**covariant type in java ?**

Covariant return type refers to return type of an overriding method. ... Covariant return type works only for non-primitive return types. From Java 5 onwards, we can override a method by changing its return type only by abiding the condition that return type is a subclass of that of overridden method return type

**immutal class?**

Class and data member must be final, A parameterized constructor, Getter method for all the variables in it, No setter.

Volatile Keyword in Java

Volatile keyword is used to modify the value of a variable by different threads. It is also used to make classes thread safe. It means that multiple threads can use a method and instance of the classes at the same time without any problem. The volatile keyword can be used either with primitive type or objects.

The volatile keyword does not cache the value of the variable and always read the variable from the main memory. The volatile keyword cannot be used with classes or methods. However, it is used with variables. It also guarantees visibility and ordering. It prevents the compiler from the reordering of code.

**How to create Mutual class?**

**Serializable vs deSerializable ?**

That Serializable is a marker interface means that it contains no methods. Therefore, a class implementing Serializable does not have to implement any specific methods. Implementing Serializable thus just tells the Java serialization classes that this class is intended for object serialization.

Serialization is a mechanism of converting the state of an object into a byte stream. Deserialization is the reverse process where the byte stream is used to recreate the actual Java object in memory. This mechanism is used to persist the object.

**flexiblity,security,abstraction,reuseability ==> Piller of Opps.**

**composition( Has-A relationship) vs aggregation(IS-a) ?**

Aggregation implies a relationship where the child can exist independently of the parent. Example: Class (parent) and Student (child). Delete the Class and the Students still exist.

Composition implies a relationship where the child cannot exist independent of the parent. Example: House (parent) and Room (child). Rooms don't exist separate to a House.

**method hiding ?**

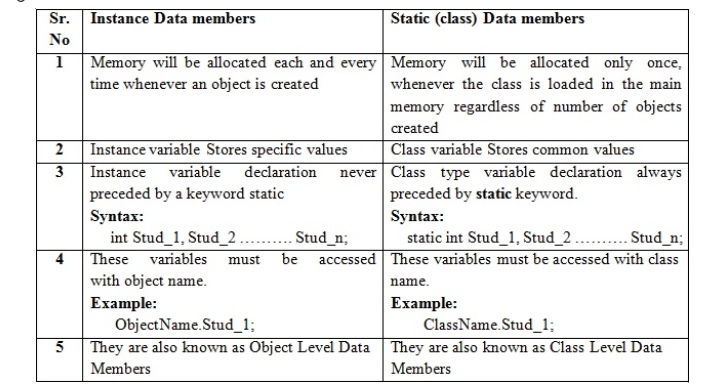
When super class and sub class contains same method including parameters and if they are static.

The method in the super class will be hidden by the one that is in the sub class. This mechanism is known as method hiding.

**automatically --exact method signnature?**

**this()-> contructor chaining .**

**instance vs static block difference?**



iterator?

**hashmap vs hashtable?**

1) HashMap is non synchronized. It is not-thread safe and can't be shared between many threads without proper synchronization code.

2) HashMap allows one null key and multiple null values.

4) HashMap is fast.

5) We can make the HashMap as synchronized by calling this code

Map m = Collections.synchronizedMap(hashMap);

6) HashMap is traversed by Iterator.

1) Hashtable is synchronized. It is thread-safe and can be shared with many threads.

2) Hashtable doesn't allow any null key or value.

## What Is Cloning?

**Cloning** is a process of creating an exact copy of an existing object in the memory. In java, **clone()** method of **java.lang.Object** class is used for cloning process. This method creates an exact copy of an object on which it is called through **field-by-field assignment** and returns the reference of that object. Not all the objects in java are eligible for cloning process. The objects which implement **Cloneable interface** are only eligible for cloning process. Cloneable interface is a [marker interface](https://javaconceptoftheday.com/marker-interface-java/) which is used to provide the marker to cloning process.

