

4.- ¿Qué es el polimorfismo?¿Como se implementa?

Es la capacidad de un mismo método para comportarse de manera diferente según el objeto que lo invoque. Se implementa mediante herencia y sobreescritura de métodos. Se puede lograr declarando métodos en una superclase y

sobrescribiéndolos en sus subclases. También se utiliza con interfaces y clases abstractas para definir métodos genéricos que las subclases implementarán de manera específica.

5.- Es posible compilar una clase que implementa una interfaz y no proporciona implementación para alguno de sus métodos?
Justifica la respuesta.

Sí, pero solo si la clase es declarada como abstracta. En Java, una clase que implementa una interfaz está obligada a proporcionar una implementación para todos sus métodos. Si no lo hace, la clase debe ser declarada como abstract y las clases derivadas serán responsables de implementar los métodos restantes.

6.- ¿En la redefinición de un método podemos cambiar el tipo de sus parámetros? Justifica la respuesta.

No, en la redefinición de un método (overriding), los parámetros deben mantenerse iguales en número y tipo. Si se cambian, no sería una sobrescritura sino una sobrecarga (overloading), que permite definir métodos con el mismo nombre pero con diferentes parámetros.

7.- Explica que es el "estado" y el "comportamiento" de un objeto

- Estado: Es la información que un objeto almacena en sus atributos en un momento dado.

- Comportamiento: Son las acciones o métodos que el objeto puede realizar. Ambos elementos definen el ciclo de vida del objeto en un programa.

8.- Explica en 2 líneas el principio de "ocultación de la información de la programación orientada a objetos.

Se refiere a la práctica de restringir el acceso directo a los detalles internos de un objeto, exponiendo solo lo necesario a través de una interfaz pública. Esto se logra mediante modificadores de acceso (private, protected, public) para mejorar la seguridad y modularidad del código.

9.- Diferencia entre ligadura estática y ligadura dinámica.

-Ligadura estática: La asociación entre una llamada a un método y su implementación se resuelve en tiempo de compilación. Se usa en métodos final, static o privados.

-Ligadura dinámica: La asociación se resuelve en tiempo de ejecución, lo que permite el **polimorfismo** y la invocación de métodos sobrescritos en clases derivadas.

10.- Crees que en una clase lo más recomendable es definir todos sus métodos como estáticos? Justifica tu respuesta.

No. Los métodos estáticos pertenecen a la clase en sí y no a una instancia en particular, lo que impide aprovechar características clave de la POO, como la herencia y el polimorfismo. Además, no pueden acceder directamente a atributos de instancia, lo que reduce su flexibilidad en el diseño orientado a objetos.