# CORE JAVA INTERVIEW QUESTIONS

1. Explain the **main** method of the class ? ( psvm() )

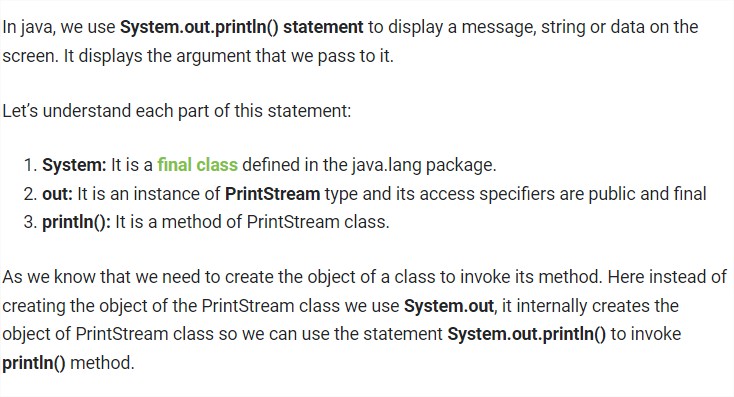
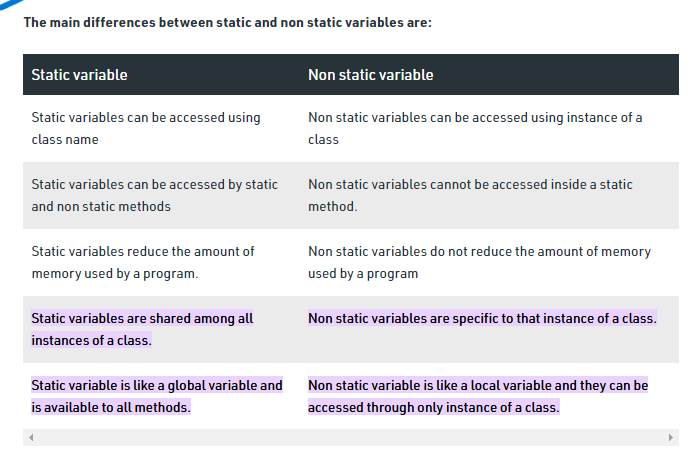
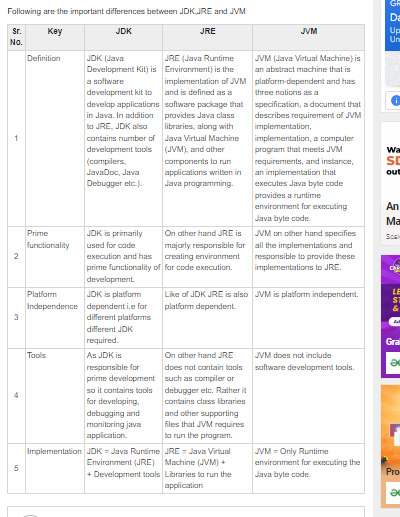
**ans:** **When the Java interpreter executes an application (by being invoked upon the application's controlling class), it starts by calling the class's main method. The main method then calls all the other methods required to run your application.**

1. The main method in Java is public so that it's visible to every other class, even which are not part of its package. if it's not public JVM classes might not able to access it.

2. The main method is static in Java so that it can be called without creating any instance. While JVM tries to execute the Java programs it doesn't know how to create instances of the main class as there is no standard constructor is defined for the main class.

3. The main method is void in Java because it doesn't return anything to the caller which is JVM.

4. String[ ] args it is used for treating all the arguments as array of string class.

1. Explain System.out.println()?; **ans: **
2. Difference between Static , Non – static , Local variables ?**ans:**
3. Difference between JDK and JRE ? 
4. What is **var** type ?

It was introduced in version 10 of the java.if a variable is declared as var it means we can store any kind of value in it.var type allocates the data type to the variable dynamically depending on the value we are storing in it.a variable with var type can not be static and non static.it can only be local variable.method argument can not be of the type var.var is not a keyword in java so we can use var as a variable name.

1. Explain about Constructors ?

A constructor should have same name as that of the class.whenever we create an object constructor will be called automatically.constructor should not be written in the main .constructor does not have any data type.it is permantely void.constructor will never get inherited.

1. Explain about Constructor overloading , Constructor chaining , Default Constructor ?

CONSTRUCTOR OVERLOADING:In constructor overloading we create more than one constructor in the same class with different no of argument /different types of argument .

CONSTRUCTOR CHAINING:When we call one constructor from another constructor then it forms a flow of chain like structure that chainig is know as constructor chaining.

DEFAULT CONSTRUCTOR:When we do not create a constructor then during compilation in dot class file automatically with no arguments,empty body constructors is created and this constructor is called as default constructors .

When an object with value is created then as a programmer explicitly we have to create matching constructor In the class.

1. What is **this** keyword ?

This is a special reference variable that automatically get created.this keyword points to the current object running in the program.using this keyword we can access non static variables.it can not access local variables.this can not be used in the static method so this cannot be run inside the main.using this() we can call constructors and this calling should be happen from another constructor and always should be a very first statement.

1. What is **super** keyword ?

Using super keyword we can access the members of the parent class .inheritance must for the super keyword.using super keyword we can access static and non static members both.super keyword can not be used inside static context/method.using super keyword we can call constructor of the parent class but then we should use super keyword in child class constructor and it should be very first statement.if we don’t keep super keyword inside child class constructor then compiler will automatically place the super keyword such that it can call only no argument constructor of parent class.super keyword is not automatically kept when there are only constructors with argument in parent class .we can not use this keyword and super keyword in the same constructor to call another constructor .as either of the stmnt becomes second stmnt then we will get the error.

10)Explain about OOPS concept ? (Inheritance , Polymorphism , Abstraction , Encapsulation )

INHERITANCE:Here we inherit the non-static members of parent class to child class with an intension of reusing it.static members do not get inheritant but the they give us the feel of inheritance by converting the statement.

JAVA DOESN’T SUPPORT MULTIPLE INHERITANCE AT CLASS LEVEL IT SUPPORT MULTIPLE INHERITANCE AT INTERFACE LEVEL.

POLIMORPHISM:polymorphism is one of the pillor of oops concept .

* Polymorphism in java is a concept by which we can perform a sigle action in different ways so polymorphism means many forms (developing a feature such that it can take more than one form)

There are 2 types of polymorphism

1.Overriding 2.overloading

* OVERRIDING:here we inherit a method and the modify the logic of inherited method in the child class by once again creating the method in the child class with some signature.
* OVERLOADING:here we create more than one method with the same name in the same class with different no of arguments or different types of argument.

ABSTRACTION:In abstraction we hide implementation details and this can be done in java by using interfaces and abstract class.

ENCAPSULATION :Encapsulation refers to bundling of the data with the methods that operate on the data restricting direct access to the variable.here we make the variable private to restrict its direct access.publicly define “getters”and “setters” methods are created to access these variables indirectly.

11)Explain about Type casting and Class casting ?

12)**What is run - time polymorphism ?( both overrideing as well class casting )ans:** Run-Time Polymorphism: **Whenever an object is bound with the functionality at run time**, this is known as runtime polymorphism. The runtime polymorphism can be achieved by method overriding. Java virtual machine determines the proper method to call at the runtime, not at the compile time.

1. Explain about Interfaces ?

Interface consist of incomplete methods.it help us to achieve good designing of the program.in java interfaces support multiple inheritance.a class can implements more than one interface.we can use extends and implements together but extends is used first and then implements.every variable in the interface by default is static and final.an object for an interface cannot be created but the reference can be created.if we want to define the incomplete methods it is not mandatory to use abstract keyword. Interfaces can also contain complete method using default keyword

1. Explain all access specifiers ? ( public , private , protected , default )

Private:if we make a **variable/method** private then it can be accessed only in same class.if u make **constructor** private then its object can be created in same class only but not in different class,different package.

Default: if we make a **variable/method** default then it can be accessed only in same package but not in different package.if u make **constructor** default then its object can be created in same classs and in same package but not in different package.if we make a **class** .

default it can be access only in the same package.

Protected: if we make a **variable/method** protected then it can be accessed in same package and in different package only through inheritance.if u make **constructor** protected then its object can be created in same classs and in same package but not in different package.

Private: : if we make a **variable/method** private then it can be accessed in same package and in different packag.if u make a class as public then it can be access in same package and in different package as well.if u make **constructor** private then its object can be created in same classs and in same package and in different package as well.

1. What is final keyword ?

1.if we make a variable final then changing of the value is not allowed.

2.if we make a method as final then overriding of the method is not allowed.

3.if we make non-static/static final then initialization of the variable is mandatory.

4.if we make a class as final then inheritance of the class is not allowed.

14)Explain new features of Java 8 ?( default , functional interfaces , lambda expressions , stream API ) ?

Default keyword:it was introduced in the version 8 of java.using default keyword we can create complete methods in an interface.

Functional interface:functional interface should consist of exactly one incomplete method/abstract method in it.functional interface can not have 0 abstract method.a functional interface should consist of exactly one incomplete method but can consist of any no of complete methods.

Lambda expressions:it reduces no of lines of the code during development.it is applicable only on functional interface.it is the faster way to implement but it decreases the readability.because of lambda expressions java is known as functionl proframming language

Non static members can be access without object creation by using lambda expressions which is functional thing.

Syntax for lambda expressions: A a1=()->{ };

Lambda Expressions were added in Java 8. A lambda expression is **a short block of code which takes in parameters and returns a value**. Lambda expressions are similar to methods, but they do not need a name and they can be implemented right in the body of a method

16)Explain about abstract class ?

An abstract class should consist of both complete and incomplete methods in it.to define incomplete methods in abstract class it is mandatory to use abstract keyword.we can create static and non-static variables in an abstract class.abstract class doesn’t support multiple inheritance.we can create main method in an abstract class.an object of the abstract class can not be created but reference of that variable of abstract class can be created.

If we declare a method in a class aabstract to use it we have to override this method in the subclass but overriding is not possible with static methods therefore an adstract class can not be static .

1. Difference between Interfaces and abstract class ?

|  |  |
| --- | --- |
| interface | Abstract class |
| 1.interface is 100% incomplete method | 1.abstract class can have 0-100% incomplete methods. |
| 2.support multiple inheritance | 2.doesnt support multiple inheritance. |
| 3.all variables by default final and static  4.to define incomplete methods it is not mandatory to use abstract keyword | 3.all variables can be static and non-static  4.to define incomplete methods in abstract class it is not mandatory to use abstract keyword. |
| 5.usage of the constructors is not allowed | 5.constructors can be used. |

1. Explain about Exceptions ?
2. **What is Exception propagation ?**

when an exception happens, Propagation is a process in which the exception is being dropped from to the top to the bottom of the stack. If not caught once, the exception again drops down to the previous method and so on until it gets caught or until it reach the very bottom of the call stack. This is called exception propagation and this happens in case of Unchecked Exceptions.

In the example below, exception occurs in m() method where it is not handled, so it is propagated to previous n() method where it is not handled, again it is propagated to p() method where exception is handled.  
Exception can be handled in any method in call stack either in main() method, p() method, n() method or m() method.