**Shallow cloning vs Deep cloning :**

**Shallow cloning**

**The process of creating bitwise copy of an object is called Shallow Cloning .**

**2. If the main object contain any primitive variables then exactly duplicate copies**

**will be created in cloned object.**

**3. If the main object contain any reference variable then the corresponding object**

**won't be created just reference variable will be created by pointing to old**

**contained object.**

**4. By using main object reference if we perform any change to the contained object**

**then those changes will be reflected automatically to the cloned object , by**

**default Object class clone( ) meant for Shallow Cloning.**

**5.Shallow cloning is the best choice , if the Object contains only primitive values.**

**Deep cloning**

**1.The process of creating exactly independent duplicate object(including contained**

**objects also) is called deep cloning.**

**2. In Deep cloning , if main object contain any reference variable then the**

**corresponding Object copy will also be created in cloned object.**

**3. Object class clone( ) method meant for Shallow Cloning , if we want Deep**

**cloning then the programmer is responsible to implement by overriding clone( )**

**method.**

**In Deep cloning by using main Object reference if we perform any change to the**

**contained Object those changes won't be reflected to the cloned object.**

**Lazy Copy**:  
A lazy copy can be defined as a combination of both shallow copy and deep copy. The mechanism follows a simple approach – at the initial state, shallow copy approach is used. A counter is also used to keep a track on how many objects share the data. When the program wants to modify the original object, it checks whether the object is shared or not. If the object is shared, then the deep copy mechanism is initiated.

constructor?

default size of the array ? 10 object

Spring boot version ? 2.2.1

number of ways to generate the object of class?

================================================================

Using the new keyword

Java Class.newInstance() method

Using newInstance() method of Class class

Using clone() method

Using deserialization

======================================================

**Why java won't provide support for multiple inheritance? :**

**java**, every class is a child of Object class. When it **inherits** from more than one super class, sub class gets the ambiguity to acquire the property of Object class

**method signature-> method name+method parameter**

**Why ambiguity problem won't be there in interfaces? 🡺 Because there is no body of method**

**why Cyclic inheritance is not allowed in java? -> ambiguity**

**Automatic promotion in overloading.?**

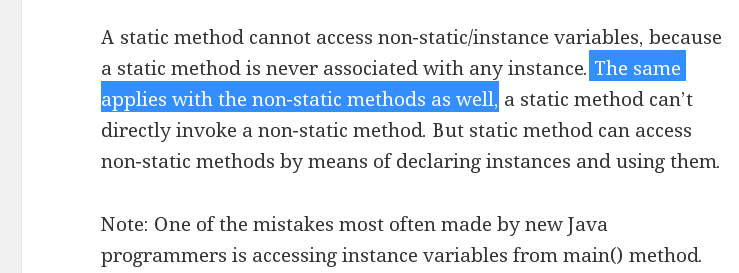
While resolving overloaded methods if the exact match method is not available, you will not immediately get any compile time error. First, the compiler promotes lower type argument to the higher type argument and checks whether the match method is available or not. If the match method is available, it will be considered.

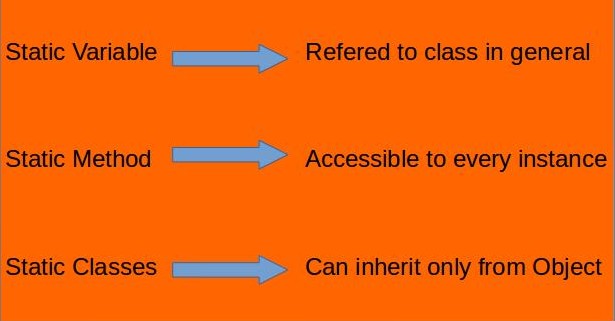
If the match method is not available, compiler promotes argument once again to the higher type. This process will be continued until all possible promotions are not checked. Still, if the match method is not available, we will get compile time error. This process is called automatic type promotion in [method overloading](https://www.scientecheasy.com/2019/02/method-overloading-in-java.html/).

