Q1. Explain JDK, JRE and JVM?

JDK vs JRE vs JVM

JDK	JRE	JVM
It stands for Java Development Kit.	It stands for Java Runtime Environment.	It stands for Java Virtual Machine.
It is the tool necessary to compile, document and package Java programs.	JRE refers to a runtime environment in which Java bytecode can be executed.	It is an abstract machine. It is a specification that provides a run-time environment in which Java bytecode can be executed.
It contains JRE + development tools.	It's an implementation of the JVM which physically exists.	JVM follows three notations: Specification, **Implementation,** and **Runtime Instance**.

Q2. Explain public static void main(String args[]) in Java.

main() in Java is the entry point for any Java program. It is always written as **public static void main(String[] args)**.

- **public**: Public is an access modifier, which is used to specify who can access this method. Public means that this Method will be accessible by any Class.
- **static**: It is a keyword in java which identifies it is class-based. main() is made static in Java so that it can be accessed without creating the instance of a Class. In case, main is not made static then the compiler will throw an error as **main**() is called by the JVM before any objects are made and only static methods can be directly invoked via the class.
- **void**: It is the return type of the method. Void defines the method which will not return any value.
- **main**: It is the name of the method which is searched by JVM as a starting point for an application with a particular signature only. It is the method where the main execution occurs.
- **String args[]**: It is the parameter passed to the main method.

Q3. Why Java is platform independent?

Java is called platform independent because of its byte codes which can run on any system irrespective of its underlying operating system.

Q4. Why Java is not 100% Object-oriented?

Java is not 100% Object-oriented because it makes use of eight primitive data types such as boolean, byte, char, int, float, double, long, short which are not objects.

Q5. What are wrapper classes in Java?

Wrapper classes convert the Java primitives into the reference types (objects). Every primitive data type has a class dedicated to it. These are known as wrapper classes because they “wrap” the primitive data type into an object of that class. Refer to the below image which displays different primitive type, wrapper class and constructor argument.

Q6. What are constructors in Java?

In Java, constructor refers to a block of code which is used to initialize an object. It must have the same name as that of the class. Also, it has no return type and it is automatically called when an object is created.

There are two types of constructors:

1. **Default Constructor:** In Java, a default constructor is the one which does not take any inputs. In other words, default constructors are the no argument constructors which will be created by default in case you no other constructor is defined by the user. Its main purpose is to initialize the instance variables with the default values. Also, it is majorly used for object creation.
2. **Parameterized Constructor:** The parameterized constructor in Java, is the constructor which is capable of initializing the instance variables with the provided values. In other words, the constructors which take the arguments are called parameterized constructors.

Q7. What is singleton class in Java and how can we make a class singleton?

Singleton class is a class whose only one instance can be created at any given time, in one JVM. A class can be made singleton by making its constructor private.

Q8. What is the difference between Array list and vector in Java?

ArrayList	Vector
Array List is not synchronized. Array List is fast as it's non-synchronized. If an element is inserted into the Array List, it increases its Array size by 50%. Array List does not define the increment size. Array List can only use Iterator for traversing an Array List.	Vector is synchronized. Vector is slow as it is thread safe. Vector defaults to doubling size of its array. Vector defines the increment size. Vector can use both Enumeration and Iterator for traversing.

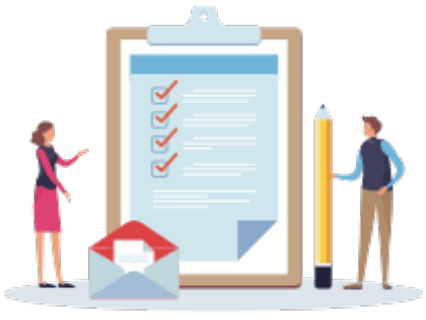
Q9. What is the difference between equals() and == in Java?

Equals() method is defined in Object class in Java and used for checking equality of two objects defined by business logic.

"==" or equality operator in Java is a binary operator provided by Java programming language and used to compare primitives and objects. *public boolean equals(Object o)* is the method provided by the Object class. The default implementation uses == operator to compare two objects. For example: method can be overridden like String class. equals() method is used to compare the values of two objects.

Q10. When can you use the super keyword?

In Java, the super keyword is a reference variable that refers to an immediate parent class object.



When you create a subclass instance, you're also creating an instance of the parent class, which is referenced to by the super reference variable.

The uses of the Java super Keyword are-

1.To refer to an immediate parent class instance variable, use super.

2.The keyword super can be used to call the method of an immediate parent class.

3.Super() can be used to call the constructor of the immediate parent class.

Q11. What makes a HashSet different from a TreeSet?

HashSet	TreeSet
It is implemented through a hash table.	TreeSet implements SortedSet Interface that uses trees for storing data.
It permits the null object.	It does not allow the null object.
It is faster than TreeSet especially for search, insert, and delete operations.	It is slower than HashSet for these operations.
It does not maintain elements in an ordered way.	The elements are maintained in a sorted order.
It uses equals() method to compare two objects.	It uses compareTo() method for comparing two objects.
It does not permit a heterogenous object.	It permits a heterogenous object.

Q12. What are the differences between HashMap and HashTable in Java?

HashMap	Hashtable
It is non synchronized. It cannot be shared between many threads without proper synchronization code.	It is synchronized. It is thread-safe and can be shared with many threads.
It permits one null key and multiple null values.	It does not permit any null key or value.
is a new class introduced in JDK 1.2.	It was present in earlier versions of java as well.
It is faster.	It is slower.
It is traversed through the iterator.	It is traversed through Enumerator and Iterator.
It uses fail fast iterator.	It uses an enumerator which is not fail fast.
It inherits AbstractMap class.	It inherits Dictionary class.

Q13. What is the importance of reflection in Java?

Reflection is a runtime API for inspecting and changing the behavior of methods, classes, and interfaces. Java Reflection is a powerful tool that can be really beneficial. Java Reflection allows you to analyze classes, interfaces, fields, and methods during runtime without knowing what they are called at compile time. Reflection can also be used to create new objects, call methods,

and get/set field values. External, user-defined classes can be used by creating instances of extensibility objects with their fully-qualified names. Debuggers can also use reflection to examine private members of classes.

Q14. How to not allow serialization of attributes of a class in Java?

The NonSerialized attribute can be used to prevent member variables from being serialized.
You should also make an object that potentially contains security-sensitive data nonserializable if possible. Apply the NonSerialized attribute to certain fields that store sensitive data if the object must be serialized. If you don't exclude these fields from serialisation, the data they store will be visible to any programmes with serialization permission.

Q15. Can you call a constructor of a class inside another constructor?

Yes, we can call a constructor of a class inside another constructor. This is also called as constructor chaining. Constructor chaining can be done in 2 ways-

1. **Within the same class:** For constructors in the same class, the this() keyword can be used.
2. **From the base class:** The super() keyword is used to call the constructor from the base class.
   The constructor chaining follows the process of inheritance. The constructor of the sub class first calls the constructor of the super class. Due to this, the creation of sub class's object starts with the initialization of the data members of the super class. The constructor chaining works similarly with any number of classes. Every constructor keeps calling the chain till the top of the chain.

Q16. Contiguous memory locations are usually used for storing actual values in an array but not in ArrayList. Explain.

An array generally contains elements of the primitive data types such as int, float, etc. In such cases, the array directly stores these elements at contiguous memory locations. While an ArrayList does not contain primitive data types. An arrayList contains the reference of the objects at different memory locations

instead of the object itself. That is why the objects are not stored at contiguous memory locations.

Q17. How is the creation of a String using new() different from that of a literal?

When we create a string using new(), a new object is created. Whereas, if we create a string using the string literal syntax, it may return an already existing object with the same name.

Q18. Why is synchronization necessary? Explain with the help of a relevant example.

Java allows multiple threads to execute. They may be accessing the same variable or object. Synchronization helps to execute threads one after another. It is important as it helps to execute all concurrent threads while being in sync. It prevents memory consistency errors due to access to shared memory. An example of synchronization code is-

```
public synchronized void
increment()
{
a++;
}
```

As we have synchronized this function, this thread can only use the object after the previous thread has used it.

Q19. Explain the term “Double Brace Initialization” in Java?

Double Brace Initialization is a Java term that refers to the combination of two independent processes. There are two braces used in this. The first brace creates an anonymous inner class. The second brace is an initialization block. When these both are used together, it is known as Double Brace Initialization. The inner class has a reference to the enclosing outer class, generally using the ‘this’ pointer. It is used to do both creation and initialization in a single

statement. It is generally used to initialize collections. It reduces the code and also makes it more readable.

Q20. Why is it said that the length() method of String class doesn't return accurate results?

The length() method of String class doesn't return accurate results because it simply takes into account the number of characters within in the String. In other words, code points outside of the BMP (Basic Multilingual Plane), that is, code points having a value of U+10000 or above, will be ignored.