**Note :**By default, Unchecked Exceptions are forwarded in calling chain (propagated).

1. Explain about multi catch ?

ANS: **A try block can be followed by one or more catch blocks**. Each catch block must contain a different exception handler. So, if you have to perform different tasks at the occurrence of different exceptions, use java multi-catch block

1. Explain about Serialization and de-Serialization ?
2. What is Transient keyword ?
3. Explain about mutable and immutable ?
4. **Why String is immutable ?**

The String pool cannot be possible if String is not immutable in Java. A lot of heap space is saved by JRE. The same string variable can be referred to by more than one string variable in the pool. **String interning**(String Interning is a method of storing only one copy of each distinct String Value, which must be immutable.) can also not be possible if the String would not be immutable.

* If we don’t make the String immutable, it will pose a serious security threat to the application. For example**, database usernames, passwords are passed as strings to receive database connections. The socket programming host and port descriptions are also passed as strings.** The String is immutable, so its value cannot be changed. If the String doesn’t remain immutable, any hacker can cause a security issue in the application by changing the reference value

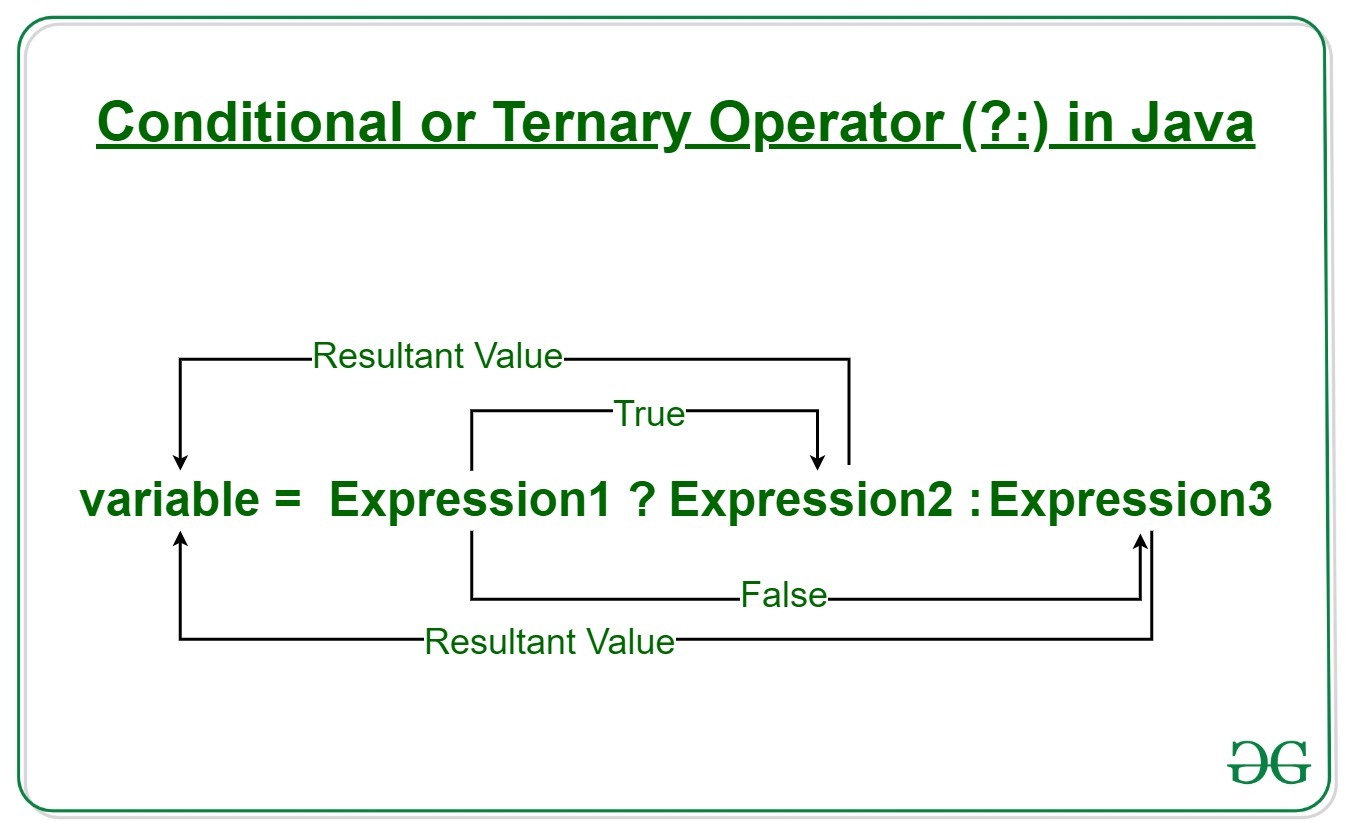
1. How to create immutable class ?
2. Explain about String constant pool ?
3. Difference between String , String Buffer , String Builder ?

* **Ans:** *If a string is going to remain constant throughout the program, then use the String class object because a String object is immutable.*
* *If a string can change (for example: lots of logic and operations in the construction of the string) and will only be accessed from a single thread, using a StringBuilder is good enough.*
* *If a string can change and will be accessed from multiple threads, use a StringBuffer because StringBuffer is synchronous, so you have thread-safety.*
* *If you don’t want thread-safety than you can also go with StringBuilder class as it is not synchronized.*

|  |  |
| --- | --- |
| **StringBuffer** | **StringBuilder** |
| 1) | StringBuffer is *synchronized* i.e. thread safe. It means two threads can't call the methods of StringBuffer simultaneously. | StringBuilder is *non-synchronized* i.e. not thread safe. It means two threads can call the methods of StringBuilder simultaneously. |
| 2) | StringBuffer is *less efficient* than StringBuilder. | StringBuilder is *more efficient* than StringBuffer. |
| 3) | StringBuffer was introduced in Java 1.0 | StringBuilder was introduced in Java 1.5 |

1. Difference between **==** and **.equals()** method ?ans: We can use == operators for reference comparison (address comparison) and . equals() method for content comparison. In simple words, **== checks if both objects point to the same memory location whereas .** **equals() evaluates to the comparison of values in the objects**.
2. Difference between continue and break statement?

28)What is ternary operator ?ans: Java ternary operator is the only conditional operator that takes three operands. It’s a one-liner replacement for the if-then-else statement and is used a lot in Java programming. We can use the ternary operator in place of if-else conditions or even switch conditions using nested ternary operators. Although it follows the same algorithm as of if-else statement, the conditional operator takes less space and helps to write the if-else statements in the shortest way possible.



**Syntax:**

variable = Expression1 ? Expression2: Expression3

|  |  |
| --- | --- |
| Array | vector |
| 1.it is not synchronized  2.The size of the array is established when the array is created.  3.where as array is a primitive data type  4.Elements in the array can not be deleted  5.they are statically allocated | 1.it is synchronized  2.As the Vector is growable, the size changes when it grows.  3.Vector implements the List interface  4.where as a Vector can.  5. they are dynamically allocated |

1. Difference between Array and vector ?ans:

29)Explain about increment and decrement operators ? ans: **Increment operator**is used to incrementing a value by 1. There are two varieties of increment operator:

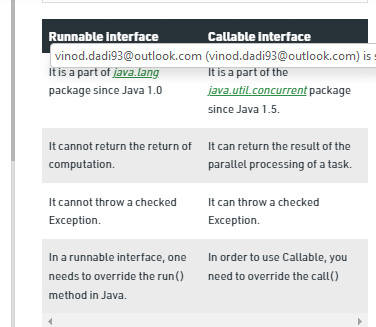
* **Post-Increment:**Value is first used for computing the result and then incremented.
* **Pre-Increment:** Value is incremented first and then the result is computed.

**Decrement operator**is used for decrementing the value by 1. There are two varieties of decrement operators.

* **Post-decrement:** Value is first used for computing the result and then decremented.
* **Pre-decrement:** Value is decremented first and then the result is computed

1. What are Threads ?

A thread is **the path followed when executing a program**. The Java Virtual Machine allows an application to have multiple threads of execution running concurrently. Every thread has a priority. Threads with higher priority are executed in preference to threads with lower priority.

1. How many ways we can create threads ? 2 ways
   1. By using thread class where it has start () and run () method
   2. By using runnable interface where it has run() method.
2. What are runnable interfaces and callable interfaces ?ans: 
3. What is Synchronization , advantages and disadvantages ?

Ans: Synchronization makes sure that shared resources or data can be accessed by only one thread at a time while execution. its advantage is that **it prevent data inconsistency and disadvantage is that it makes execution slower as it makes other thread wait till current thread completes execution**.

1. Explain Thread priority ?
2. Explain thread schedular ? (wait , notify , notify all , sleep )
3. Explain the Thread life cycle ?
4. What is Thread pool , join thread , detach ?

thread pool **A thread pool reuses previously created threads to execute current tasks and offers a solution to the problem of thread cycle overhead and resource thrashing.** Since the thread is already existing when the request arrives, the delay introduced by thread creation is eliminated, making the application more responsive

**Detach: Separates the thread of execution from the thread object, allowing execution to continue independently**. Any allocated resources will be freed once the thread exits.

1. What are Enum , Wrapper , **optional , local** , anonymous ?

Ans: An enum type is **a special data type that enables for a variable to be a set of predefined constants**. The variable must be equal to one of the values that have been predefined for it. Common examples include compass directions (values of NORTH, SOUTH, EAST, and WEST) and the days of the week.

 a **wrapper class** is a class that encapsulates types, so that those types can be used to create object instances and methods in another class that need those types

1. **What are inner classes and singleton classes** ?

**Ans**: **Java inner class** or nested class is **a class that is declared inside the class or interface**. We use inner classes to logically group classes and interfaces in one place to be more readable and maintainable. Additionally, it can access all the members of the outer class, including private data members and methods.

In Java, Singleton is **a design pattern that ensures that a class can only have one object**.

To create a singleton class, a class must implement the following properties:

* Create a private constructor of the class to restrict object creation outside of the class.
* Create a private attribute of the class type that refers to the single object.
* Create a public static method that allows us to create and access the object we created. Inside the method, we will create a condition that restricts us from creating more than one object.

1. Explain about final , finally , finalize ?
2. Difference between throws and throw keyword ?
3. What are regular expressions in java ?
4. What is tokenizer ?

**Ans:** String Tokenizer class in Java is used to break a string into tokens. In this we have hasMoreTokens( ) method which is Boolean type and nextToken( ) method which reads data init.

1. What is cloning ?

**Ans:** The process of creating the replica of particular object by copying the content of one object completely into another object is called cloning. We have cloneable which is marker interface that has to be implemented in class which we want to do cloning and we have clone method ( ) to copy object. and we may get cloneNotSupportException.

1. What is hash code ?

**Ans: H**ash Code in Java is **a function that returns the hash code value of an object on calling**. It returns an integer or a 4 bytes value which is generated by the hashing algorithm. The process of assigning a unique value to an object or attribute using an algorithm, which enables quicker access, is known as hashing

1. What is JDBC ?z

Ans: Java Database Connectivity is an application programming interface for the programming language Java, which defines how a client may access a database. It is a Java-based data access technology used for Java database connectivity. It is part of the Java Standard Edition platform

1. What is main principle of SQL ?

ans: SQL is used **to communicate with a database**. it is the standard language for relational database management systems. SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database.