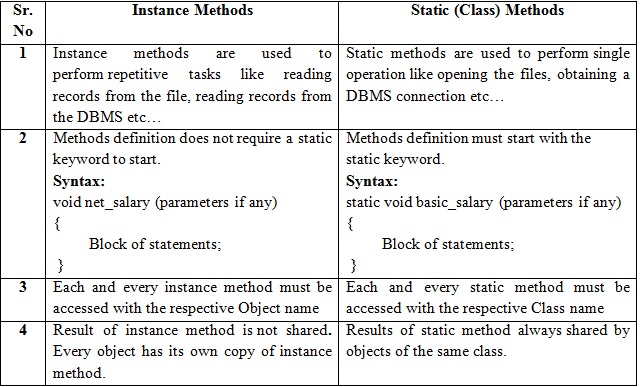
**why abstract class have constructor ? : yes so that we can create object of abstract class.**

**Can we create object of the Abstract Class? yes**

**Static variable,class,method vs InstanceBlock,variable,class,method**







# Singleton Class in Java

In object-oriented programming, a singleton class is a class that can have only one object (an instance of the class) at a time.  
After first time, if we try to instantiate the Singleton class, the new variable also points to the first instance created. So whatever modifications we do to any variable inside the class through any instance, it affects the variable of the single instance created and is visible if we access that variable through any variable of that class type defined.  
To design a singleton class:

1. Make constructor as private.
2. Write a static method that has return type object of this singleton class. Here, the concept of [Lazy initialization](https://en.wikipedia.org/wiki/Lazy_initialization) is used to write this static method.

## **HTTP Methods**

## **The GET Method**

**GET is used to request data from a specified resource.**

**GET is one of the most common HTTP methods.**

Note that the query string (name/value pairs) is sent in the URL of a GET request:

/test/demo\_form.php?name1=value1&name2=value2

* GET requests can be cached
* GET requests remain in the browser history
* GET requests can be bookmarked
* GET requests should never be used when dealing with sensitive data
* GET requests have length restrictions
* GET requests are only used to request data (not modify)

## **The POST Method**

**POST is used to send data to a server to create/update a resource.**

The data sent to the server with POST is stored in the request body of the HTTP request:

POST /test/demo\_form.php HTTP/1.1  
Host: w3schools.com  
name1=value1&name2=value2

**POST is one of the most common HTTP methods.**

**Some other notes on POST requests:**

* POST requests are never cached
* POST requests do not remain in the browser history
* POST requests cannot be bookmarked
* POST requests have no restrictions on data length

## **The PUT Method**

**PUT is used to send data to a server to create/update a resource.**

The difference between POST and PUT is that PUT requests are idempotent. That is, calling the same PUT request multiple times will always produce the same result. In contrast, calling a POST request repeatedly have side effects of creating the same resource multiple times.

Difference between PUT and Patch

Also, another difference is that when you want to update a resource with PUT request, you have to send the full payload as the request whereas with PATCH, you only send the parameters which you want to update.