The reason for this is historical. One of Java's original goals was to consider all text as Unicode; yet, Unicode did not define code points outside of the BMP at the time. It was too late to modify char by the time Unicode specified such code points.

Q21. What are the differences between Heap and Stack Memory in Java?

The major difference between Heap and Stack memory are:

Features	Stack	Heap
Memory	Stack memory is used only by one thread of execution.	Heap memory is used by all the parts of the application.
Access	Stack memory can't be accessed by other threads.	Objects stored in the heap are globally accessible.
Memory Management	Follows LIFO manner to free memory.	Memory management is based on the generation associated with each object.
Lifetime	Exists until the end of execution of the thread.	Heap memory lives from the start till the end of application execution.
Usage	Stack memory only contains local primitive and reference variables to objects in heap space.	Whenever an object is created, it's always stored in the Heap space.

Q22. What is a package in Java? List down various advantages of packages.

Packages in Java, are the collection of related classes and interfaces which are bundled together. By using packages, developers can easily modularize the code and optimize its reuse. Also, the code within the packages can be imported by other classes and reused. Below I have listed down a few of its advantages:

- •Packages help in avoiding name clashes
- •They provide easier access control on the code
- •Packages can also contain hidden classes which are not visible to the outer classes and only used within the package
- •Creates a proper hierarchical structure which makes it easier to locate the related classes

Q23. Why pointers are not used in Java?

Java doesn't use pointers because they are unsafe and increases the complexity of the program. Since, Java is known for its simplicity of code, adding the concept of pointers will be contradicting. Moreover, since JVM is responsible for implicit memory allocation, thus in order to avoid direct access to memory by the user, pointers are discouraged in Java.

Q24. What is JIT compiler in Java?

JIT stands for Just-In-Time compiler in Java. It is a program that helps in converting the Java bytecode into instructions that are sent directly to the processor. By default, the JIT compiler is enabled in Java and is activated whenever a Java method is invoked. The JIT compiler then compiles the bytecode of the invoked method into native machine code, compiling it "just in time" to execute. Once the method has been compiled, the JVM summons the compiled code of that method directly rather than interpreting it. This is why it is often responsible for the performance optimization of Java applications at the run time.

Q25. What are access modifiers in Java?

In Java, access modifiers are special keywords which are used to restrict the access of a class, constructor, data member and method in another class. Java supports four types of access modifiers:

1. Default
2. Private
3. Protected
4. Public

Modifier	Default	Private	Protected	Public
Same class	YES	YES	YES	YES
Same Package subclass	YES	NO	YES	YES
Same Package non-subclass	YES	NO	YES	YES
Different package subclass	NO	NO	YES	YES
Different package non-subclass	NO	NO	NO	YES

Q26. Define a Java Class.

A class in Java is a blueprint which includes all your data. A class contains fields (variables) and methods to describe the behavior of an object. Let's have a look at the syntax of a class.

```
class Abc {
member variables // class body
methods}
```

Q27. What is an object in Java and how is it created?

An object is a real-world entity that has a state and behavior. An object has three characteristics:

1. State
2. Behavior
3. Identity

An object is created using the 'new' keyword. For example:

ClassName obj = new ClassName();

Q28. What is Object Oriented Programming?

Object-oriented programming or popularly known as OOPs is a programming model or approach where the programs are organized around objects rather

than logic and functions. In other words, OOP mainly focuses on the objects that are required to be manipulated instead of logic. This approach is ideal for the programs large and complex codes and needs to be actively updated or maintained.

Q29. What are the main concepts of OOPs in Java?

Object-Oriented Programming or OOPs is a programming style that is associated with concepts like:

1. Inheritance: Inheritance is a process where one class acquires the properties of another.
2. Encapsulation: Encapsulation in Java is a mechanism of wrapping up the data and code together as a single unit.
3. Abstraction: Abstraction is the methodology of hiding the implementation details from the user and only providing the functionality to the users.
4. Polymorphism: Polymorphism is the ability of a variable, function or object to take multiple forms.

Q30. What is the difference between a local variable and an instance variable?

In Java, a **local variable** is typically used inside a method, constructor, or a **block** and has only local scope. Thus, this variable can be used only within the scope of a block. The best benefit of having a local variable is that other methods in the class won't be even aware of that variable.

Example

```
if(x > 100)
{
String test =
"Edureka";
}
```

Whereas, an **instance variable** in Java, is a variable which is bounded to its object itself. These variables are declared within a **class**, but outside a method. Every object of that class will create it's own copy of the variable while using it.

Thus, any changes made to the variable won't reflect in any other instances of that class and will be bound to that particular instance only.

```
class Test{
public String EmpName;
public int empAge;
}
```

Q31. Differentiate between the constructors and methods in Java?

Methods	Constructors
1. Used to represent the behavior of an object	1. Used to initialize the state of an object
2. Must have a return type	2. Do not have any return type
3. Needs to be invoked explicitly	3. Is invoked implicitly
4. No default method is provided by the compiler	4. A default constructor is provided by the compiler if the class has none
5. Method name may or may not be same as class name	5. Constructor name must always be the same as the class name

Q32. What is final keyword in Java?

final is a special keyword in Java that is used as a non-access modifier. A final variable can be used in different contexts such as:

- **final variable**

When the final keyword is used with a variable then its value can't be changed once assigned. In case the no value has been assigned to the final variable then using only the class constructor a value can be assigned to it.

- **final method**

When a method is declared final then it can't be overridden by the inheriting class.

- **final class**

When a class is declared as final in Java, it can’t be extended by any subclass class but it can extend other class.

Q33. What is the difference between break and continue statements?

break	continue
1. Can be used in switch and loop (for, while, do while) statements	1. Can be only used with loop statements
2. It causes the switch or loop statements to terminate the moment it is executed	2. It doesn’t terminate the loop but causes the loop to jump to the next iteration
3. It terminates the innermost enclosing loop or switch immediately	3. A continue within a loop nested with a switch will cause the next loop iteration to execute

Example break:

```
for (int i = 0; i < 5;
i++)</div>
<div>
<pre>{
if (i == 3)
{
break;
}
System.out.println(i);
}
```

Example continue:

```
for (int i = 0; i < 5; i+
+)
{
if(i == 2)
{
continue;
}
System.out.println(i);
}
```

Q34. What is an infinite loop in Java? Explain with an example.

An infinite loop is an instruction sequence in Java that loops endlessly when a functional exit isn’t met. This type of loop can be the result of a programming error or may also be a deliberate action based on the application behavior. An infinite loop will terminate automatically once the application exits.

For example:

```
public class InfiniteForLoopDemo
{
public static void main(String[] arg) {
for(;;)
System.out.println("Welcome to Edureka!");
// To terminate this program press ctrl + c in the
console.
}
}
```

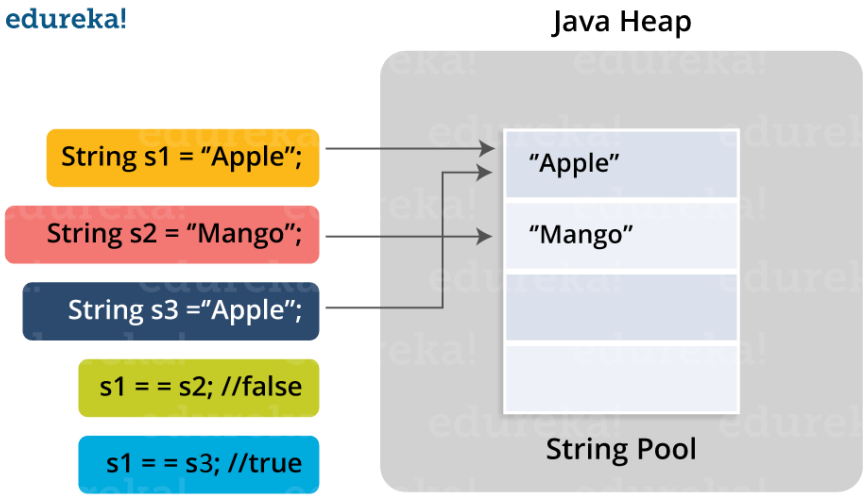
Q35. What is the difference between this() and super() in Java?

In Java, super() and this(), both are special keywords that are used to call the constructor.

this()	super()
1. this() represents the current instance of a class	1. super() represents the current instance of a parent/base class
2. Used to call the default constructor of the same class	2. Used to call the default constructor of the parent/base class
3. Used to access methods of the current class	3. Used to access methods of the base class
4. Used for pointing the current class instance	4. Used for pointing the superclass instance
5. Must be the first line of a block	5. Must be the first line of a block

Q36. What is Java String Pool?

Java String pool refers to a collection of Strings which are stored in heap memory. In this, whenever a new object is created, String pool first checks whether the object is already present in the pool or not. If it is present, then the same reference is returned to the variable else new object will be created in the String pool and the respective reference will be returned.