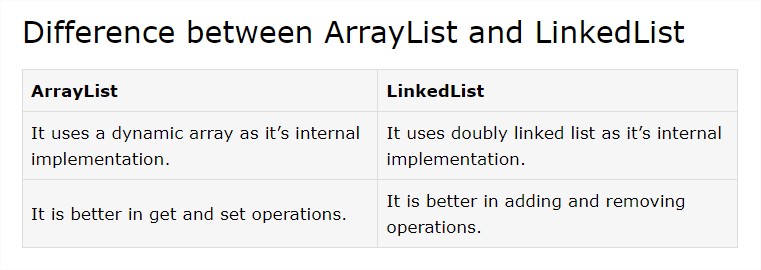
1. Explain about Collections ?

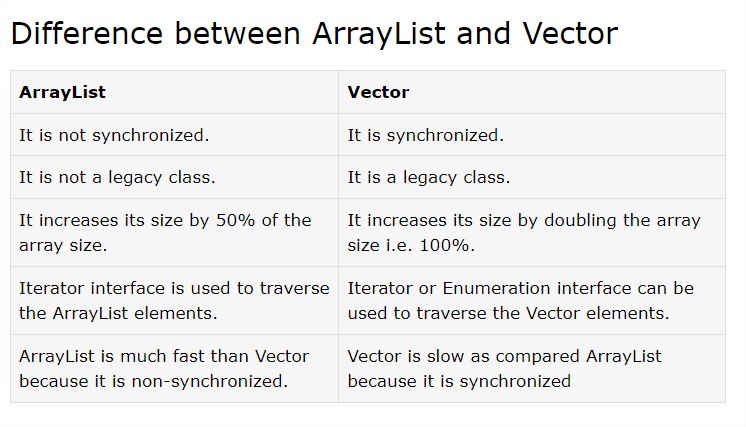
Ans: 1.**Collections** it is a utility class and

2.it is defines several utility methods that are used to operate on collection interfaces.like sort() method min() method Max() method and

3.it contains only static methods

1. Difference between Array list and Linked list , vectors ?





1. How to find the length of the list ? ( .size() )
2. Explain about hash table?

**Ans:**1.It stores the content as key value pair.

2.it is synchronized.

3.Thready safety is not achieved

4.Hash table does not allow any null values or key init.

5.It does not maintain insertion order.

1. What is hashing technique ?

Hashing is the process of mapping the data to some representative integer value using the concept of hashing algorithms. In Java, a hash code is an integer value that is linked with each object. Hashing finds its data structure implementation in HashTables and HashMaps.

Assume an array is consisting 5 buckets then it follows as like down:

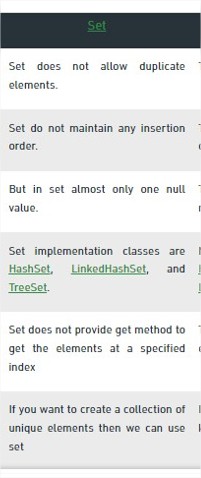
Key=”Mia” then each character is converted in ASCII code (i.e each character has its numeric value)& add the ASCII code and divide the combined ASCII number with the size of an element in this it is 5, we will take a reminder of the element that is 4 and there we will put the name Mia i.e. in 4th Slot/ Bucket.



1. What is collision ?

Ans: When two values are being stored at the same index number it is called as collision to solve this problem in hash table we store data as list is mapped to the same index number.

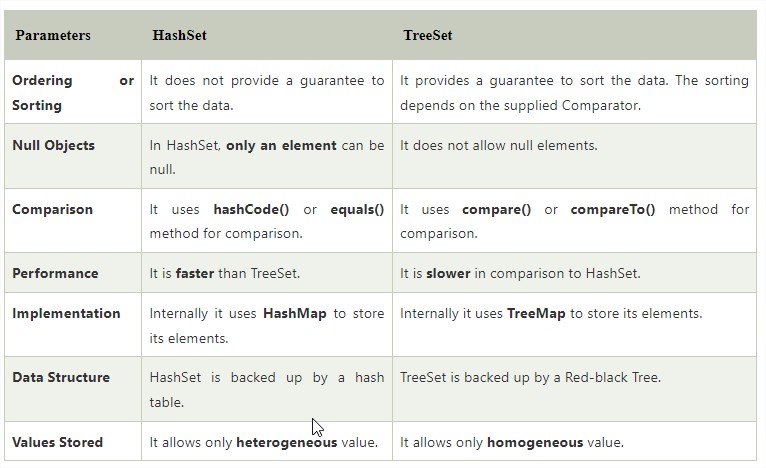
1. Explain about Set interface ?



1. Difference between List and Set ?

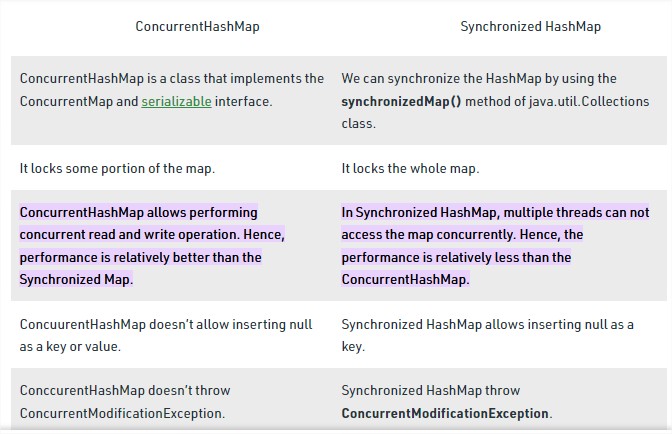


1. Difference between hash set and Tree set ?

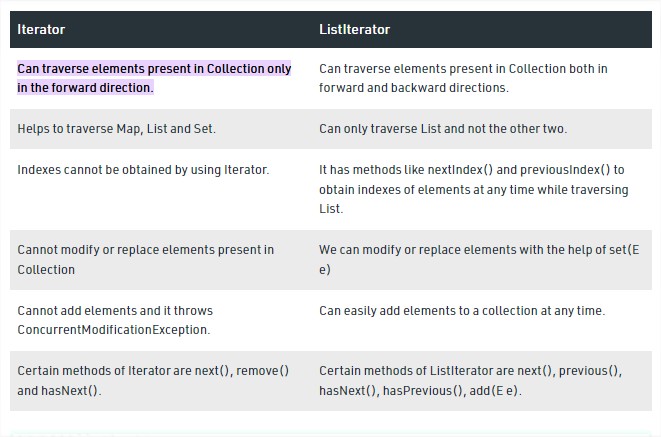


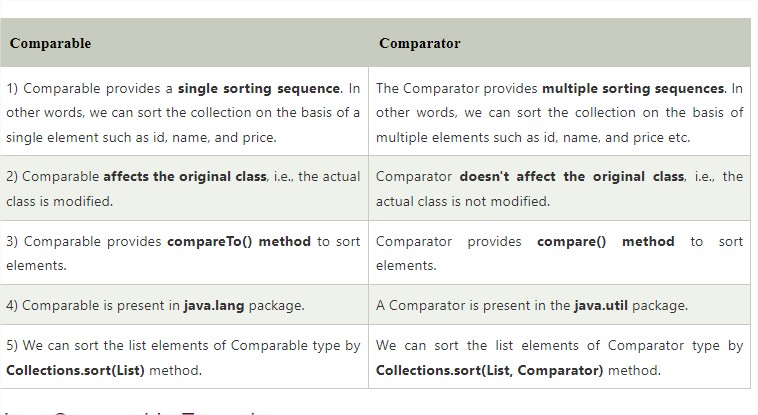
1. Difference between hash map and **concurrent hash map ?**

|  |  |
| --- | --- |
| **HashMap** | **Concurrent HashMap** |
| 1.HashMap is a class which is under traditional collection. | 1.Concurrent HashMap Is a class which is under concurrent collection. |
| 2.It is not synchronized | 2.it is synchronized. |
| 3.It is not achieved thread safety | 3.It achieve thread safety. |
| 4.HashMap performance is better. | 4.ConcurrentHashMap performance is slower than HashMap. |
| 5.In this While we add /modify the content we get ConcurrentModificationException. | 5.In this while we add/modify the content we don’t get any exception. |
| 6.Null values are allowed for single key and many null values . | 6.Null values is not allowed for both key and values. |



1. Difference between hash set , hash map , hash table ?
2. What is the common collection concept used in spring boot ? list interface, map
3. Difference between Iterator and List Iterator ?



1. Difference between Comparator and Comparable ? ans:***if sorting of*** 

59)difference between hashset and linked hashset?

|  |  |  |
| --- | --- | --- |
| **Property** | **HashSet** | **LinkedHashSet** |
| **Data structure** | It uses a Hashtable to store the elements. | It uses a HashTable and doubly linked list to store and maintain the insertion order of the elements. |
| **Technique to store the elements** | Hashing | Hashing |
| **Insertion Order** | It does not provide any insertion order. We can not predict the order of elements. | It provides an insertion order; we can predict the order of elements. |
| **Null elements** | It allows only one null element. | It also allows only one null element. |
| **Memory** | It requires less memory. | It requires more memory than HashSet. |
| **Performance** | It provides slightly faster performance than LinkedHashSet | It provides low performance than HashSet |
| **Synchronized** | Non-synchronized | Non-synchronized |
| **Complexity for the insertion, removal, retrieval operations** | O (1) | O (1) |
| **Declaration** | HashSet obj = new HashSet(); | LinkedHashSet obj = new LinkedHashSet(); |
| **Extends** | AbstractSet class | HashSet class |
| **Implements** | Set interface | Set interface |
| **Initial Capacity** | 16 | 16 |
| **Package** | java.util | Java.util |

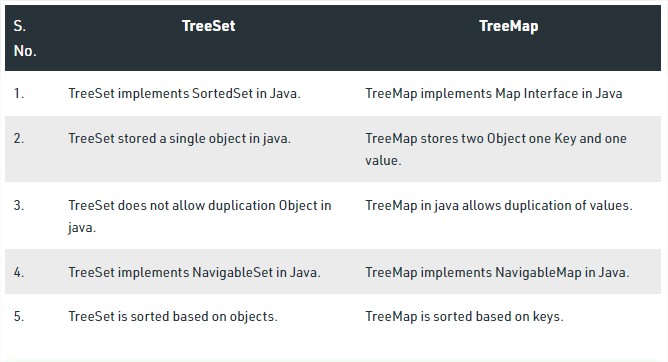
1. Difference between Hash Map and Hash Table.

|  |  |
| --- | --- |
| **HashMap** | **Hashtable** |
| 1) HashMap is **non synchronized**. It is not-thread safe and can't be shared between many threads without proper synchronization code. | Hashtable is **synchronized**. It is thread-safe and can be shared with many threads. |
| 2) HashMap **allows one null key and multiple null values**. | Hashtable **doesn't allow any null key or value**. |
| 3) HashMap is a **new class introduced in JDK 1.2**. | Hashtable is a **legacy class**. |
| 4) HashMap is **fast**. | Hashtable is **slow**. |
| 5) We can make the HashMap as synchronized by calling this code Map m = Collections.synchronizedMap(hashMap); | Hashtable is internally synchronized and can't be unsynchronized. |
| 6) HashMap is **traversed by Iterator**. | Hashtable is **traversed by Enumerator and Iterator**. |
| 7) Iterator in HashMap is **fail-fast**. | Enumerator in Hashtable is **not fail-fast**. |
| 8) HashMap inherits **AbstractMap** class. | Hashtable inherits **Dictionary** class. |

1. Difference between Hash Map and Concurrent Hash Map.

|  |  |  |
| --- | --- | --- |
| **Parameters** | **HashMap** | **ConcurrentHashMap** |
| **Synchronization** | Non-synchronized | synchronized |
| **Thread-safety** | Not thread-safe | Thread-safe |
| **Iterator** | It is fail-fast and throws an exception during iteration | It is fail-safe and performs iteration by multiple threads |
| **Null Values** | It allows for storing null keys and values. | It does not allow to store null key/values. |
| **Performance** | faster | Slower than Hashmap |

1. Difference between Tree Set and Tree Map



1. Difference between Iterator and Iterable.
2. What are the new versions of JDK ? **JDK18 (launched on march 22nd)**
3. How you implement rest full in project ?

