Tp5: Interfaces

Luego, debata y responda las siguientes preguntas:

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

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

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

4. ¿Se puede instanciar una interface?

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

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

1. Las ventajas que brindan las interfaces respecto a polimorfismo es que se pueden agrupar varias clases que sean polimórficas solo en algún/nos métodos en particular que interesan para el problema.

2.No siempre se pueden usar clases abstractas porque hay clases que no comparten casi nada en común salvo algún método en particular, entonces al no tener sentido poner las clases en la misma jerarquía se utiliza la interface.

3. Las ventajas que tiene una clase abstracta son: que si vamos a trabajar con clases similares en estructura y comportamiento podemos extender fácilmente el sistema ya que es simple crear nuevas clases porque los atributos y métodos en común ya están definidos en la clase abstracta.

En cambio, la interfaz lo único que hace es obligar a que las clases que la implementar tengan un cierto comportamiento en común no toda su estructura.

4.No se puede instanciar una interface.

5.No es recomendable modificar o incrementar las firmas de una interface porque dejaría de compilar todas las clases que la implementan y debería implementar esos métodos, en caso de que sean muchas clases las que la implementan seria engorroso.

6. En lenguajes como smalltalk no son necesarias debido a que como son lenguajes dinámicamente tipados , no hace falta definir una interfaz ya que no necesitamos delimitar un tipo que cumpla con un protocolo en específico, por lo que en caso de la clase no conocer el método buscado se deberá implementar, pero no es una restricción obligatoria como lo es en los lenguajes estáticamente tipados.

Interfaces, Colecciones y otras yerbas

Como notará, los tipos de las colecciones no están definidos. ¿Qué ocurriría si utiliza en lugar de XXX, YYY, WWW y ZZZ las clases e interfaces recientemente estudiadas?

Si utilizamos las interfaces y clases recientemente estudiadas funcionaria la clase, pero hay algunos métodos que solo funcionarían con lists y otros que si funcionarían con el total de colecciones , como es el caso de get que necesita un índice para obtener el elemento, o el método getSubColecction que también se necesitan índices por parámetro y solo las list entienden ese mensaje.

De Smalltalk a Java: Personas y Mascotas

1. Defina la clase Persona y modélela en Smalltalk/Wollok. Una persona tiene un nombre y una fecha de nacimiento, por lo que debe ser posible pedirle su nombre, fecha de nacimiento, edad y debe ser posible compararla en edad con otra persona mediante el mensaje “<”.

2. Defina la clase Mascota y modélela en Smalltalk/Wollok. Una mascota tiene un nombre y una raza (String), por lo que debe ser posible pedirle tales datos.

3. Instancie dos Personas, dos Mascotas e inserte los cuatro objetos en una colección. Itere sobre la colección e imprima un listado con los nombres de todos los objetos de la misma.

4. Durante la iteración, ¿fue necesario distinguir si el objeto era una Persona o una Mascota para poder imprimir su nombre? ¿Cómo se llama el mecanismo de abstracción que permite esto?

5. ¿Podría asegurar que tanto mascotas como personas respondan al mensaje getName()mediante herencia? ¿Cómo? ¿Qué ocurre si Persona es subclase de otra clase que no puede ser modificada por ud.? ¿De qué otra manera puede asegurar el polimorfismo, teniendo en cuenta que está implementando en Java?

6. Implemente los puntos 1, 2 y 3 según las consideraciones de 4 y 5.

4.no fue necesario distinguir si el objeto era persona o mascota , el mecanismo de abstracción que permite esto es el poliformismo, es decir que muchas clases pueden conocer el mismo mensaje al pertenecer a la misma jerarquía.

5. se podría asegurar que las mascotas y las personas conozcan mediante herencia el mensaje getName(), haciendo formar parte de la misma jerarquía a persona y mascota, que sean un subclases de una superclase que conozca ese método. Lo que sucedería si no se puede modificar la superclase de persona es que no podría asegurar el polimorfismo , a menos que como en java cuente con herramientas como las interfaces, que puedo hacer que la clase persona y mascota implementen una interface que tenga un método abstracto llamada getName()