Q37. Differentiate between static and non-static methods in Java.

Static Method	Non-Static Method
1. *The static* keyword must be used before the method name	1. No need to use the *static* keyword before the method name
2. It is called using the class (className.methodName)	2. It is can be called like any general method
3. They can't access any non-static instance variables or methods	3. It can access any static method and any static variable without creating an instance of the class

Q38. Explain the term "Double Brace Initialisation" in Java?

Double Brace Initialization is a Java term that refers to the combination of two independent processes. There are two braces used in this. The first brace creates an anonymous inner class. The second brace is an initialization block. When these both are used together, it is known as Double Brace Initialisation. The inner class has a reference to the enclosing outer class, genertally using the 'this' pointer. It is used to do both creation and initialization in a single statement. It is generally used to initialize collections. It reduces the code and also makes it more readable.

Q39. What is constructor chaining in Java?

In Java, constructor chaining is the process of calling one constructor from another with respect to the current object. Constructor chaining is possible only through legacy where a subclass constructor is responsible for invoking the superclass' constructor first. There could be any number of classes in the constructor chain. Constructor chaining can be achieved in two ways:

1. Within the same class using this()
2. From base class using super()

Q40. Difference between String, StringBuilder, and StringBuffer.

Factor	String	StringBuilder	StringBuffer
Storage Area	Constant String Pool	Heap Area	Heap Area
Mutability	Immutable	Mutable	Mutable
Thread Safety	Yes	No	Yes
Performance	Fast	More efficient	Less efficient

Q41. What is a classloader in Java?

The **Java ClassLoader** is a subset of JVM (Java Virtual Machine) that is responsible for loading the class files. Whenever a Java program is executed it is first loaded by the classloader. Java provides three built-in classloaders:

1. Bootstrap ClassLoader
2. Extension ClassLoader
3. System/Application ClassLoader

Q42. Why Java Strings are immutable in nature?

In Java, string objects are immutable in nature which simply means once the String object is created its state cannot be modified. Whenever you try to update the value of that object instead of updating the values of that particular object, Java creates a new string object. Java String objects are immutable as String objects are generally cached in the String pool. Since String literals are usually shared between multiple clients, action from one client might affect the rest. It enhances security, caching, synchronization, and performance of the application.

Q43. What is the difference between an array and an array list?

Array	ArrayList
Cannot contain values of different data types	Can contain values of different data types.
Size must be defined at the time of declaration	Size can be dynamically changed
Need to specify the index in order to add data	No need to specify the index
Arrays are not type parameterized	Arraylists are type
Arrays can contain primitive data types as well as objects	Arraylists can contain only objects, no primitive data types are allowed

Q44. What is a Map in Java?

In Java, Map is an interface of Util package which maps unique keys to values. The Map interface is not a subset of the main Collection interface and thus it behaves little different from the other collection types. Below are a few of the characteristics of Map interface:

1. Map doesn’t contain duplicate keys.
2. Each key can map at max one value.

Q45. What is collection class in Java? List down its methods and interfaces.

In Java, the collection is a framework that acts as an architecture for storing and manipulating a group of objects. Using Collections you can perform various tasks like searching, sorting, insertion, manipulation, deletion, etc. Java collection framework includes the following:

- Interfaces
- Classes
- Methods

The below image shows the complete hierarchy of the Java Collection.

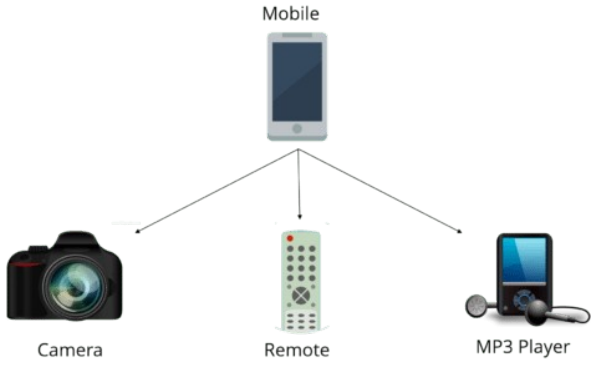
OOPS Java Interview Questions

Q1. What is Polymorphism?

Polymorphism is briefly described as “one interface, many implementations”. Polymorphism is a characteristic of being able to assign a different meaning or usage to something in different contexts – specifically, to allow an entity such as a variable, a function, or an object to have more than one form. There are two types of polymorphism:

1. Compile time polymorphism
2. Run time polymorphism

Compile time polymorphism is method overloading whereas Runtime time polymorphism is done using inheritance and interface.



Q2. What is runtime polymorphism or dynamic method dispatch?

In Java, runtime polymorphism or dynamic method dispatch is a process in which a call to an overridden method is resolved at runtime rather than at compile-time. In this process, an overridden method is called through the reference variable of a superclass. Let’s take a look at the example below to understand it better.

```
class Car {
void run()
{
System.out.println(&ldquo;car is running&rdquo;);
}
}
class Audi extends Car {
void run()
{
System.out.prinltn(&ldquo;Audi is running safely with
100km&rdquo;);
}
public static void main(String args[])
{
Car b= new Audi();    //upcasting
b.run();
}
}
```

Q3. What is abstraction in Java?

Abstraction refers to the quality of dealing with ideas rather than events. It basically deals with hiding the details and showing the essential things to the user. Thus you can say that abstraction in Java is the process of hiding the implementation details from the user and revealing only the functionality to them. Abstraction can be achieved in two ways:

1. **Abstract Classes** (0-100% of abstraction can be achieved)
2. **Interfaces** (100% of abstraction can be achieved)

Q4. What do you mean by an interface in Java?

An interface in Java is a blueprint of a class or you can say it is a collection of abstract methods and static constants. In an interface, each method is public and abstract but it does not contain any constructor. Thus, interface basically is a group of related methods with empty bodies. Example:

```
public interface Animal {
  public void eat();
  public void sleep();
  public void run();
}
```

Q5. What is the difference between abstract classes and interfaces?

Abstract Class	Interfaces
An abstract class can provide complete, default code and/or just the details that have to be overridden	An interface cannot provide any code at all, just the signature
In the case of an abstract class, a class may extend only one abstract class	A Class may implement several interfaces
An abstract class can have non-abstract methods	All methods of an Interface are abstract
An abstract class can have instance variables	An Interface cannot have instance variables
An abstract class can have any visibility: public, private, protected	An Interface visibility must be public (or) none
If we add a new method to an abstract class then we have the option of providing default implementation and therefore all the existing code might work properly	If we add a new method to an Interface then we have to track down all the implementations of the interface and define implementation for the new method
An abstract class can contain constructors	An Interface cannot contain constructors
Abstract classes are fast	Interfaces are slow as it requires extra indirection to find the corresponding method in the actual class

Q6. What is inheritance in Java?

Inheritance in Java is the concept where the properties of one class can be inherited by the other. It helps to reuse the code and establish a relationship between different classes. Inheritance is performed between two types of classes:

1. Parent class (Super or Base class)
2. Child class (Subclass or Derived class)

A class which inherits the properties is known as Child Class whereas a class whose properties are inherited is known as Parent class.

Q7. What are the different types of inheritance in Java?

Java supports four types of inheritance which are:

1. **Single Inheritance:** In single inheritance, one class inherits the properties of another i.e there will be only one parent as well as one child class.
2. **Multilevel Inheritance:** When a class is derived from a class which is also derived from another class, i.e. a class having more than one parent class but at different levels, such type of inheritance is called Multilevel Inheritance.
3. **Hierarchical Inheritance:** When a class has more than one child classes (subclasses) or in other words, more than one child classes have the same parent class, then such kind of inheritance is known as hierarchical.
4. **Hybrid Inheritance:** Hybrid inheritance is a combination of two *or more types* of inheritance.

Q8. What is method overloading and method overriding?

Method Overloading :

- In Method Overloading, Methods of the same class shares the same name but each method must have a different number of parameters or parameters having different types and order.
- Method Overloading is to “add” or “extend” more to the method’s behavior.
- It is a compile-time polymorphism.
- The methods must have a different signature.
- It may or may not need inheritance in Method Overloading.

Let’s take a look at the example below to understand it better.

```
class Adder {
Static int add(int a, int b)
{
return a+b;
}
Static double add( double a, double b)
{
return a+b;
}
```

```
public static void main(String args[])
{
System.out.println(Adder.add(11,11));
System.out.println(Adder.add(12.3,12.6))
;
}}
```

Method Overriding:

- In Method Overriding, the subclass has the same method with the same name and exactly the same number and type of parameters and same return type as a superclass.
- Method Overriding is to “Change” existing behavior of the method.
- It is a run time polymorphism.
- The methods must have the same signature.
- It always requires inheritance in Method Overriding.