By implementing RestTemplet class and it has method getForObject(url,EntityclassName.class)

### What is dynamic binding and static binding ? ans:

### *Static Binding*

The binding which can be resolved at compile time by the compiler is known as static or early binding. The binding of all the static, private, and final methods is done at compile-time.

### Dynamic Binding

In Dynamic binding compiler doesn’t decide the method to be called. Overriding is a perfect example of dynamic binding. In overriding both parent and child classes have the same method.

1. Write 2 -d array syntax : int arr[][]=new int[3][4] 3-rows and 4-columns
2. Explain about spring , spring bean , Spring bean scope ?

Ans: : Basically Spring we have 5 different types of bean scopes, which are described below.

### Types of Spring Bean Scopes :

**1) singleton:**It returns a single bean instance per Spring IoC container.

**2) prototype:**It returns a new bean instance each time when requested to create a bean.

**3) request:**It returns a single instance for every HTTP request.

**4) session:**It returns a single instance for the entire HTTP session.

**5) global session:**global session scope is equal to session scope on portlet-based web applications.

1. Code snippet for string object ?

Ans: **Literal:** Any constant value which can be assigned to the variable is called literal/constant.

**There are two ways to create a String object:**

1. By string literal : Java String literal is created by using double quotes. For Example: String s=“Welcome”;
2. By new keyword : Java String is created by using a keyword “new”. For example: String s=new String(“Welcome”);
3. Actuators ?

Ans: **Spring Boot Actuator** is a sub-project of the Spring Boot Framework. It includes a number of additional features that help us to monitor and manage the Spring Boot application. It contains the actuator endpoints (the place where the resources live). We can use **HTTP** and **JMX** endpoints to manage and monitor the Spring Boot application. **If we want to get production-ready features in an application, we should use the Spring Boot actuator.**

### **Spring Boot Actuator Features**

There are **three** main features of Spring Boot Actuator:

* **Endpoints (the place where the resources live)**
* **Metrics { a Java library which provides measuring instruments for Java applications}**
* **Audit { tracking and logging every change we make to our persisted data**,}

**For reference:**

Metrics is **a Java library which provides measuring instruments for Java applications**. It has several modules, and in this article, we will elaborate metrics-core module, metrics-healthchecks module, metrics-servlets module, and metrics-servlet module, and sketch out the rest, for your reference.

The **JMX** technology **enables Java applications to be managed without heavy investment**. A JMX technology-based agent (JMX agent) can run on most Java technology-enabled devices. Consequently, Java applications can become manageable with little impact on their design.

What is auditing in spring boot?

Auditing basically involves **tracking and logging every change we make to our persisted data**, which consists of tracking and storing every insert, update, and delete activity. Auditing aids in the preservation of history records, which can later be used to trace user activity.

1. Starters in spring boot ?

Ans: Before Spring Boot was introduced, Spring Developers used to spend a lot of time on Dependency management. Spring Boot **Starters** were introduced to solve this problem so that the developers can spend more time on actual code than Dependencies**. Spring Boot Starters are dependency descriptors** that can be added under the **<dependencies>** section in pom.xml.

**These starters give all the dependencies under a single name.**

**For example**, if you want to use Spring Data JPA for database access, you can include **spring-boot-starter-data-jpa**dependency.

1. Design patterns : Singleton , prototype

Ans: Design your class such that only one object of that class can be created.

1)By making constructors private i.e nothing but class can create only one object.

2)write a static method that has the return type object of this singleton class .here the concept of lazy initialization is used to write

This static method.

@**Autowired** follows singleton pattern in spring boot application.

The main purpose of singleton class is to restrict the limit no.of object creation to only one ,

Eg. Socket or database connection.

**Prototype** allows us to hide the complexity of making new instances from the client. The concept is to copy an existing object rather than creating a new instance from scratch

The **prototype pattern** is a creational design pattern. Prototype patterns are required, when object creation is time consuming, and costly operation, so we create objects with the existing object itself. One of the best available ways to create an object from existing objects is the **clone() method**. Clone is the simplest approach to implement a prototype pattern.

1. Java bi predicate?

Ans: The Functional Interface **PREDICATE** is defined in the java.util.Function package. It improves manageability of code, helps in unit-testing them separately, and contain some methods like:

Abstract method Test();

Static method equals();

Default methods or(),And(),notNull()

The **BiPredicate<T, V>** interface was introduced in **JDK 8**. This interface is packaged in **java.util.function** package. It operates on two objects and returns a predicate value based on that condition. It is a [functional interface](https://www.geeksforgeeks.org/functional-interfaces-java/) and thus can be used in [lambda expression](https://www.geeksforgeeks.org/lambda-expressions-java-8/)also.

Syntax: public interface BiPredicate<T, V>

1. API flow , Rest API?

Flow API is an a api which is introduced in java 9.

Ans: It’s composed of a few interfaces and only one implementation:

* The interface [Flow.Publisher](https://docs.oracle.com/javase/9/docs/api/java/util/concurrent/Flow.Publisher.html" \t "_blank" \o "interface in java.util.concurrent)<T> defines methods to produce items and control signals.
* The interface [Flow.Subscriber](https://docs.oracle.com/javase/9/docs/api/java/util/concurrent/Flow.Subscriber.html" \t "_blank" \o "interface in java.util.concurrent)<T> defines methods to receive those messages and signals.
* The interface [Flow.Subscription](https://docs.oracle.com/javase/9/docs/api/java/util/concurrent/Flow.Subscription.html" \t "_blank" \o "interface in java.util.concurrent) defines the methods to link both the Publisher and the Subscriber.
* The interface [Flow.Processor](https://docs.oracle.com/javase/9/docs/api/java/util/concurrent/Flow.Processor.html" \t "_blank" \o "interface in java.util.concurrent)<T,R> defines methods to do some advanced operations like chaining transformations of items from publishers to subscribers.
* Finally, the class [SubmissionPublisher](https://docs.oracle.com/javase/9/docs/api/java/util/concurrent/SubmissionPublisher.html" \t "_blank" \o "class in java.util.concurrent)<T>implements Flow.Publisher<T> and it's a flexible producer of items, compliant with the [Reactive Streams](http://www.reactive-streams.org/) initiative.

**A RESTful API** is an architectural style for an application program interface (API) that **uses HTTP requests to access and use data**. That data can be used to GET, PUT, POST and DELETE data types, which refers to the reading, updating, creating and deleting of operations concerning resources

1. JBOSS architecture
2. Stream methods

Different Operations On Streams-  
**Intermediate Operations:**

**1.map:**The map method is used to returns a stream consisting of the results of applying the given function to the elements of this stream.

List number = Arrays.asList(2,3,4,5);  
List square = number.stream().map(x->x\*x).collect(Collectors.toList());

**2.filter:** The filter method is used to select elements as per the Predicate passed as argument.  
List names = Arrays.asList("Reflection","Collection","Stream");  
List result = names.stream().filter(s->s.startsWith("S")).collect(Collectors.toList());

**3.sorted:** The sorted method is used to sort the stream.  
List names = Arrays.asList("Reflection","Collection","Stream");  
List result = names.stream().sorted().collect(Collectors.toList());

**Terminal Operations:**

1. **collect:** The collect method is used to return the result of the intermediate operations performed on the stream.  
   List number = Arrays.asList(2,3,4,5,3);  
   Set square = number.stream().map(x->x\*x).collect(Collectors.toSet());
2. **forEach:** The forEach method is used to iterate through every element of the stream.  
   List number = Arrays.asList(2,3,4,5);  
   number.stream().map(x->x\*x).forEach(y->System.out.println(y));
3. **reduce:** The reduce method is used to reduce the elements of a stream to a single value.  
   The reduce method takes a BinaryOperator as a parameter.

List number = Arrays.asList(2,3,4,5);  
int even = number.stream().filter(x->x%2==0).reduce(0,(ans,i)-> ans+i);

Here ans variable is assigned 0 as the initial value and i is added to it .

1. Copy constructor

In Java, **a copy constructor** is a special type of constructor that creates an object using another object of the same Java class. It returns a duplicate copy of an existing object of the class.

## **Copy Constructor Vs clone() Method**

* If we are using the clone() method it is necessary to import the **Cloneable** The method may throw the exception **CloneNotSupportException.** So, handling the exception in a program is a complex task. While in copy constructor there are no such complexities.
* We cannot assign a value if the fields are **final**. While in the copy constructor we can assign values to the final fields.
* The object returned by the clone() method must be **typecast**. While in copy constructor there is no such requirement.

1. Types of constructor

There are 3 types of constructors :

* 1. No-args constructor
  2. Parametrized constructor
  3. Default constructor

1. Executor service

Ans: The **Java ExecutorService** is the interface which allows us to execute tasks on threads asynchronously. The Java ExecutorService interface is present in the java.util.concurrent package. **The ExecutorService helps in maintaining a pool of threads and assigns them tasks. It also provides the facility to queue up tasks until there is a free thread available if the number of tasks is more than the threads available.**

## Task Delegation

Here is a diagram illustrating a thread delegating a task to a Java ExecutorService for asynchronous execution:

|  |
| --- |
| A thread delegating a task to an ExecutorService for asynchronous execution. |
| **A thread delegating a task to an ExecutorService for asynchronous execution.** |

Once the thread has delegated the task to the ExecutorService, the thread continues its own execution independent of the execution of that task. The ExecutorService then executes the task concurrently, independently of the thread that submitted the task.

We can use Java ExecutorService to create a single thread, a pool of threads, or a scheduled pool of threads. The Executors class provides factory methods to instantiate an ExecutorService as follows-

* execute(Runnable)
* submit(Runnable)
* submit(Callable)
* invokeAny(...)
* invokeAll(...)
* executorService.shutdown();

1. Call procedure from .java

Ans: The CallableStatement of JDBC API is used to call a stored procedure. A Callable statement can have output parameters, input parameters, or both. The prepareCall() method of connection interface will be used to create CallableStatement object.

1. J unit
2. **Login framework**
3. Expose and consume of web services?
4. Hibernate JPA vs JPA repository
5. Prepared statements vs statements

Ans:



|  |  |
| --- | --- |
| Statement | PreparedStatement |
| It is used when SQL query is to be executed only once. | It is used when SQL query is to be executed multiple times. |
| You can not pass parameters at runtime. | You can pass parameters at runtime. |
| Used for CREATE, ALTER, DROP statements. | Used for the queries which are to be executed multiple times. |
| Performance is very low. | Performance is better than Statement. |
| It is base interface. | It extends statement interface. |
| Used to execute normal SQL queries. | Used to execute dynamic SQL queries. |
| We can not use statement for reading binary data. | We can use Preparedstatement for reading binary data. |



1. JDBC template?

Ans: JdbcTemplate class is **the central class in the JDBC core package**. It simplifies the use of JDBC and helps to avoid common errors. It executes core JDBC workflow, leaving the application code to provide SQL and extract results.

