Clases Abstractas e Interfaces

Preguntas:

1. ¿Qué ventajas, en cuanto a polimorfismo, brindan las interfaces?

Las interfaces permiten que diferentes clases no relacionadas entre sí implementen un conjunto común de métodos, lo que facilita el polimorfismo. Esto permite que se pueda trabajar con objetos de diferentes clases a través de una referencia común (la interfaz), promoviendo la flexibilidad y la extensibilidad del código.

2. ¿Por qué no siempre puedo utilizar clases abstractas para agrupar clases polimórficas?

Las clases abstractas no permiten herencia múltiple en Java. Si una clase ya extiende otra clase, no puede extender una clase abstracta adicional. En cambio, las interfaces pueden ser implementadas por cualquier clase, independientemente de su jerarquía de herencia, además una clase si puede implementar varias interfaces a la vez, ya que no contienen implementación de estado, lo que evita los conflictos que podrían surgir con la herencia múltiple de clases.

3. ¿Qué ventajas tienen las clases abstractas sobre las interfaces?

Las clases abstractas pueden contener tanto métodos abstractos como métodos concretos (con implementación), así como campos de instancia. Esto permite compartir código común entre las subclases. Además, las clases abstractas pueden tener constructores y definir estados compartidos, lo que no es posible en las interfaces.

4. ¿Se puede instanciar una interfaz?

No, no se puede instanciar una interfaz directamente porque no tiene implementación de métodos. Solo se puede implementar en una clase concreta.

5. ¿Por qué no es recomendable incrementar o modificar las firmas definidas en una interfaz?

Porque cualquier cambio en una interfaz afecta a todas las clases que la implementan. Esto puede romper el código existente y requerir modificaciones en todas las implementaciones, lo que puede ser costoso y propenso a errores.

6. ¿Por qué, en lenguajes como Smalltalk, no es necesaria la implementación de interfaces?

En Smalltalk, el sistema de tipos es dinámico y se basa en el concepto de "duck typing". Esto significa que no es necesario declarar explícitamente una interfaz, ya que lo importante es que un objeto implemente los métodos esperados en tiempo de ejecución. Esto elimina la necesidad de interfaces formales como en Java.

Interfaces y Colecciones

Collection:

- Es una interfaz raíz en el marco de colecciones de Java.
- Define métodos generales para trabajar con grupos de objetos, como agregar, eliminar y verificar si contiene elementos.
- No proporciona una implementación concreta, solo especifica el comportamiento general.

List:

- Es una subinterfaz de Collection.
- Representa una colección ordenada (secuencial) de elementos, donde se permite el acceso por índice.
- Permite elementos duplicados.
- Define métodos adicionales como get(int index), set(int index, E element), add(int index, E element) y remove(int index).

ArrayList:

- Es una implementación concreta de la interfaz List basada en un array dinámico.
- Ofrece acceso rápido a elementos por índice (O(1)).
- Es más eficiente para operaciones de lectura, pero menos eficiente para inserciones o eliminaciones en el medio de la lista, ya que requiere desplazamiento de elementos (O(n)).

LinkedList:

- Es otra implementación concreta de la interfaz List, pero basada en una lista doblemente enlazada.
- Es más eficiente para inserciones y eliminaciones en el medio de la lista (O(1) si se tiene la referencia al nodo).
- Es menos eficiente para acceso por índice, ya que requiere recorrer la lista desde el inicio o el final (O(n)).

En resumen:
Usa ArrayList cuando necesites acceso rápido por índice y no realices muchas inserciones/eliminaciones en el medio.
Usa LinkedList cuando realices muchas inserciones/eliminaciones en el medio y no necesites acceso rápido por índice.

En todos los tipos están definidos los mismos métodos con los mismos nombres, pero implementados se formas distintas, son polimórficos, lo que nos permite cambiar de tipo de colección y aun así seguir usando los mismos métodos. Solo hay dos métodos que no están definidos en la interfaz de Collection: get() y subList().