Let’s take a look at the example below to understand it better.

```
class Car {
void run(){
System.out.println(&ldquo;car is running&rdquo;);
}
Class Audi extends Car{
void run()
{
System.out.prinltn("Audi is running safely with
100km");
}
public static void main( String args[])
{
Car b=new Audi();
b.run();
}
}
```

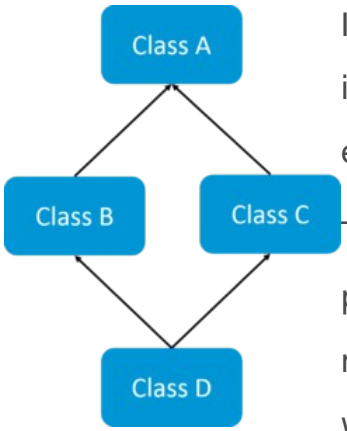
Q9. Can you override a private or static method in Java?

You cannot override a private or static method in Java. If you create a similar method with the same return type and same method arguments in child class then it will hide the superclass method; this is known as method hiding. Similarly, you cannot override a private method in subclass because it’s not accessible there. What you can do is create another private method with the

same name in the child class. Let's take a look at the example below to understand it better.

```
class Base {
private static void display() {
System.out.println("Static or class method from Base");
}
public void print() {
System.out.println("Non-static or instance method from
```

Q10. What is multiple inheritance? Is it supported by Java?



If a child class inherits the property from multiple classes is known as multiple inheritance. Java does not allow to extend multiple classes.

The problem with multiple inheritance is that if multiple parent classes have the same method name, then at runtime it becomes difficult for the compiler to decide which method to execute from the child class.

Therefore, Java doesn't support multiple inheritance. The problem is commonly referred to as Diamond Problem.

Q11. What is encapsulation in Java?

Encapsulation is a mechanism where you bind your data(variables) and code(methods) together as a single unit. Here, the data is hidden from the outer world and can be accessed only via current class methods. This helps in protecting the data from any unnecessary modification. We can achieve encapsulation in Java by:

- Declaring the variables of a class as private.
- Providing public setter and getter methods to modify and view the values of the variables.

Q12. What is an association?

Association is a relationship where all object have their own lifecycle and there is no owner. Let's take the example of Teacher and Student. Multiple students can associate with a single teacher and a single student can associate with multiple teachers but there is no ownership between the objects and both have

their own lifecycle. These relationships can be one to one, one to many, many to one and many to many.

Q13. What do you mean by aggregation?

An aggregation is a specialized form of Association where all object has their own lifecycle but there is ownership and child object can not belong to another parent object. Let's take an example of Department and teacher. A single teacher can not belong to multiple departments, but if we delete the department teacher object will not destroy.

Q14. What is composition in Java?

Composition is again a specialized form of Aggregation and we can call this as a "death" relationship. It is a strong type of Aggregation. Child object does not have their lifecycle and if parent object deletes all child object will also be deleted. Let's take again an example of a relationship between House and rooms. House can contain multiple rooms there is no independent life of room and any room can not belongs to two different houses if we delete the house room will automatically delete.

Q15. What is a marker interface?

A Marker interface can be defined as the interface having no data member and member functions. In simpler terms, an empty interface is called the Marker interface. The most common examples of Marker interface in Java are Serializable, Cloneable etc. The marker interface can be declared as follows.

```
1public interface Serializable{
2}
```

Q16. What is object cloning in Java?

Object cloning in Java is the process of creating an exact copy of an object. It basically means the ability to create an object with a similar state as the original object. To achieve this, Java provides a method **clone()** to make use of this functionality. This method creates a new instance of the class of the current object and then initializes all its fields with the exact same contents of

corresponding fields. To object clone(), the marker interface **java.lang.Cloneable** must be implemented to avoid any runtime exceptions. One thing you must note is Object clone() is a protected method, thus you need to override it.

Q17. What is a copy constructor in Java?

Copy constructor is a member function that is used to initialize an object using another object of the same class. Though there is no need for copy constructor in Java since all objects are passed by reference. Moreover, Java does not even support automatic pass-by-value.

Q18. What is a constructor overloading in Java?

In Java, constructor overloading is a technique of adding any number of constructors to a class each having a different parameter list. The compiler uses the number of parameters and their types in the list to differentiate the overloaded constructors.

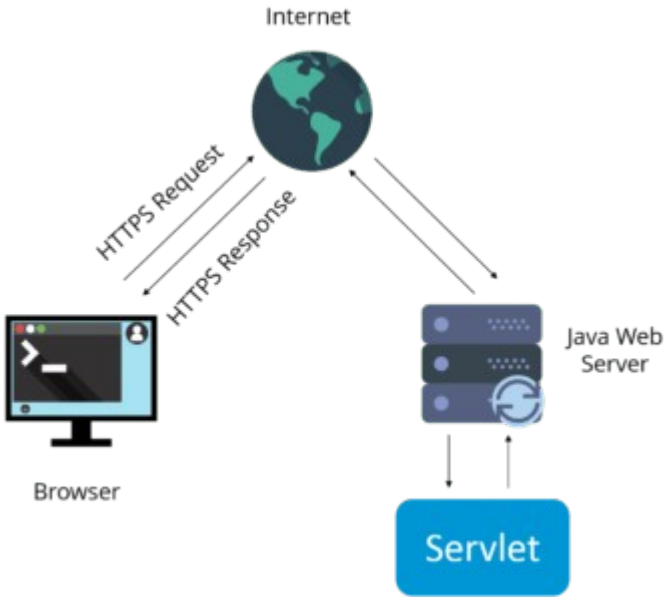
```
class Demo
{
int i;
public Demo(int a)
{
i=k;
}
public Demo(int a, int
b)
{
//body
}
}
```

Servlets – Java Interview Questions

Q1. What is a servlet?

- Java Servlet is server-side technologies to extend the capability of web servers by providing support for dynamic response and data persistence.
- The javax.servlet and javax.servlet.http packages provide interfaces and classes for writing our own servlets.

- All servlets must implement the javax.servlet.Servlet interface, which defines servlet lifecycle methods. When implementing a generic service, we can extend the GenericServlet class provided with the Java Servlet API. The HttpServlet class provides methods, such as doGet() and doPost(), for handling HTTP-specific services.
- Most of the times, web applications are accessed using HTTP protocol and thats why we mostly extend HttpServlet class. Servlet API hierarchy is shown in below image.



Q2. What are the differences between Get and Post methods?

Get	Post
Limited amount of data can be sent because data is sent in header.	Large amount of data can be sent because data is sent in body.
Not Secured because data is exposed in URL bar.	Secured because data is not exposed in URL bar.
Can be bookmarked	Cannot be bookmarked
Idempotent	Non-Idempotent
It is more efficient and used than Post	It is less efficient and used

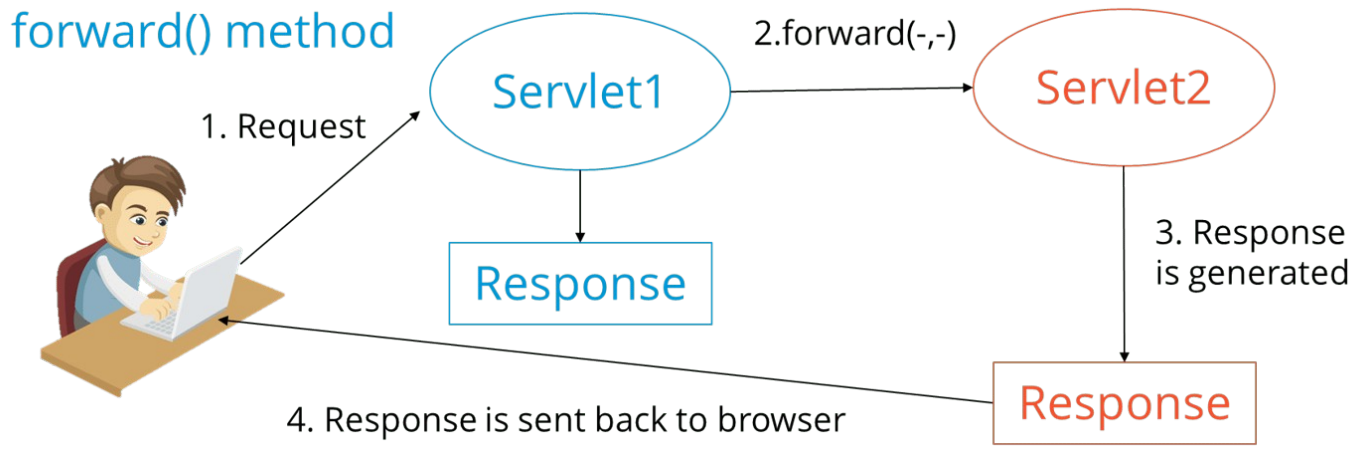
Q3. What is Request Dispatcher?