1. What is SDLC ?

Ans: Software Development Life Cycle (SDLC) is **a process used by the software industry to design, develop and test high quality softwares**. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

1.water fall model

2.agile

1. What are present in object class ?

|  |
| --- |
| The Object class provides many methods. They are as follows: |

|  |  |
| --- | --- |
| **Method** | **Description** |
| public final Class getClass() | returns the Class class object of this object. The Class class can further be used to get the metadata of this class. |
| public int hashCode() | returns the hashcode number for this object. |
| public boolean equals(Object obj) | compares the given object to this object. |
| protected Object clone() throws CloneNotSupportedException | creates and returns the exact copy (clone) of this object. |
| public String toString() | returns the string representation of this object. |
| public final void notify() | wakes up single thread, waiting on this object's monitor. |
| public final void notifyAll() | wakes up all the threads, waiting on this object's monitor. |
| public final void wait(long timeout)throws InterruptedException | causes the current thread to wait for the specified milliseconds, until another thread notifies (invokes notify() or notifyAll() method). |
| public final void wait(long timeout,int nanos)throws InterruptedException | causes the current thread to wait for the specified milliseconds and nanoseconds, until another thread notifies (invokes notify() or notifyAll() method). |
| public final void wait()throws InterruptedException | causes the current thread to wait, until another thread notifies (invokes notify() or notifyAll() method). |
| protected void finalize()throws Throwable | is invoked by the garbage collector before object is being garbage collected. |

1. Spring security

Ans: Spring Security is a framework which provides various security features like: authentication, authorization to create secure Java Enterprise Applications.

## **Advantages**

Spring Security has numerous advantages. Some of that are given below.

* Comprehensive support for authentication and authorization.
* Protection against common tasks
* Servlet API integration
* Integration with Spring MVC
* Portability

1. Email API
2. Parallel stream (java 8)

Stream in java by generally it is sequential Stream () it is synchronized. we can convert sequential to parallel by parallelStream () it is not synchronized.

Ans: Any stream in Java can easily be transformed from sequential to parallel.

We can achieve this by **adding the *parallel* method to a sequential stream or by creating a stream using the *parallelStream*method of a collection**:

List<Integer> listOfNumbers = Arrays.asList(1, 2, 3, 4);

listOfNumbers.parallelStream().forEach(number ->

System.out.println(number + " " + Thread.currentThread().getName())

);

1. Types of design pattern ?

Ans: **Design Patterns** are very popular among software developers. A design pattern is a well-described solution to a common software problem

**Java Design Patterns** are divided into three categories – **creational**, **structural**, and **behavioral** design patterns

1. How to optimize code ?

Ans: It is necessary that the code which we are writing is not only clean, without defects but also optimized i.e. the time taken by the code to execute should be within intended limits. In order to achieve this, we need to refer to the Java coding standards and review our code to make sure if it is as per the standards.

**1. Avoid Writing Long Methods**

**2. Avoid Multiple If-else Statements**

**3. Avoid Getting the Size of the Collection in the Loop**

**4. Avoid Using String Objects For Concatenation**

**5. Use Primitive Types Wherever Possible**

**6. Using**[PreparedStatement](https://www.geeksforgeeks.org/difference-between-statement-and-preparedstatement/)**instead of Statement**

**7. Select Required Columns in a Query**

**8. Fetch the Data Using Joins**

1. Bean factory and application context ?
2. How to exclude in spring boot ?
3. Types of inheritance in hibernate?

We can map the inheritance hierarchy classes with the table of the database. There are three inheritance mapping strategies defined in the hibernate:

1. Table Per Hierarchy
2. Table Per Concrete class
3. Table Per Subclass
4. Types of bean scope?

### Ans: Types of Spring Bean Scopes :

**1) singleton:**It returns a single bean instance per Spring IoC container.

**2) prototype:**It returns a new bean instance each time when requested to create a bean.

**3) request:**It returns a single instance for every HTTP request.

**4) session:**It returns a single instance for the entire HTTP session.

**5) global session:**global session scope is equal to session scope on portlet-based web applications.

1. Types of Auto wiring

In Spring framework, you can wire beans automatically with auto-wiring feature. To enable it, just define the “autowire” attribute in <bean>. In Spring, 5 Auto-wiring modes are supported.

***no***: Default, no auto wiring, set it manually via “ref” attribute

***byName***: Auto wiring by property name. If the name of a bean is same as the name of other bean property, auto wire it.

***byType***: Auto wiring by property data type. If data type of a bean is compatible with the data type of other bean property, auto wire it.

***constructor***: byType mode in constructor argument.

***autodetect***: If a default constructor is found, use “autowired by constructor”; Otherwise, use “autowire by type”.

1. Key components of spring boot

**1.spring actuator**

**2.spring starter**

**3.spring CLI**

**4.spring Auto Configurator**

1. Drop vs truncate

**Difference between DELETE, DROP and TRUNCATE**

Ans: **1. DELETE :**

Basically, it is a [Data Manipulation Language Command (DML)](https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/). It is used to delete one or more tuples of a table. With the help of the “DELETE” command, we can either delete all the rows in one go or can delete rows one by one. i.e., we can use it as per the requirement or the condition using the Where clause. It is comparatively slower than the TRUNCATE command. The TRUNCATE command does not remove the structure of the table.

* **SYNTAX –**   
  If we want to delete all the rows of the table:

DELETE from;

* **SYNTAX –**   
  If we want to delete the row of the table as per the condition then we use the WHERE clause,

DELETE from WHERE ;

**Note –** Here we can use the “ROLLBACK” command to restore the tuple because it does not auto-commit.

**2. DROP :**

It is a Data Definition Language Command (DDL). It is used to drop the whole table. With the help of the “DROP” command we can drop (delete) the whole structure in one go i.e. it removes the named elements of the schema. By using this command the existence of the whole table is finished or say lost.

* **SYNTAX –**   
  If we want to drop the table:

DROP table ;

**Note –** Here we can’t restore the table by using the “ROLLBACK” command because it auto commits.

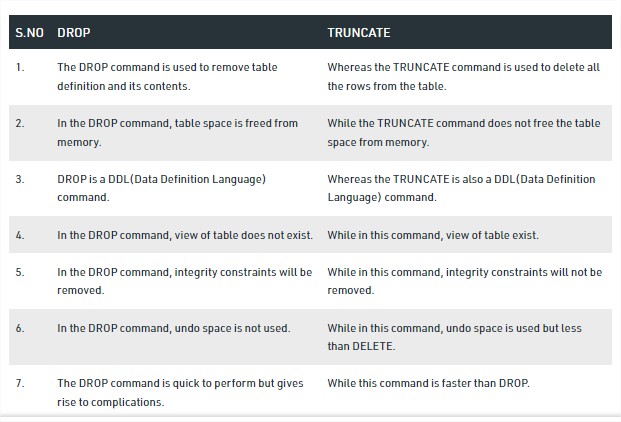
**3. TRUNCATE :**

It is also a Data Definition Language Command (DDL). It is used to delete all the rows of a relation (table) in one go. With the help of the “TRUNCATE” command, we can’t delete the single row as here WHERE clause is not used. By using this command the existence of all the rows of the table is lost. It is comparatively faster than the delete command as it deletes all the rows fastly.

* **SYNTAX –**   
  If we want to use truncate :

TRUNCATE;

**Note –** Here we can’t restore the tuples of the table by using the “ROLLBACK” command.



1. Modes of auto wired

Ans: there are 5 modes :

1.no

2.ByName

3.ByType

4.Constructor

5.autoDetect

1. Spring security

Ans: Spring Security is **a framework which provides various security features like: authentication, authorization to create secure Java Enterprise Applications**.

* for *authentication* of users, to reliably and securely determine who is currently executing Java code, regardless of whether the code is running as an application, an applet, a bean, or a servlet; and
* for *authorization* of users to ensure they have the access control rights (permissions) required to do the actions performed.

1. Spring IOC

* Spring IOC is an area in spring boot framework that perform dependency injection and maintains complete lifecycle of spring Bean.
* Spring IOC consists of core of spring framework.

1. Streams in java(Stream api which was introduced in java8)
2. Aggregation composition

Ans: Objects are related to each other using more than one relationship, such as Aggregation, Composition, Association, etc.

**Composition** is a **"belong-to"** type of relationship in which one object is logically related with other objects. It is also referred to as **"has-a"** relationship.

Composition is a strong type of **"has-a"** relationship because the containing object is its owner. So, objects are tightly coupled, which means if we delete the parent object, the child object will also get deleted with it.

**Aggregation** relationship is also a **"has-a"** relationship. The only difference between Aggregation and Composition is that in Aggregation, objects are not tightly coupled or don't involve owning. All the objects are independent of each other and can exist even if the parent object gets deleted.



**Association** is a relation between two separate classes which establishes through their Objects. Association can be one-to-one, one-to-many, many-to-one, many-to-many. In Object-Oriented programming, an Object communicates to another object to use functionality and services provided by that object. **Composition** and **Aggregation** are the two forms of association.

1. Fail fast ,fail safe

Ans: [Iterators](https://contribute.geeksforgeeks.org/iterators-in-java/) in java are used to iterate over the Collection objects.

**Fail-Fast iterators** immediately throw *ConcurrentModificationException* if there is **structural modification** of the collection. Structural modification means adding, removing any element from collection while a thread is iterating over that collection. Iterator on ArrayList, HashMap classes are some examples of fail-fast Iterator.  
**Fail-Safe iterators** don’t throw any exceptions if a collection is structurally modified while iterating over it. This is because, they operate on the clone of the collection, not on the original collection and that’s why they are called fail-safe iterators. Iterator on CopyOnWriteArrayList, ConcurrentHashMap classes are examples of fail-safe Iterator.

1. @component

Ans: **@Component** is **an annotation that allows Spring to automatically detect our custom beans**. In other words, without having to write any explicit code, Spring will: Scan our application for classes annotated with @Component. Instantiate them and inject any specified dependencies into them.

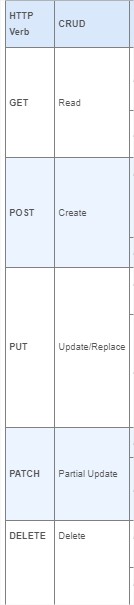
**For reference:** With @Component , @Repository , @Service and @Controller annotations in place and automatic component scanning enabled, Spring will **automatically import the beans into the container and inject to dependencies**. These annotations are called Stereotype annotations as well.

1. What is server validation and Client validation

Ans:

|  |  |
| --- | --- |
| **Client-Side Validation** | **Server-Side Validation** |
| In client-side validation method, all the input validation and error recovery process are carried out on the client side. | In server-side validation method, all the input validations and error recovery process are carried out on the server side. |
| This sort of approval is finished utilizing JavaScript. | This sort of approval is done at the server side where application dwells. |
| This kind of approval is quick and simpler for client. | This kind of approval is similar slower for client. |
| This sort of approval is for the most part shaky as the end client has simple access to the code of the page. | This sort of approval is more secure as the end client does not have simple access to the code. |
| This sort of approval is for the most part done first. | This kind of approval is for the most part done after customer side approval. |
| This sort of approval is done to lessen the blunders which can come amid server side preparing. | This kind of approval is done to safeguard any sort of harm which can come amid customer side approval. |
| It can be done using JavaScript, AJAX, HTML5 etc. | It can be done using programming languages like C#.NET, VB.NET, Java and JSP, Python, Ruby on Rails etc. |

1. What is SAM interface? **(single abstract method)**
2. Difference between var and let in JavaScript?
3. HTTP methods in spring boot?

