Q=how to provide priority in executer framework?

Q=dieference between data hiding and abstraction?

Ans=Data hiding is the process by which access modifiers are used to hide the visibility of java methods and variables. They access modifiers are: public, private and protected.

Abstraction is the process by which we define a specific behavior by beans of abstract classes and methods which form the skeleton for any class that would be extending this class

**Encapsulation** in Java is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class. Therefore, it is also known as **data hiding**.

To achieve encapsulation in Java −

* Declare the variables of a class as private.
* Provide public setter and getter methods to modify and view the variables values.

Benefits of Encapsulation

* The fields of a class can be made read-only or write-only.
* A class can have total control over what is stored in its fields.

**abstraction** is a process of hiding the implementation details from the user, only the functionality will be provided to the user. In other words, the user will have the information on what the object does instead of how it does it.

In Java, abstraction is achieved using Abstract classes and interfaces.

## Abstract Class

A class which contains the **abstract** keyword in its declaration is known as abstract class.

* Abstract classes may or may not contain *abstract methods*, i.e., methods without body ( public void get(); )
* But, if a class has at least one abstract method, then the class **must**be declared abstract.
* If a class is declared abstract, it cannot be instantiated.
* To use an abstract class, you have to inherit it from another class, provide implementations to the abstract methods in it.
* If you inherit an abstract class, you have to provide implementations to all the abstract methods in it.

Q=query for find duplicate record in table?

Ans=SELECT

name, email, COUNT(\*)

FROM

users

GROUP BY

name, email

HAVING

COUNT(\*) > 1

Q=**Difference between WHERE and HAVING clause:**

**Ans:**

|  |  |
| --- | --- |
| down vote | **Difference between WHERE and HAVING clause:**  **1.** ***WHERE clause*** can be used with - Select, Insert, and Update statements, where as ***HAVING clause*** can only be used with the Select statement.  **2.** ***WHERE*** filters rows before aggregation (GROUPING), where as, ***HAVING*** filters groups, after the aggregations are performed.  **3.** Aggregate functions cannot be used in the ***WHERE clause***, unless it is in a sub query contained in a **HAVING clause**, whereas, aggregate functions can be used in Having clause.  **Filtering Groups:**  WHERE clause is used to filter rows before aggregation, where as HAVING clause is used to filter groups after aggregations  Select City, SUM(Salary) as TotalSalary  from tblEmployee  Where Gender = 'Male'  group by City  Having City = 'London'  In SQL Server we have got lot of aggregate functions. **Examples**   1. Count() 2. Sum() 3. avg() 4. Min() 5. Max() |

Q=get() method of e=executer framework is blocking how u use nonblocking?

Ans= we will use CompletableFuture class – introduced as a Java 8 Concurrency API

Q=how many thread pool we should create for n number of processors?

Ans= ExecutorService e = Executors.newFixedThreadPool(Runtime.getRuntime().availableProcessors());

// Do work using something like either

e.execute(new Runnable() {

public void run() {

// do one task

}

});

Q=when we will use int and when interger in java?

Ans= We sue wrapper classes in class lavel variable, because when class loads its default value will be null, so it will not consume any memory, but if we will use primitive type it will store in memory and its default value will load, and in method level variable we should use int primitive type.

Q= difference between fork join pool and executer service?

Ans=