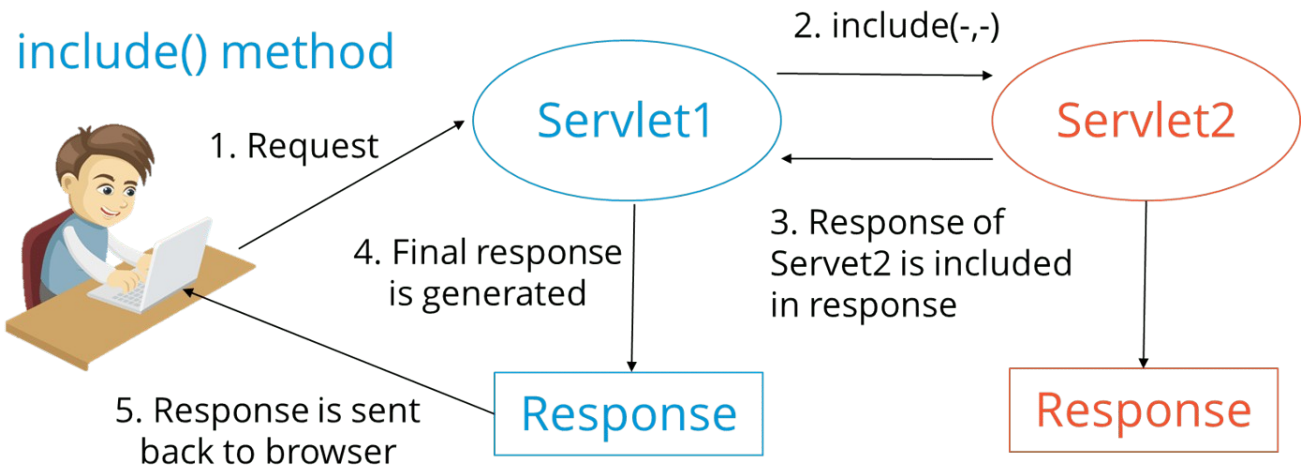
RequestDispatcher interface is used to forward the request to another resource that can be HTML, JSP or another servlet in same application. We can also use this to include the content of another resource to the response.

There are two methods defined in this interface:

1.void forward()

2.void include()





Q4. What are the differences between forward() method and sendRedirect() methods?

forward() method	SendRedirect() method
forward() sends the same request to another resource.	sendRedirect() method sends new request always because it uses the URL bar of the browser.
forward() method works at server side.	sendRedirect() method works at client side.
forward() method works within the server only.	sendRedirect() method works within and outside the server.

Q5. What is the life-cycle of a servlet?

There are 5 stages in the lifecycle of a servlet:

1. Servlet is loaded
2. Servlet is instantiated
3. Servlet is initialized
4. Service the request
5. Servlet is destroyed

Q6. How does cookies work in Servlets?

- Cookies are text data sent by server to the client and it gets saved at the client local machine.
- Servlet API provides cookies support through javax.servlet.http.Cookie class that implements Serializable and Cloneable interfaces.
- HttpServletRequest getCookies() method is provided to get the array of Cookies from request, since there is no point of adding Cookie to request, there are no methods to set or add cookie to request.
- Similarly HttpServletResponse addCookie(Cookie c) method is provided to attach cookie in response header, there are no getter methods for cookie.

Q7. What are the differences between ServletContext vs ServletConfig?

The difference between ServletContext and ServletConfig in Servlets JSP is in below tabular format.

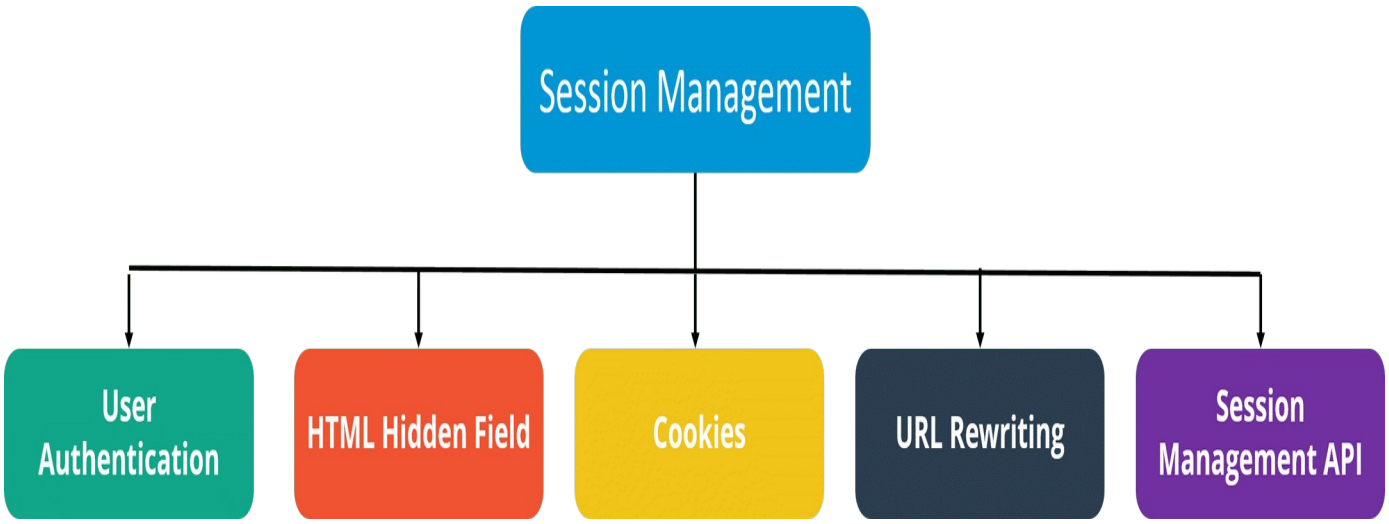
ServletConfig	ServletContext
Servlet config object represent single servlet	It represent whole web application running on particular JVM and common for all the servlet
Its like local parameter associated with particular servlet	Its like global parameter associated with whole application
It's a name value pair defined inside the servlet section of web.xml file so it has servlet wide scope	ServletContext has application wide scope so define outside of servlet tag in web.xml file.
getServletConfig() method is used to get the config object	getServletContext() method is used to get the context object.
for example shopping cart of a user is a specific to particular user so here we can use servlet config	To get the MIME type of a file or application session related information is stored using servlet context object.

Q8. What are the different methods of session management in servlets?

Session is a conversational state between client and server and it can consists of multiple request and response between client and server. Since HTTP and Web Server both are stateless, the only way to maintain a session is when some unique information about the session (session id) is passed between server and client in every request and response.

Some of the common ways of session management in servlets are:

1. User Authentication
2. HTML Hidden Field
3. Cookies
4. URL Rewriting
5. Session Management API



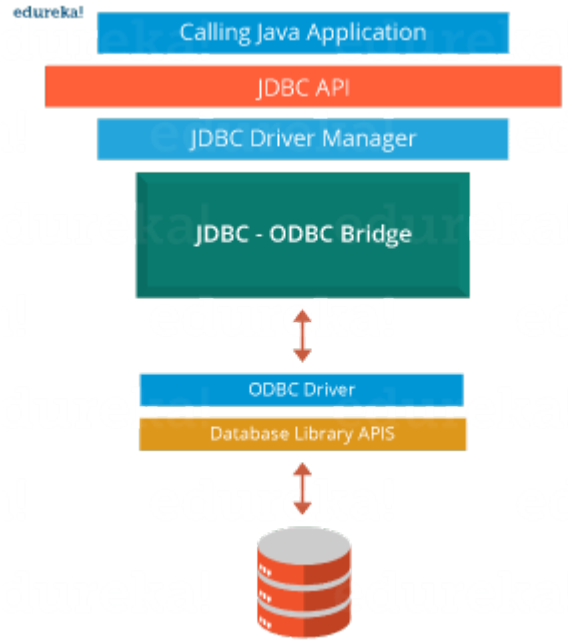
Apart from this blog, if you want to get trained by professionals on this technology, you can opt for structured training from edureka! Click below to know more.

JDBC – Java Interview Questions

1. What is JDBC Driver?

JDBC Driver is a software component that enables java application to interact with the database. There are 4 types of JDBC drivers:

1. JDBC-ODBC bridge driver
2. Native-API driver (partially java driver)
3. Network Protocol driver (fully java driver)
4. Thin driver (fully java driver)



2. What are the steps to connect to a database in java?

- Registering the driver class
- Creating connection
- Creating statement
- Executing queries
- Closing connection

3. What are the JDBC API components?

The java.sql package contains interfaces and classes for JDBC API.

Interfaces:

- Connection
- Statement
- PreparedStatement
- ResultSet
- ResultSetMetaData
- DatabaseMetaData
- CallableStatement etc.