Ans: 

1. Agile methodology?

Ans: It is a methodology that helps us to built the software quickly in least possible time without compromising with quality of work.

1. Difference between monolithic architecture and micro service?

| **Sr. No.** | **Key** | **Monolithic architecture** | **Microservices architecture** |
| --- | --- | --- | --- |
| 1 | Basic | Monolithic architecture is built as one large system and is usually one code-base | Microservices architecture is built as small independent module based on business functionality |
| 2 | Scale | It is not easy to scale based on demand | It is easy to scale based on demand. |
| 3 | Database | It has shared database | Each project and module has their own database |
| 4 | Deployment | Large code base makes IDE slow and build time gets increase. | Each project is independent and small in size. So overall build and development time gets decrease. |
| 5 | Tightly Coupled and Loosely coupled | It extremely difficult to change technology or language or framework because everything is tightly coupled and depend on each other | Easy to change technology or framework because every module and project is independent |

1. OAUTH2, pact tool?
2. Semantic monitory

Ans: Semantic monitoring **uses synthetic transactions and emulates end-user flows through a microservices-based application to track performance and alert on anomalies**. Semantic monitoring runs a set of automated tests against an application on production systems at regular intervals

1. Clint certificate
2. What you will do if micro service fail?
3. Ans: Use asynchronous communication (for example, message-based communication) across internal microservices. ...
4. Use retries with exponential backoff. ...
5. Work around network timeouts. ...
6. Use the Circuit Breaker pattern. ...
7. Provide fallbacks.
8. Hibernate architecture (refer notes)
9. Spring initializer(refer notes)
10. Spring security **see above**
11. Difference between bean factory and application context(refer notes)4
12. Different spring Boot scopes

Ans: The latest version of the Spring framework defines 6 types of scopes:

* singleton
* prototype
* request
* session
* application
* WebSocket

1. Different hibernate objects

Ans: 1. entity object

2.object of session factory interface

3.object of session

4.object of transaction

5.object of configuration.

Difference between mapping and configuration classes in hibernate?

**Configuration:**

* Configuration is a class which is present in org.hibernate.cfg package. It activates Hibernate framework. It reads both configuration file and mapping files.
* It activate Hibernate Framework
* **Configuration cfg=new Configuration();**
* It read both cfg file and mapping files
* **cfg.configure();**
* It checks whether the config file is syntactically correct or not.
* If the config file is not valid then it will throw an exception. If it is valid then it creates a meta-data in memory and returns the meta-data to object to represent the config file.

Mapping file can be used when a pojo class without any annotations .

1. Load balancing

Ans: The load balancer **attempts to evenly distribute the workload among multiple Application Server instances (either stand-alone or clustered), thereby increasing the overall throughput of the system**. Using a load balancer also enables requests to fail over from one server instance to another.

1. Circuit breaker:

Ans: Circuit breaker is **a design pattern used in modern software development**. It is used to detect failures and encapsulates the logic of preventing a failure from constantly recurring, during maintenance, temporary external system failure or unexpected system difficulties

1. Jeera.
2. GIT HUB repository
3. Front controller it is one of design pattern

The front controller design pattern means that all requests that come for a resource in an application will be handled by a single handler and then dispatched to the appropriate handler for that type of request. The front controller may use other helpers to achieve the dispatching mechanism.

1. @ qualifier:

Whenever spring boot has confusion which object address need to be injected into the reference variable. we resolve this confusion by using this annotation.

1. @path variable?

It helps us to map url data to method argument in webServices.

1. Count number of objects in java?

Ans: We create a static int type variable and put this a static variable with an increment operator so that it increases by 1 in the constructor.

// Java program Find Out the Number of Objects Created

// of a Class

class Test {

static int noOfObjects = 0;

// Instead of performing increment in the constructor

// instance block is preferred to make this program generic.

{

noOfObjects += 1;

}

// various types of constructors

// that can create objects

public Test()

{

}

public Test(int n)

{

}

public Test(String s)

{

}

public static void main(String args[])

{

Test t1 = new Test();

Test t2 = new Test(5);

Test t3 = new Test("GFG");

// We can also write t1.noOfObjects or

// t2.noOfObjects or t3.noOfObjects

System.out.println(Test.noOfObjects);

}

}

1. Functional interface sub methods and types

### Ans: Some Built-in Java Functional Interfaces

Since Java SE 1.8 onwards, there are many interfaces that are converted into functional interface. All these interfaces are annotated with @FunctionalInterface. These interfaces are as follows –

* **Runnable –>** This interface only contains the run() method.
* **Comparable –>** This interface only contains the compareTo() method.
* **ActionListener –>** This interface only contains the actionPerformed() method.
* **Callable –>** This interface only contains the call() method.

**Java SE 8 included four main kinds of functional interfaces**which can be applied in multiple situations. These are:

1. Consumer
2. Predicate
3. Function
4. Supplier

Amidst the previous four interfaces, the first three interfaces,i.e., Consumer, Predicate, and Function, likewise have additions that are provided beneath –

1. Consumer -> Bi-Consumer
2. Predicate -> Bi-Predicate
3. Function -> Bi-Function, Unary Operator, Binary Operator

#### **1. Consumer**

The consumer interface of the functional interface is the one that accepts only one argument or a gentrified argument. The consumer interface has no return value.

**Syntax / Prototype of Consumer Functional Interface –**

Consumer<Integer> consumer = (value) -> System.out.println(value);

**2.predicate**

**predicate functional** interface of java is a type of function which accepts a single value or argument and does some sort of processing on it, and returns a boolean (True/ False) answer. The implementation of the Predicate functional interface also encapsulates the logic of filtering (a process that is used to filter stream components on the base of a provided predicate) in Java.

**Bi-Predicate –**Bi-Predicate is also an extension of the Predicate functional interface, which, instead of one, takes two arguments, does some processing, and returns the boolean value.

**Syntax**: Predicate predicate = (value) -> value != null;

#### **3. Function**

A function is a type of functional interface in Java that receives only a single argument and returns a value after the required processing

#### **4. Supplier**

The Supplier functional interface is also a type of functional interface that does not take any input or argument and yet returns a single output.

1. How will you pass xml in application Properties?

To convert XML file into properties file, best way is to use java.util.Properties class. Process is :

1. Load XML file into java.util.Properties class object, using [Properties.loadFromXML()](https://docs.oracle.com/javase/10/docs/api/java/util/Properties.html" \l "loadFromXML(java.io.InputStream)" \t "_blank) method.
2. Use [Properties.store()](https://docs.oracle.com/javase/10/docs/api/java/util/Properties.html" \l "store(java.io.OutputStream,java.lang.String)" \t "_blank) method to write the content as properties.

|  |
| --- |
| **package** com.howtodoinjava.demo;    **import** java.io.FileInputStream;  **import** java.io.FileOutputStream;  **import** java.io.IOException;  **import** java.io.InputStream;  **import** java.io.OutputStream;  **import** java.util.InvalidPropertiesFormatException;  **import** java.util.Properties;    **public** **class** XMLToProperties  {  **public** **static** **void** main(String[] args) **throws** InvalidPropertiesFormatException, IOException    {      String outPropertiesFile = "application.properties";      String inXmlFile = "applicationProperties.xml";        InputStream inStream = **new** FileInputStream(inXmlFile);      //Input XML File      OutputStream outStream = **new** FileOutputStream(outPropertiesFile); //Output properties File        Properties props = **new** Properties();        //Load XML file      props.loadFromXML(inStream);        //Store to properties file      props.store(outStream, "Converted from applicationProperties.xml");        //Use properties in code      System.out.println(props.get("input.dir"));     //Prints 'c:/temp/input'    }  } |
| XMLToProperties.java |

1. What is postman?

Ans: postman is software tool used to test web services.

Postman is **an API client that makes it easy for developers to create, share, test and document APIs**. This is done by allowing users to create and save simple and complex HTTP/s requests, as well as read their responses. The result - more efficient and less tedious work.

1. Repository layer and how it will work?

**Java repositories** are commonly known as JPA based repositories used under JAVA spring framework. Repositories define a new elegant method of storing, updating, and extracting the data +………..stored from JAVA applications in the backend. All of the CRUD (Create, read, update, and delete) operations can be implemented with the help of a repository interface.

**How does the JPA Repository Work?**

JPA repositories are created by extending the JpaRepository library consisting of implementation of different functions, methods, and other related dependent data types to enable persistence in web or desktop applications designed using JAVA. To have these interfaces working all the dependent libraries are to be loaded in the classpath.

**Once the interface is created then functions like “save()”, “count()”, “info()”, “findAll()”, “sort()” and others are used to accomplish the data query or required data manipulation**

1. Where we store java code?

Ans: In the standard Java development environment, Java source code, binaries, and resources are stored as files in a file system, as follows:

* Source code files are saved as . java files.
* Compiled Java binary files are saved as .class files.
* Resources are any data files, such as .properties or .ser files, that are stored in the file system hierarchy and are loaded and used at run time.

1. Jenkins

Ans:**Jenkins** – an open source automation server which enables developers around the world to reliably build, test, and deploy their software.

1. Difference between session and session factory?
2. Replace and replace all methods?

Ans:

**replace()** : The **Java String class replace()** method returns a string replacing all the old char or Char Sequence to new char or Char Sequence.

There are two types of replace() methods in Java String class

1**.public** String replace(**char** oldChar, **char** newChar)

**e.g:**

1. **public** **class** ReplaceExample1{
2. **public** **static** **void** main(String args[]){
3. String s1="javatpoint is a very good website";
4. String replaceString=s1.replace('a','e');//replaces all occurrences of 'a' to 'e'
5. System.out.println(replaceString);
6. }}

2.**public** String replace(CharSequence target, CharSequence replacement)

**e.g:**

1. **public** **class** ReplaceExample2{
2. **public** **static** **void** main(String args[]){
3. String s1="my name is khan my name is java";
4. String replaceString=s1.replace("is","was");//replaces all occurrences of "is" to "was"
5. System.out.println(replaceString);
6. }}

**replaceAll()** :The Java String class **replaceAll()** method returns a string replacing all the sequence of characters matching regex and replacement string.

**Syntax: public** String replaceAll(String regex, String replacement)

e.g:

1. **public** **class** ReplaceAllExample2{
2. **public** **static** **void** main(String args[]){
3. String s1="My name is Khan. My name is Bob. My name is Sonoo.";
4. String replaceString=s1.replaceAll("is","was");//replaces all occurrences of "is" to "was"
5. System.out.println(replaceString);
6. }}
7. How to read application properties file?

Ans:

1.Please note that to be able to read a property from application.properties file I needed to use the @Autowired annotation to inject the Environment object. Then I can simply call its getProperty(String key) method to get the value of a requested property.

2. Another very simple way to read application properties is to use @Value annotation. Simply annotation the class field with @Value annotation providing the name of the property you want to read from application. Properties file and class field variable will be assigned that value.