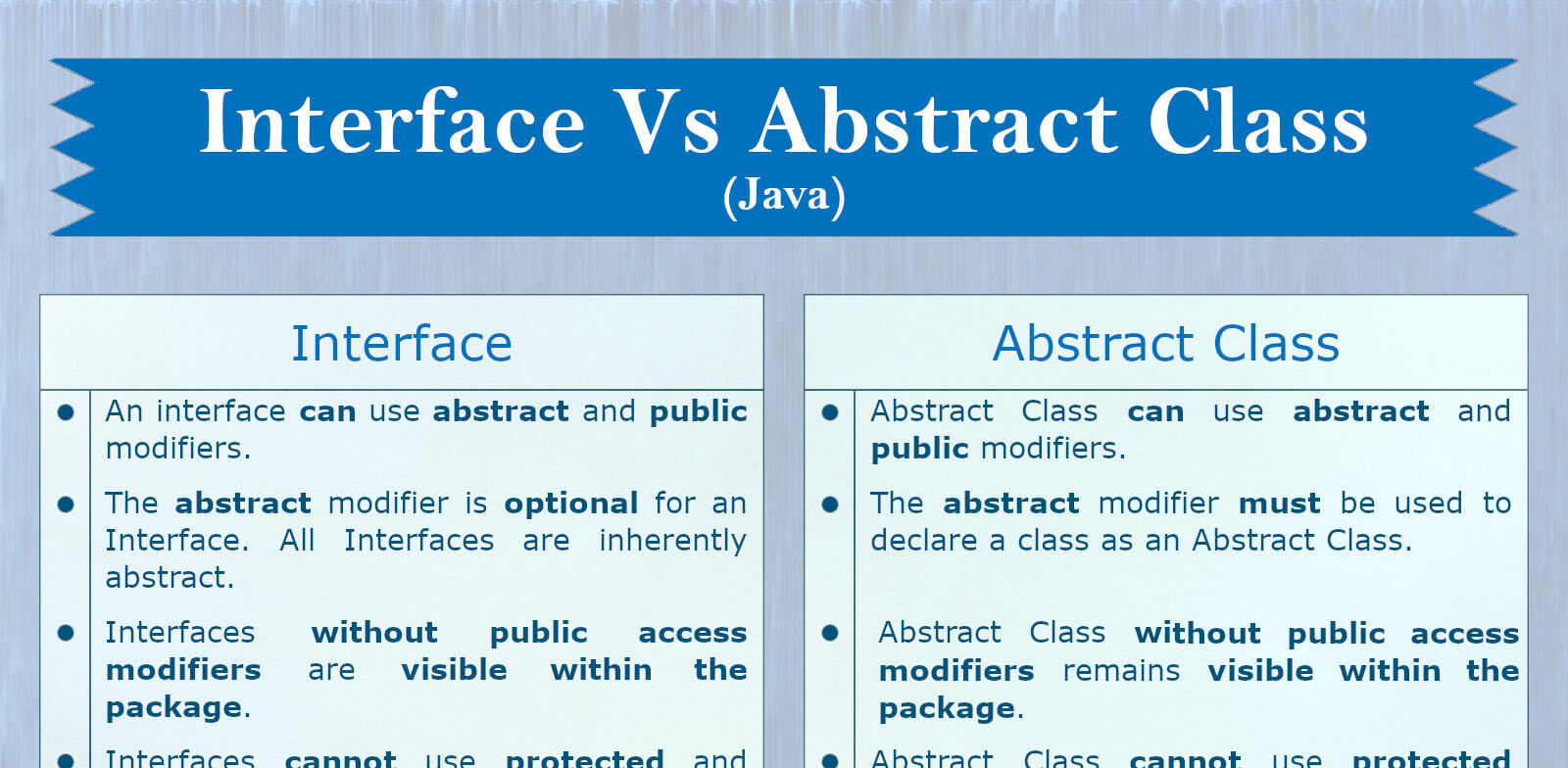
**difference between sleep and yield in java**

Yield method temporarily pauses the currently executing thread to give a chance to the remaining waiting threads of the same priority to execute.  
If there is no waiting thread or all the waiting threads of low priority then the current thread will continue its execution

WhereAs

Sleep  method causes the currently executing thread to sleep for the number of milliseconds or the  nanoseconds specified in the argument.

java 8 interface vs abstract class?



# Try with Resources

In Java, the try-with-resources statement is a try statement that declares one or more resources. The resource is as an object that must be closed after finishing the program. The try-with-resources statement ensures that each resource is closed at the end of the statement execution.

You can pass any object that implements java.lang.AutoCloseable, which includes all objects which implement java.io.Closeable.

The following example writes a string into a file. It uses an instance of FileOutputStream to write data into the file. FileOutputStream is a resource that must be closed after the program is finished with it. So, in this example, closing of resource is done by itself try.

#### **Note - In a try-with-resources statement, catch or finally block executes after closing of the declared resources.**

# Java Marker Interface: It is an empty interface (no field or methods)

### Marker interface is used as a tag to inform a message to the Java compiler so that it can add special behaviour to the class implementing it. Java marker interface has no members in it.

### Java Marker Interface Examples

* java.lang.Cloneable
* java.io.Serializable
* java.util.EventListener

# We cannot create marker interfaces, as you cannot instruct JVM to add special behavior to all classes implementing (directly) that special interface.

# Internal Working of HashMap in Java : it use hashset.

# @componet :

# Hashmap

# Java 8 Stream Intermediate And Terminal Operations