Classes:

- DriverManager
- Blob
- Clob

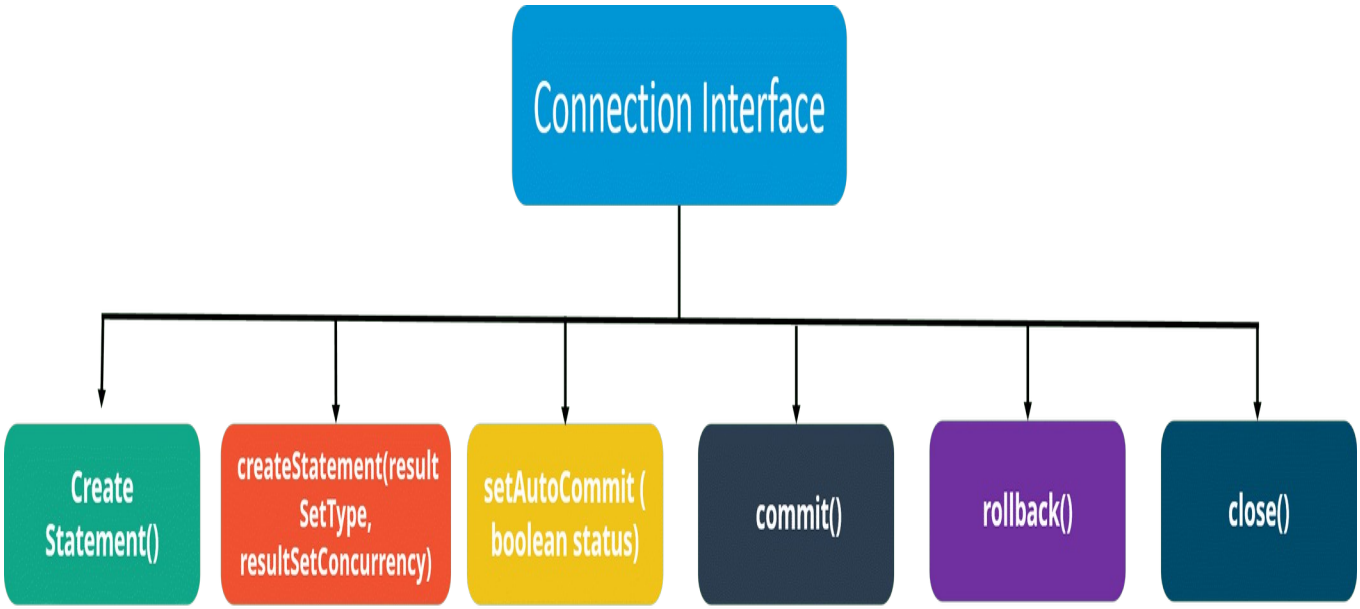
- Types
- SQLException etc.

4. What is the role of JDBC DriverManager class?

The DriverManager class manages the registered drivers. It can be used to register and unregister drivers. It provides factory method that returns the instance of Connection.

5. What is JDBC Connection interface?

The Connection interface maintains a session with the database. It can be used for transaction management. It provides factory methods that returns the instance of Statement, PreparedStatement, CallableStatement and DatabaseMetaData.



6. What is the purpose of JDBC ResultSet interface?

The ResultSet object represents a row of a table. It can be used to change the cursor pointer and get the information from the database.

7. What is JDBC ResultSetMetaData interface?

The ResultSetMetaData interface returns the information of table such as total number of columns, column name, column type etc.

8. What is JDBC DatabaseMetaData interface?

The DatabaseMetaData interface returns the information of the database such as username, driver name, driver version, number of tables, number of views etc.

9. What do you mean by batch processing in JDBC?

Batch processing helps you to group related SQL statements into a batch and execute them instead of executing a single query. By using batch processing technique in JDBC, you can execute multiple queries which makes the performance faster.

10. What is the difference between execute, executeQuery, executeUpdate?

Statement ***execute(String query)*** is used to execute any SQL query and it returns TRUE if the result is an ResultSet such as running Select queries. The output is FALSE when there is no ResultSet object such as running Insert or Update queries. We can use *getResultSet()* to get the ResultSet and *getUpdateCount()* method to retrieve the update count.

Statement ***executeQuery(String query)*** is used to execute Select queries and returns the ResultSet. ResultSet returned is never null even if there are no records matching the query. When executing select queries we should use executeQuery method so that if someone tries to execute insert/update statement it will throw java.sql.SQLException with message “executeQuery method can not be used for update”.

Statement **executeUpdate(String query**) is used to execute Insert/Update/Delete (DML) statements or DDL statements that returns nothing. The output is int and equals to the row count for SQL Data Manipulation Language (DML) statements. For DDL statements, the output is 0.

You should use execute() method only when you are not sure about the type of statement else use executeQuery or executeUpdate method.

Q11. What do you understand by JDBC Statements?

JDBC statements are basically the statements which are used to send SQL commands to the database and retrieve data back from the database. Various methods like execute(), executeUpdate(), executeQuery, etc. are provided by JDBC to interact with the database.

JDBC supports 3 types of statements:

1. Statement: Used for general purpose access to the database and executes a static SQL query at runtime.
2. PreparedStatement: Used to provide input parameters to the query during execution.
3. CallableStatement: Used to access the database stored procedures and helps in accepting runtime parameters.

Spring Framework – Java Interview Questions

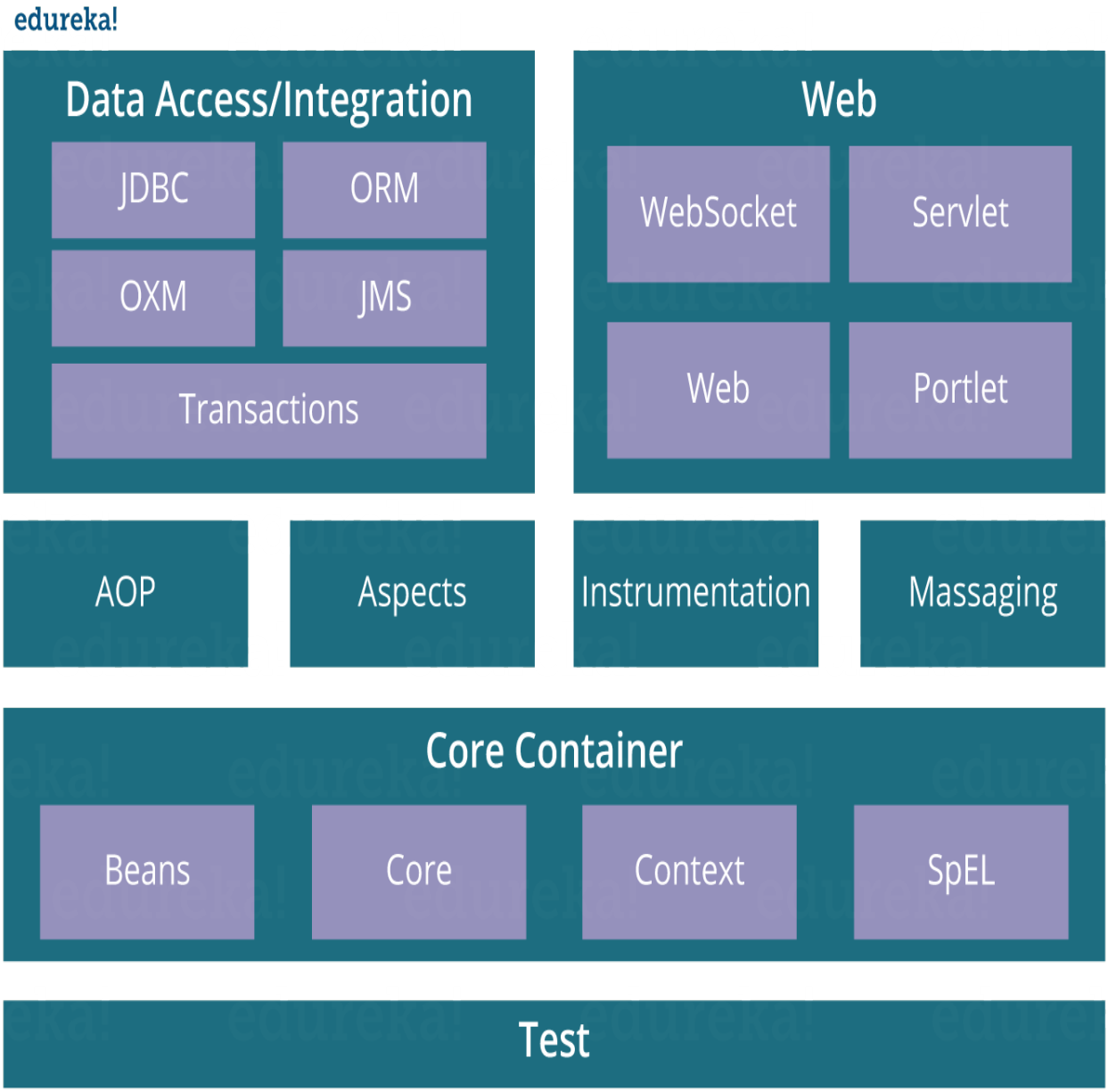
Q1. What is Spring?