1. Trust full API?
2. How to copy one hash map to other?

Ans: Given a [HashMap](https://www.geeksforgeeks.org/java-util-hashmap-in-java-with-examples/), there are three ways one can copy the given **HashMap** to another:

1. By normally iterating and putting it to another HashMap using put(k, v) method.
2. Using putAll() method.
3. Using copy constructor.
4. Address of last data in linked list?**(null)**
5. Which control used in project?
6. What is DDL, DML

Ans: **DDL is Data Definition Language which is used to define data structures**. For example: create table, alter table are instructions in SQL. DML: **DML is Data Manipulation Language which is used to manipulate data itself**. For example: insert, update, delete are instructions in SQL.

1. Difference between Web server and Application server

(Refer notes)

1. How to test Web service in postman?

Ans: **Basics of API Testing Using Postman**

1. Go to your workspace in Postman.
2. Click on the + symbol to open a new tab.
3. Enter the API Endpoint where it says, “Enter request URL” and select the method (action type GET, POST, etc.) for that request as shown below.
4. Click on the Send button.
5. Can multiple finally blocks execute? no
6. How to connect two tables in spring?

@Entity

**public** **class** **Employee** {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

**private** **long** id;

**private** String name;

**private** **int** age;

**@ManyToOne**

**private** Department department;

**@OneToMany(mappedBy = "employee")**

**private** List<Phone> phones;

// getters and setters...

}

Each Employee will be assigned to only one Department:

Finally, each Employee will have multiple Phones:

# ADVANCE JAVA QUESTIONS

1. Explain the life cycle of Servlet , JSP ?

Servlet:initially servlet present inside the tomcat container.for the first time when we start the tomcat init()method will run only once .but the services()method in which we have dopost()and doget() can be called any no of times.finally when the destroy()method is called servlet life comes to an end.

JSP:what ever java code we write in jsp file with the help of jsp translator we translate the java code into servlet .after tomcat is started JSP init()method will run only once .but the -\_jspservices()method can be called any no of times.finally when the jspdestroy()method is called jsp life comes to an end.destroy()method will be called automatically.

### **Advantages of JSP over Servlet**

There are many advantages of JSP over the Servlet. They are as follows:

#### **1) Extension to Servlet**JSP technology is the extension to Servlet technology. We can use all the features of the Servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy.

#### **2) Easy to maintain**JSP can be easily managed because we can easily separate our business logic with presentation logic. In Servlet technology, we mix our business logic with the presentation logic.

#### **3) Fast Development: No need to recompile and redeploy**If JSP page is modified, we don't need to recompile and redeploy the project. The Servlet code needs to be updated and recompiled if we have to change the look and feel of the application.

#### **4) Less code than Servlet**In JSP, we can use many tags such as action tags, JSTL, custom tags, etc. that reduces the code. Moreover, we can use EL, implicit objects, etc.

1. What is servlet , jsp , html pages ?

**Servlet** :it is a java class and it is the subclass in the http servlet it is used to interact with the front end applications.

**Jsp:** (Jakarta server page) jsp is a collection of technologies that help software developers create dynamically generated webpages based on html,xml,soap or other document type. Along with the html code we can also insert partial java code in it and this will help us to do the development in a much easier manner.in order too write java code we use something called as jsp tags.

1.servlet tag(<%.......%>)

2.descriptive tag(<%!........%>)

3.expression tag(<%=………%>)

4.directive tag(<[%@..............%](mailto:%25@..............%25)>)

**Html pages**:html stands for hyper text markup language .html is the standard markup language for creating web pages.it is used to describe the structure of a web pages.it consists of a series of elements .this elements tell the browser how to display the content.html elements label pieces of content such as “this is a heading”,”this is a paragraph”,”this is a link”etc..

157)What is Inter Servlet connectivity ? ( ISC )

When one servlet calls another servlet its called as inter servlet connectivity.

1. What are session variables ?

Session: A **session** refers to all the connections that a single client makes to a server in the course of viewing any pages associated with a given application. Sessions are specific(fixed) to both the individual user and the application. As a result, every user of an application has a separate session and has access to a separate set of session variables.  
  
This logical view of a session begins with the first connection to an application by a client and ends after that client's last connection. However, because of the stateless nature of the web, it is not always possible to define a precise(exact) point at which a session ends. A session should end when the user finishes using an application. In most cases, however, a web application has no way of knowing if a user has finished or is just lingering over a page.

Session variables once the value is stored in session variable then that value can be access across the application it doesn’t matter how we call that method (with the requestdispatcher or we call using url). If we wont use request Dispatcher then request.getAttribute gives null value

e.g from login servlet from mini project to create session variable

HttpSession session = request.getSession(**true**);//create a new session

session.setAttribute("email", email);

session.setMaxInactiveInterval(180);

E.g; from logout servlet

HttpSession session = request.getSession(**false**);// calls the existing session

session.invalidate(); // destroy session

**other servlets we need to give like**

HttpSession session = request.getSession(**false**);

**if** (session.getAttribute("email") != **null**)

1. Explain about JSP tags ?

**Jsp:** (Jakarta server page) jsp is a collection of technologies that help software developers create dynamically generated webpages based on html,xml,soap or other document type. Along with the html code we can also insert partial java code in it and this will help us to do the development in a much easier manner.in order too write java code we use something called as jsp tags.

The scripting elements provides the ability to insert java code inside the jsp.

**scriptlet tag:**A  scriptlet tag is used to execute java source code in JSP.in scriptlet tag we can not develop a method and we can not use the access specifiers.

Syntax is as follows:<%  java source code %>

**expression tag**:A expression tag are used to print the output on the web page.in expression tag we can not write more than one statement .do not end your statement with semicolon while using expression tags.

Syntax of JSP expression tag**: <**%=  statement %**>**

**declaration tag**:if we want to develop the methods and if we want to use the access specifiers we use declaration tag.in declaration tag we can not use implicit objects in it.the implicit objects in jsp are out,request,response,session,config,application,pagecontext,page,exception.

The syntax of the declaration tag is as follows: Hello**<**%!  field or method declaration%>

### **Directive tag:**The **jsp directives** are messages that tells the web container how to translate a JSP page into the corresponding servlet.

### **JSP page directive**

### The page directive defines attributes that apply to an entire JSP page.(Attributes of JSPpagedirective**:**import,contentType,extends,info,buffer,language,isELIgnored,isThreadSafeautoFlush,session,pageEncoding,errorPage,isErrorPage.)

### Syntax of JSP page directive<%@ page attribute="value" %>

# Jsp Include Directive

The include directive is used to include the contents of any resource it may be jsp file, html file or text file. The include directive includes the original content of the included resource at page translation time (the jsp page is translated only once so it will be better to include static resource).

### Syntax of include directive<%@ include file="resourceName" %>

# JSP Taglib directive

# The JSP taglib directive is used to define a tag library that defines many tags. We use the TLD (Tag Library Descriptor) file to define the tags. In the custom tag section we will use this tag so it will be better to learn it in custom tag.

#### **Syntax JSP Taglib directive**

#### <%@ taglib uri="uriofthetaglibrary" prefix="prefixoftaglibrary" %>

1. Explain about MVC architecture ?

Here we are dividing the application into 3layed application.

MODEL,VIEW,CONTROLLER

* All the front end code like html/jsp is develop in the view layer of the application.
* Controller layer is responsible to interact with the view layer receive the input and futher passes on those inputs to the model layer and also takes the output from the model layer and gives it to the view.usually we use servlets to built controller layer .
* In model layer all the logical implementation or database operations are performed.

1. Write a logic for JDBS connection ?

JDBC is an [Application Programming Interface](https://www.geeksforgeeks.org/introduction-to-apis/) for Java which connects a Java application with a database to perform **CRUD** operations.

Connection con=DriverManager.getConnection(“jdbc:mysql://localhost:3306/db name”,”username”,”password”);

1. Explain about CSS and Java Script ?

**CSS**:CSS stands for cascading style sheets.it is a style sheet language which is used to describe the look and formatting of a document written in the markup language.it provides an additional features to html.it is generally used to change the style of web pages and user interfaces.it can also be used with any kind of xml documents. **JAVASCRIPT:** JavaScript is used to create client-side dynamic pages.javascript is an object based scripting language which is lightweight and cress platform.

1. Explain about JSTL tags ?

JAKARTA STANDARD TAG LIBRARY(JSP STANDARD TAG LIBRARY)

The JSP Standard Tag Library (JSTL) represents a set of tags to simplify the JSP development.

Jstl tags helps us to write java code In the form of tags .which would make our development much easier.

## **Advantage of JSTL**

1. **Fast Development** JSTL provides many tags that simplify the JSP.
2. **Code Reusability** We can use the JSTL tags on various pages.
3. **No need to use scriptlet tag** It avoids the use of scriptlet tag.

## **JSTL Tags**

There JSTL mainly provides five types of tags:

|  |  |
| --- | --- |
| **Tag Name** | **Description** |
| [Core tags](https://www.javatpoint.com/jstl-core-tags) | The JSTL core tag provide variable support, URL management, flow control, etc. The URL for the core tag is **http://java.sun.com/jsp/jstl/core**. The prefix of core tag is **c**. |
| [Function tags](https://www.javatpoint.com/jstl-function-tags) | The functions tags provide support for string manipulation and string length. The URL for the functions tags is **http://java.sun.com/jsp/jstl/functions** and prefix is **fn**. |
| [Formatting tags](https://www.javatpoint.com/jstl-formatting-tags) | The Formatting tags provide support for message formatting, number and date formatting, etc. The URL for the Formatting tags is **http://java.sun.com/jsp/jstl/fmt** and prefix is **fmt**. |
| [XML tags](https://www.javatpoint.com/jstl-xml-tags) | The XML tags provide flow control, transformation, etc. The URL for the XML tags is **http://java.sun.com/jsp/jstl/xml** and prefix is **x**. |
| [SQL tags](https://www.javatpoint.com/jstl-sql-tags) | The JSTL SQL tags provide SQL support. The URL for the SQL tags is **http://java.sun.com/jsp/jstl/sql** and prefix is **sql**. |

#### **For creating JSTL application, you need to load the jstl.jar file**

SOME OF THE TAGS:

1.CORE TAG: c:out,c:import,c:set,c:remove,c:catch,c:foreach,c:url,c:if,c:when,c:redirect

2.FUNCTION TAGS: fn:contains(),fn:endswith(),fn:indexof(),fn.split(),fn:length(),fn:replace.

3.FORMATTING TAGS: fmt:timezone,fmt:formatnumber,fmt:settimezone,fmt:message,fmt:formatdate

4.XML TAGS:x:out,x:set,x:when,x:if,x:choose

5.SQL TAGS:sql:setDataSource,sql:query,sql:update,sql:dateparam,sql:transaction.

1. Different types of creating Sessions ?

**Getting or Creating a Session**

By default, a session is automatically created when the user visits the website. To obtain the HttpSession object representing the user’s session, invoke the getSession() method of the HttpServletRequest interface in doGet() or doPost() method of a Java Servlet.

Note that the HttpServletRequest .getSession() method returns the current session associated with the request, or create a new one if the request doesn’t have a session. That means the returned HttpSession object is always not null.

