1°Desde el punto de vista del diseño de software, describa que propósito tiene implementar FIFO y LIFO en Estructuras de datos.

El propósito de implementar FIFO Y LIFO es la de implementar una conducta en las estructuras de datos con el propósito de mejorar el manejo de los datos en situaciones particulares.

Un ejemplo básico de esto es el de la estructura Queue (cola) que posee la conducta de FIFO (first in, first out) que significa que quien entra primero en la cola sale primero de esta, con lo que se pueden hacer variados modelos donde el orden de llegada es una parte fundamental del problema como lo serian hospitales, supermercados, museo, cine, etc.

2°Explique ¿qué utilidad o beneficio se obtiene la definir estructuras de datos genéricas?

Los beneficios que trae el uso de estructuras de datos genéricas son múltiples, una de las que más destaca es el poder utilizar clases como parámetros, lo que facilita el almacenado de múltiples objetos instanciados. Otro beneficio que posee es que esta provee el “compile-time type safety” (seguro de tipado en tiempo de compilación) lo que permite “atrapar (realizar un catch)” de un tipado invalido.

3° Explique, ¿Cuál es el propósito de usar abstracciones en la programación orientada a objetos?

El propósito de usar abstracción en POO es el de poder extender los atributos o métodos que posea la clase abstracta a una clase hija, para que esta se pueda “especializar”, esto se puede ver desde el punto de vista de que la clase abstracta es una generalización de varias otras clases hijas que son diferentes entre sí pero poseen en su “núcleo” aspectos y funciones similares que se reúnen en la clase abstracta. Un ejemplo de esto es el de la venta de instrumentos en una tienda de música, los instrumento poseen sus propios set de atributos y métodos, pero no sólo existe un solo un tipo de instrumento, ya que existen los instrumentos de cuerda, viento, percusión, etc., que a su vez poseen otros atributos que los difieren entre sí, estos siguen poseyendo los atributos de cuando son calificados como una generalización pero se especializan en cosas distintas.

4°Explique, ¿Qué diferencia conceptual existe entre los conceptos Clase Abstracta e Interface?

Conceptualmente la clase abstracta y la interface se parecen bastante, pero la diferencias que las separa es a la hora de ver la función que cumplen cada una de ellas, la clase abstracta es una generalización (en termino de diccionario) de las clases a la que se extiende, reuniendo tanto atributos como métodos que puedan compartir. La Interface en cambio es una generalización de diversos métodos que se pueden repetir, estos “métodos abstractos” luego se pueden implementar en diversas clases incluyendo a la misma clase abstracta, se podría decir que la clase abstracta otorga un “esqueleto” para las clases a las que se extienden, mientras que la interface entrega un “set de instrucciones” para las clases que la implementan.

5° Explique, ¿De qué manera es posible instanciar una Interface?

Una interface no se puede instanciar, solo se puede implementar a una clase que luego se puede instanciar en un objeto obteniendo así los métodos que posea la interface.

6°Explique, ¿A qué refiere el concepto “Diseño de Software”?

Diseño de software se refiere a un proceso donde se toman las decisiones de cómo se va a realizar un programa para que este alcance el propósito específico que pidió el stakeholder, esto haciéndose a través de espacio de diseño donde se ven las decisiones en sí de cómo se va a realizar el programa tomando en cuenta los atributos, clases, interfaces, etc., a la hora de desarrollar el software para ver si el software se puede mejorar además o refactorizar, etc.

7° Explique, ¿A qué refiere el concepto “Refactorización”?

La refactorización se puede referir como el arreglo, mejoría o agregación de nuevas funciones al código original, para que esté funcione de mejor manera.

8° Explique, ¿a qué refiere los conceptos “Patrón y Antipatrón de Diseño”?

Los conceptos de Patrón y Antipatrón se refieren a técnicas de diseños que son usados a la hora de diseñar software en programación, los patrones son técnicas buenas de diseños que son recomendadas de usar y los antipatrones son los opuesto, técnica de diseño malas que entorpecen a esta. Ejemplos de antipatrones serían: código duplicado, la clase “dios”, entre varios.

9° Explique, ¿Qué elementos de diseño de software justifican el interés por implementar “Java Collection Framework”?

10° Describa, ¿Cuáles son los elementos centrales que conforman “Java Collection Framework”?

Los elementos centrales que conforman “Java Collection Framework” son List, Queue y Set, los cuales son interfaces que permiten almacenar datos cada uno con diferentes características, List por ejemplo puede guardar datos además de ordenarlos con collections, Queue solo puede guardar datos siguiendo FIFO, y Set no puede tener elementos repetidos.

11° Explique conceptualmente, ¿A qué refiere Iterator?

Iterator es una interface disponible en la “Java Collection Framework” la cual permite recorrer cada elemento en las colecciones que se tiene como List, Queue, etc.

12°Explique, ¿Qué rol y propósito tiene la utilización de un hashcode en una estructura de datos?

El propósito de la utilización de un hashcode es para determinar el índice de cada elemento que se posea en la estructura.

13° Explique y describa, ¿qué mecanismo(s) permite resolver posibles colisiones en la agregación de datos en un HashSet?

El elemento que permite solucionar las colisiones es llamado Separate Chaining, que permite que nuevos elementos sean almacenados en el mismo índice con un arreglo o una lista enlazada.

14° Explique, ¿Qué operaciones tienen las siguientes estructuras de datos en común: Set, List, Queue y Map?

Las operaciones que tienen en común: agregar un elemento, remover un elemento, obtener un elemento en cierto índice, ver si existe un elemento, obtener los elementos en cierto orden, obtener los elementos en el orden que fueron agregados.

15° Explique, ¿Qué tipo de información provee la notación Big-O para el diseño de software?

La notación Big-O nos entrega una medida de cómo rinden diversas estructuras de datos o algoritmos de ordenamiento, estas medidas muestran el mejor caso y el peor caso.

Explique, ¿De qué manera la incorrecta selección de una estructura de datos podría impactar el desarrollo de software?

La incorrecta selección de una estructura de datos podría dificultar: la obtención de datos deseados, el ordenamiento de datos, que el propio software no haga lo que se tenía deseado por que no sigue una conducta (como ejemplo FIFO, LIFO), etc.

17° En el marco de la utilización de los principios de ocultamiento de información, explique ¿de qué manera es posible implementar estos principios cuando se desarrolla un programa que utiliza el lenguaje de programación Java?

El encapsulamiento se implementa a la hora de elegir si una clase, variable o método es pública, privada o protegida, esto dependiendo de si uno quiere llamar por ejemplo de una variable de una clase a la otra cuando esta se instancia, pero en general uno siempre desea que estas estén protegidas para que un tercero no puede hacer de las suyas y manipularla.

18° Explique, ¿A qué refiere el concepto “Primitive Obsession”?

El concepto de “Primitive Obsession” se refiere a la obsesión que se posee de usar variables de tipo primitivo este es denominada como un antipatrón ya que no es recomendable usar variables primitivas.