Wikipedia defines the Spring framework as “an application framework and inversion of control container for the Java platform. The framework’s core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE platform.” Spring is essentially a lightweight, integrated framework that can be used for developing enterprise applications in java.

Q2. Name the different modules of the Spring framework.

Some of the important Spring Framework modules are:

- Spring Context – for dependency injection.
- Spring AOP – for aspect oriented programming.
- Spring DAO – for database operations using DAO pattern
- Spring JDBC – for JDBC and DataSource support.
- Spring ORM – for ORM tools support such as Hibernate
- Spring Web Module – for creating web applications.
- Spring MVC – Model-View-Controller implementation for creating web applications, web services etc.



Q3. List some of the important annotations in annotation-based Spring configuration.

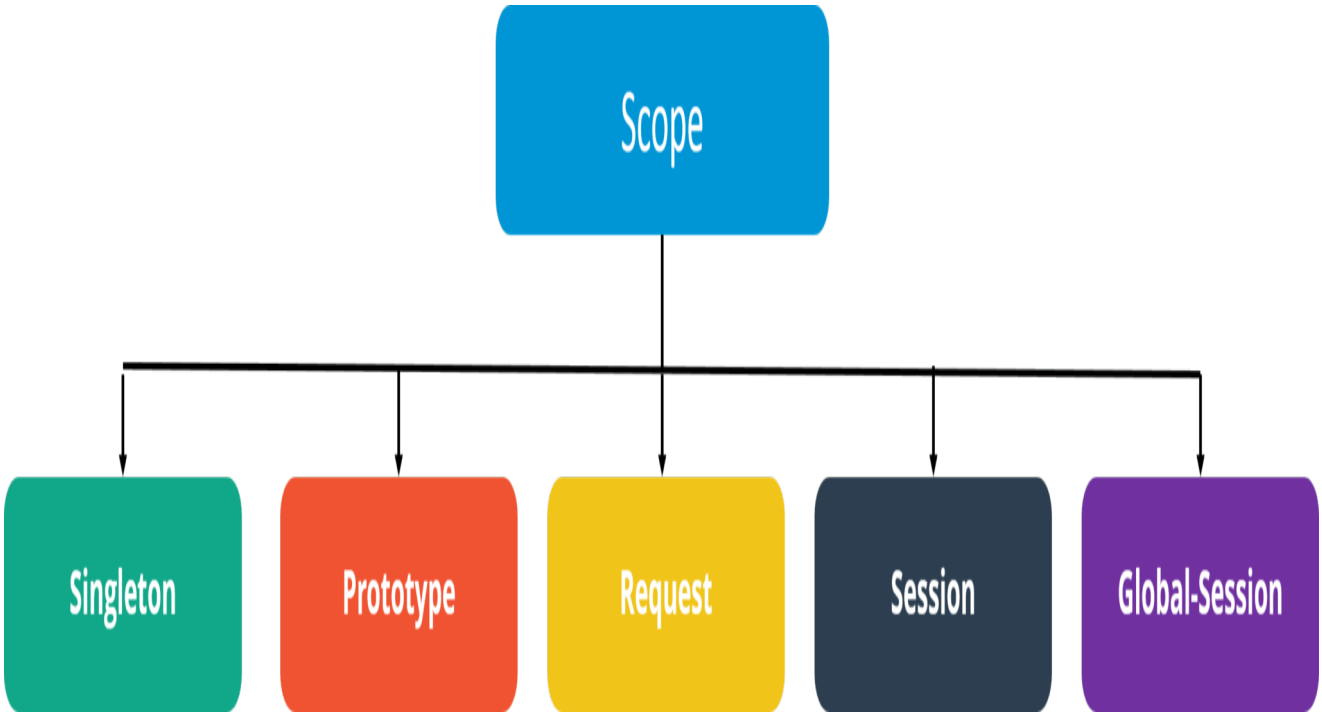
The important annotations are:

- @Required
- @Autowired
- @Qualifier
- @Resource
- @PostConstruct
- @PreDestroy

Q4. Explain Bean in Spring and List the different Scopes of Spring bean.

Beans are objects that form the backbone of a Spring application. They are managed by the Spring IoC container. In other words, a bean is an object that is instantiated, assembled, and managed by a Spring IoC container.

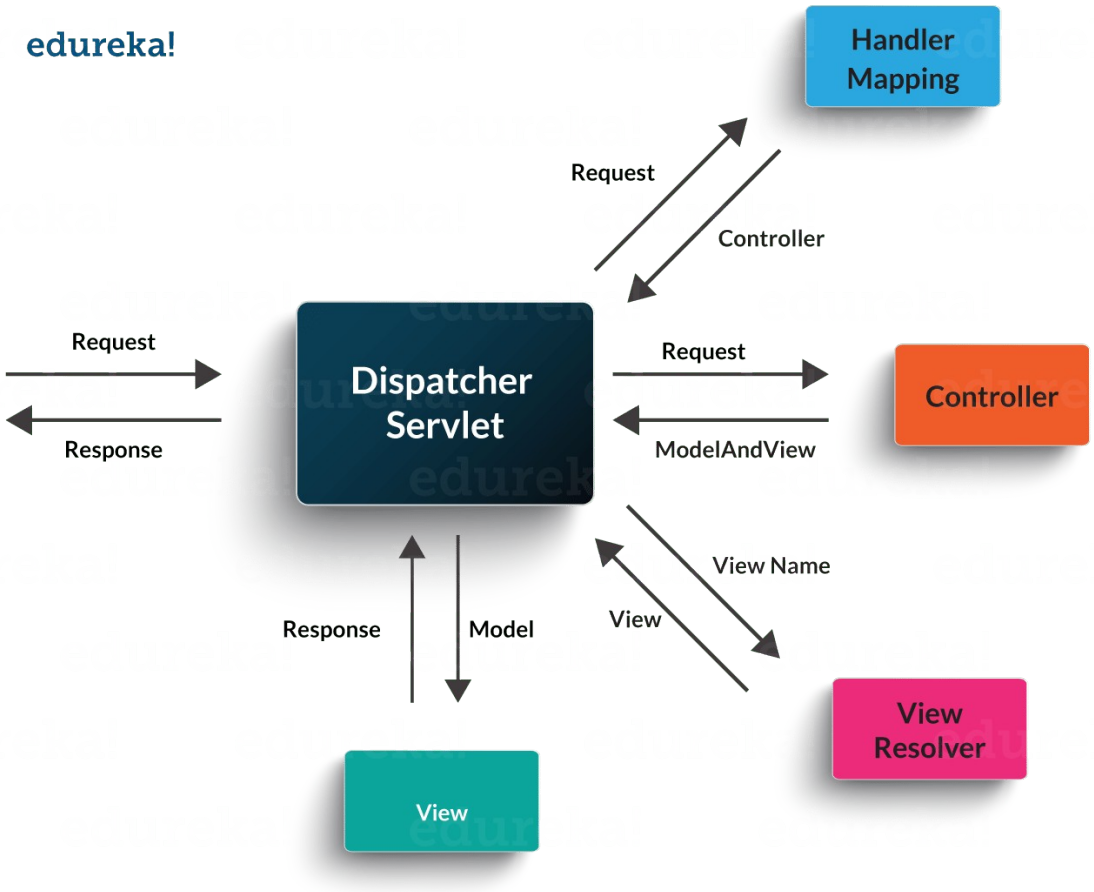
There are five Scopes defined in Spring beans.



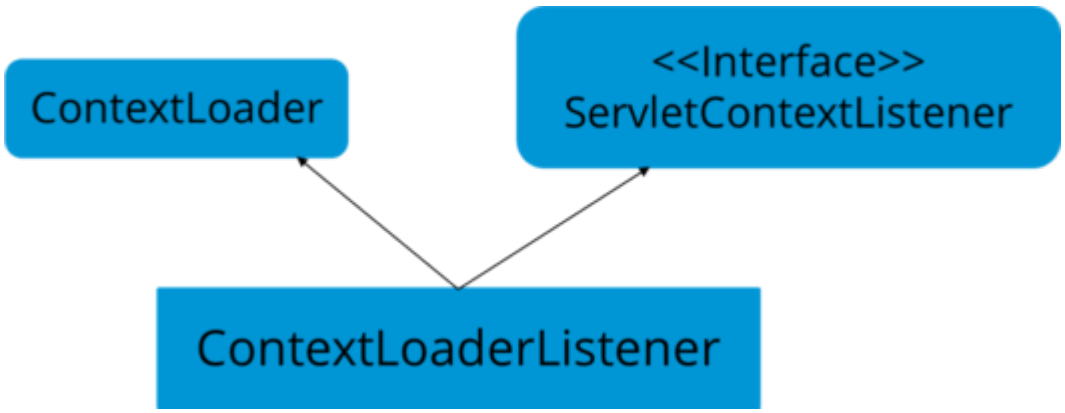
- **Singleton**: Only one instance of the bean will be created for each container. This is the default scope for the spring beans. While using this scope, make sure spring bean doesn't have shared instance variables otherwise it might lead to data inconsistency issues because it's not thread-safe.
- **Prototype**: A new instance will be created every time the bean is requested.
- **Request**: This is same as prototype scope, however it's meant to be used for web applications. A new instance of the bean will be created for each HTTP request.
- **Session**: A new bean will be created for each HTTP session by the container.
- **Global-session**: This is used to create global session beans for Portlet applications.