To get a session without creating a new one if not exist, you can use invoke getSession(false) on the HttpServletRequest:

In this case, the returned value can be null if no session exists – hence the if-else check for null ability is needed. That also means getSession() is equivalent to getSession(true).

1. Explain about request dispatcher ?

The RequestDispatcher interface provides the facility of dispatching the request to another resource it may be html, servlet or jsp. This interface can also be used to include the content of another resource also. It is one of the way of servlet collaboration.

There are two methods defined in the RequestDispatcher interface.

**Methods of RequestDispatcher interface**

The RequestDispatcher interface provides two methods. They are:

1. **public void forward(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException:**Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.



As you see in the above figure, response of second servlet is sent to the client. Response of the first servlet is not displayed to the user.

1. **public void include(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException:**Includes the content of a resource (servlet, JSP page, or HTML file) in the response.



As you can see in the above figure, response of second servlet is included in the response of the first servlet that is being sent to the client.

### **How to get the object of RequestDispatcher**

The getRequestDispatcher() method of ServletRequest interface returns the object of RequestDispatcher. Syntax:

#### **Syntax of getRequestDispatcher method**

**public** RequestDispatcher getRequestDispatcher(String resource);

### **Example of RequestDispatcher interface**

In this example, we are validating the password entered by the user. If password is servlet, it will forward the request to the WelcomeServlet, otherwise will show an error message: sorry username or password error!.  In this program, we are cheking for hardcoded information. But you can check it to the database also that we will see in the development chapter. In this example, we have created following files:

* **index.html file:** for getting input from the user.
* **Login.java file:** a servlet class for processing the response. If password is servlet, it will forward the request to the welcome servlet.
* **WelcomeServlet.java file:** a servlet class for displaying the welcome message.
* **web.xml file:** a deployment descriptor file that contains the information about the servlet.



1. Write SQL queries ?
2. What are data types of SQL ?

Data types will defines what kind of content we can put inside the column.when we are designing a table every DB has got the data type which are predefined.

TINY INT,INT,MEDIUM INT,SMALL INT,CHAR,VARCHAR,BLOB(binary large object)

very large strings can be stored

FLOAT,DOUBLE,ENUM,DATE,TIME,TIMESTAMP,YEAR.

1. What are constrains ?

SQL constraints are **used to specify rules for the data in a table**. Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table.

(**Not null,unique,primary key,forigen key,enum,set)**

1. Explain about joins ? refer notes
2. Difference between application server and Web server ?

|  |  |
| --- | --- |
| APPLICATION SERVER | WEB SERVER |
| 1.if we want to built the dynamic applications we go with application server. | 1.if we want to built the static application we go with the web server. |
| 2.here we use technologies like servlet, jsp, springboot ,ejb | 2.here we use technologies like html,css,javascript etc |
| 3.application servers like tomcat,glassfish,jboss etc are used | 3.we use webservers like apache http server,IIS(internet information services by ibm),Nginsx. |

1. Explain about J unit ?

We use junit framework to perform unit testing.when testing done at code level we will use unit testing.we can run the class without main method with the help of junit framework.

In junit we have different annotations by using which we can perform testing.

[1.@test:it](mailto:1.@test:it) will checks whether all the code in the method is running successfully or not.if the code is running successfully it will return true if not it will return false.

[2.@BeforeEach:it](mailto:2.@BeforeEach:it) runs before every test.

[3.@AfterEach:it](mailto:3.@AfterEach:it) runs after every test.

[4.@BeforeAll:it](mailto:4.@BeforeAll:it) should be static .it should run before all the methods.it runs only once.

[5.@AfterAll:it](mailto:5.@AfterAll:it) also should be static.it should run after all methods.it runs only once.

* Annotations decides the order of the output.

1. How to create Spring boot project ? ( maven )

Refer notes

1. Explain about application. Properties , repository layer , Controller , view , service layers ? refer notes
2. Explain about Entity classes ? ( JPA , POJO )
3. What are the annotations used in springBoot . Explain each ?
4. Explain about Session factory and logic to implement it in project ?

sessionFactory().buildSessionFactory().configuration().configure()

1. Explain about Hibernate and advantages over JDBC ?(refer notes)
2. Explain about hibernate configuration and components of its object ?

Ans: **The Key components of Hibernate are:**

* Session: It is used to get a physical network with a database.
* Transaction: It represents the unit of work with a database.
* Query: It uses SQL and HQL string to retrieve the data from the database and create objects.

1. Explain the steps to develop a web application using hibernate?
2. What are core interfaces of hibernate ?

Ans: five core interfaces are used in just about every Hibernate application. using these interfaces you can store retrieve persistent objects and control transaction.

1**-Session interface** -this is the primary interface used by hibernate applications. The instance of this interface are lightweight and are inexpensive to create and destroy. Hibernate sessions are not thread-safe.

2-**SessionFactory interface**-this is a factory that delivers the session objects to hibernate application. Generally, there will be a single SessionFactory for the whole application and it will be shared among the application threads.

3-**Configuration interface**-this interface is used to configure and bootstrap hibernate. The instance of this interface is used by the application in order to specify the location of hibernate specific mapping documents.

4-**Transaction interface**-this is an optional interface but the above three interfaces are mandatory in each and every application. This interface abstracts the code from any kind of transaction implementation such as JDBC transaction.

5-**Query and criteria interface** -this interface allows the user to perform queries and also control the flow of the query execution.

1. What are the advantages of ORM over JDBC ?
2. Explain about spring ?

**Spring:** Spring Framework is the most popular application development framework of Java. The main feature of the Spring Framework is **dependency Injection** or **Inversion of Control** (IoC). With the help of Spring Framework, we can develop a **loosely** coupled application. It is better to use if application type or characteristics are purely defined.

**Spring Boot:** Spring Boot is a module of Spring Framework. It allows us to build a stand-alone application with minimal or zero configurations. It is better to use if we want to develop a simple Spring-based application or RESTful services.

1. Difference between spring and spring boot ?

|  |  |
| --- | --- |
| **Spring** | **Spring Boot** |
| **Spring Framework** is a widely used Java EE framework for building applications. | **Spring Boot Framework** is widely used to develop **REST APIs**. |
| It aims to simplify Java EE development that makes developers more productive. | It aims to shorten the code length and provide the easiest way to develop **Web Applications**. |
| The primary feature of the Spring Framework is **dependency injection**. | The primary feature of Spring Boot is **Autoconfiguration**. It automatically configures the classes based on the requirement. |
| It helps to make things simpler by allowing us to develop **loosely coupled** applications. | It helps to create a **stand-alone** application with less configuration. |
| The developer writes a lot of code (**boilerplate code**) to do the minimal task. | It **reduces** boilerplate code. |
| To test the Spring project, we need to set up the sever explicitly. | Spring Boot offers **embedded server** such as **Jetty** and **Tomcat**, etc. |
| It does not provide support for an in-memory database. | It offers several plugins for working with an embedded and **in-memory** database such as **H2**. |
| Developers manually define dependencies for the Spring project in **pom.xml**. | Spring Boot comes with the concept of **starter** in pom.xml file that internally takes care of downloading the dependencies **JARs** based on Spring Boot Requirement. |

1. Explain about spring bean ?
2. Explain about web services ?
3. Difference between REST and SOAP ?

|  |  |  |
| --- | --- | --- |
| **No.** | **SOAP** | **REST** |
| 1) | SOAP is a **protocol**. | REST is an **architectural style**. |
| 2) | SOAP stands for **Simple Object Access Protocol**. | REST stands for **REpresentational State Transfer**. |
| 3) | SOAP **can't use REST** because it is a protocol. | REST **can use SOAP** web services because it is a concept and can use any protocol like HTTP, SOAP. |
| 4) | SOAP **uses services interfaces to expose the business logic**. | REST **uses URI to expose business logic**. |
| 5) | **JAX-WS** is the java API for SOAP web services. | **JAX-RS** is the java API for RESTful web services. |
| 6) | SOAP **defines standards**to be strictly followed. | REST does not define too much standards like SOAP. |
| 7) | SOAP **requires more bandwidth** and resource than REST. | REST **requires less bandwidth** and resource than SOAP. |
| 8) | SOAP **defines its own security**. | RESTful web services **inherits security measures** from the underlying transport. |
| 9) | SOAP **permits XML** data format only. | REST **permits different** data format such as Plain text, HTML, XML, JSON etc. |
| 10) | SOAP is **less preferred** than REST. | REST **more preferred** than SOAP. |

1. Why we use xml files over JSON ?
2. Explain about micro services ?
3. Explain about POSTMAN ?
4. What are the errors in postman . Explain about each ?
5. Explain the methods of hibernate ?
6. What is http servlet?Explain HTTP methods ? ( get , put , patch , delete , post )

The Http Servlet class extends the Generic Servlet class and implements Serializable interface. It provides http specific methods such as do Get, do Post, doDelete, doPut etc.

1. **protected void doGet(HttpServletRequest req, HttpServletResponse res)** handles the GET request. It is invoked by the web container.
2. **protected void doPost(HttpServletRequest req, HttpServletResponse res)** handles the POST request. It is invoked by the web container.
3. **protected void doPut(HttpServletRequest req, HttpServletResponse res)** handles the PUT request. It is invoked by the web container.
4. **protected void doDelete(HttpServletRequest req, HttpServletResponse res)** handles the DELETE request. It is invoked by the web container.
5. What are hibernate annotations . Explain each ?

@entity,@table,@column,@id,@generatedValue

1. Difference between hibernate with annotations and hbm.xml files ? refer notes
2. Explain about **Agile** completely ?
3. What is dispatcher servlet , how it works ?
4. How will generate pdf in all the view ?
5. View resolver , Jack Son
6. How will you redirect to view ?**by @requestMapping and by using jsp directive tag**
7. What are object states in hibernate ?

Ans: 1.transient state

2.presistent state

3.detached state

1. Explain dilate hibernate properties ?

Ans: The dialect specifies the type of database used in **hibernate** so that **hibernate** generate appropriate type of SQL statements.

1. Explain about level 1 , level 2 cache ?

Ans: 1) The primary difference is that **the first level cache is maintained at the Session level while the second level cache is maintained at the SessionFactory level**. 2) The data stored in the first level cache is accessible to the only Session that maintains it, while the second level cache is accessible to all.

1. Explain about Session tracking system ?
2. Difference between save and persist method in hibernate ?

Ans: **The first difference between save and persist is there return type**. Similar to save method, persist also INSERT records into the database, but return type of persist is void while return type of save is Serializable Object.

1. What are @primary and @qualifier annotations ?
2. How to monitor more than one micro services ?by using server.port in application properties.
3. What are roll back and commit ?
4. How to handle exceptions in spring boot ?
5. How to authenticate application ?
6. @springBoot @cross origin @bean
7. Features of html and CSS ?
8. Difference between do get , do post ?
9. Types of dependency injection ?

1.method

2.constructor

3.field

1. @Component , @controller

@controller:it defines the controller layer in the spring boot.

*@Component* is an annotation that allows Spring to automatically detect our custom beans.

In other words, without having to write any explicit code, Spring will:

* Scan our application for classes annotated with *@Component*
* Instantiate them and inject any specified dependencies into them
* Inject them wherever needed

However, most developers prefer to use the more specialized stereotype annotations to serve this function.

1. @mapping annotations , @qualifier , @primary , @response body
2. Life cycle of spring bean ?
3. What is API ?