Q5. Explain the role of DispatcherServlet and ContextLoaderListener.

DispatcherServlet is basically the front controller in the Spring MVC application as it loads the spring bean configuration file and initializes all the beans that have been configured. If annotations are enabled, it also scans the packages to configure any bean annotated with @Component, @Controller, @Repository or @Service annotations.



ContextLoaderListener, on the other hand, is the listener to start up and shut down the WebApplicationContext in Spring root. Some of its important functions includes tying up the lifecycle of Application Context to the lifecycle of the ServletContext and automating the creation of ApplicationContext.



Q6. What are the differences between constructor injection and setter injection?

No.	Constructor Injection	Setter Injection
1)	No Partial Injection	Partial Injection
2)	Doesn't override the setter property	Overrides the constructor property if both are defined.
3)	Creates a new instance if any modification occurs	Doesn't create a new instance if you change the property value
4)	Better for too many properties	Better for a few properties.

Q7. What is autowiring in Spring? What are the autowiring modes?

Autowiring enables the programmer to inject the bean automatically. We don't need to write explicit injection logic. Let's see the code to inject bean using dependency injection.

1. <bean
   id="emp" class="com.javatpoint.Employee" autowire="byName" />

The autowiring modes are given below:

No.	Mode	Description
1)	no	this is the default mode, it means autowiring is not enabled.
2)	byName	Injects the bean based on the property name. It uses setter method.
3)	byType	Injects the bean based on the property type. It uses setter method.
4)	constructor	It injects the bean using constructor

Q8. How to handle exceptions in Spring MVC Framework?

Spring MVC Framework provides the following ways to help us achieving robust exception handling.

Controller Based:

We can define exception handler methods in our controller classes. All we need is to annotate these methods with @ExceptionHandler annotation.

Global Exception Handler:

Exception Handling is a cross-cutting concern and Spring provides @ControllerAdvice annotation that we can use with any class to define our global exception handler.

HandlerExceptionResolver implementation:

For generic exceptions, most of the times we serve static pages. Spring Framework provides HandlerExceptionResolver interface that we can implement to create global exception handler. The reason behind this additional way to define global exception handler is that Spring framework also provides default implementation classes that we can define in our spring bean configuration file to get spring framework exception handling benefits.

Q9. What are some of the important Spring annotations which you have used?

Some of the Spring annotations that I have used in my project are:

@Controller – for controller classes in Spring MVC project.

@RequestMapping – for configuring URI mapping in controller handler methods. This is a very important annotation, so you should go through Spring MVC RequestMapping Annotation Examples

@ResponseBody – for sending Object as response, usually for sending XML or JSON data as response.

@PathVariable – for mapping dynamic values from the URI to handler method arguments.

@Autowired – for autowiring dependencies in spring beans.

@Qualifier – with @Autowired annotation to avoid confusion when multiple instances of bean type is present.

@Service – for service classes.

@Scope – for configuring the scope of the spring bean.

@Configuration, @ComponentScan and @Bean – for java based configurations.

AspectJ annotations for configuring aspects and advices , @Aspect, @Before, @After, @Around, @Pointcut, etc.

Q10. How to integrate Spring and Hibernate Frameworks?

We can use Spring ORM module to integrate Spring and Hibernate frameworks if you are using Hibernate 3+ where SessionFactory provides current session, then you should avoid using HibernateTemplate or HibernateDaoSupport classes and better to use DAO pattern with dependency injection for the integration.

Also, Spring ORM provides support for using Spring declarative transaction management, so you should utilize that rather than going for hibernate boiler-plate code for transaction management.

Q11. Name the types of transaction management that Spring supports.

Two types of transaction management are supported by Spring. They are:

1. **Programmatic transaction management:** In this, the transaction is managed with the help of programming. It provides you extreme flexibility, but it is very difficult to maintain.
2. **Declarative transaction management:** In this, transaction management is separated from the business code. Only annotations or XML based configurations are used to manage the transactions.

Apart from these Core Java interview questions for experienced professionals, if you want to get trained by professionals on this technology, you can opt for a structured training from edureka!

Hibernate – Java Interview Questions for Experienced Professionals

1. What is Hibernate Framework?

Object-relational mapping or ORM is the programming technique to map application domain model objects to the relational database tables. Hibernate

is Java-based ORM tool that provides a framework for mapping application domain objects to the relational database tables and vice versa.

Hibernate provides a reference implementation of Java Persistence API, that makes it a great choice as ORM tool with benefits of loose coupling. We can use the Hibernate persistence API for CRUD operations. Hibernate framework provide option to map plain old java objects to traditional database tables with the use of JPA annotations as well as XML based configuration.

Similarly, hibernate configurations are flexible and can be done from XML configuration file as well as programmatically.

2. What are the important benefits of using Hibernate Framework?

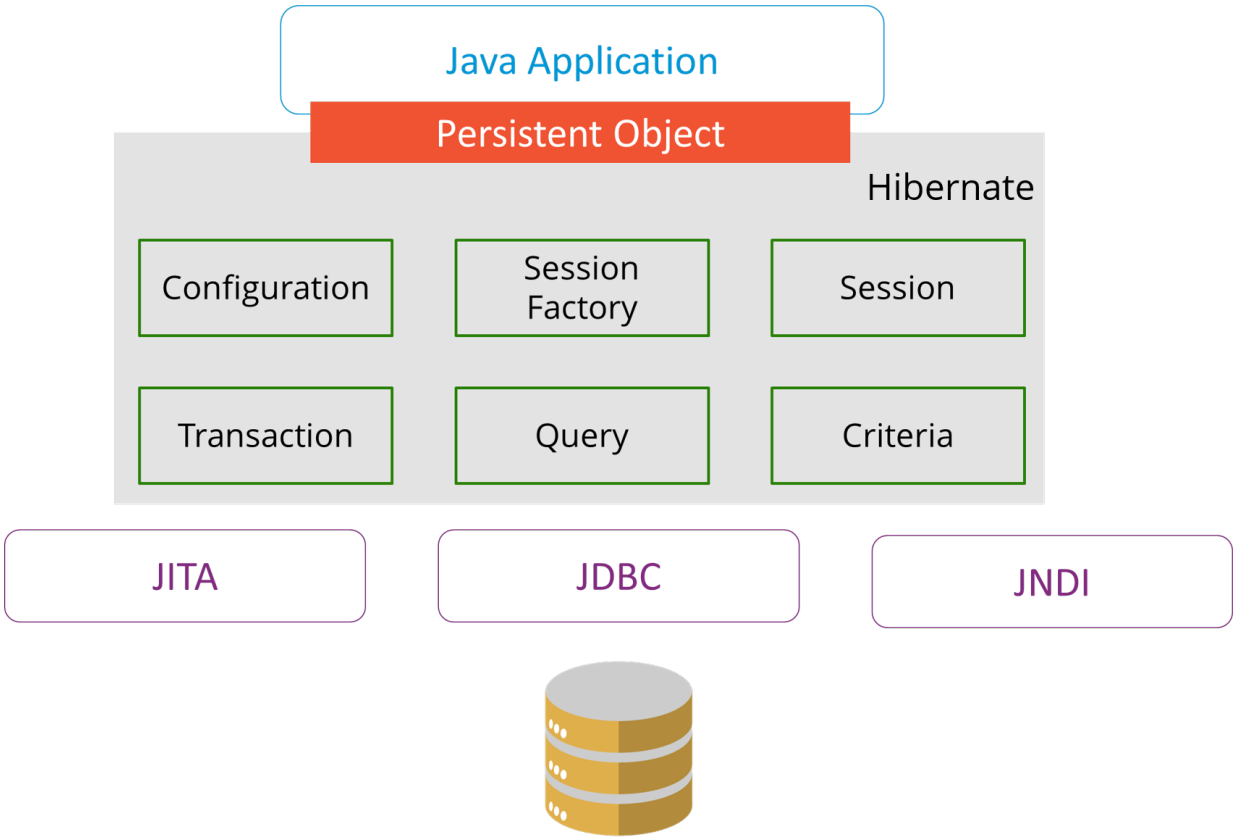
Some of the important benefits of using hibernate framework are:

1. Hibernate eliminates all the boiler-plate code that comes with JDBC and takes care of managing resources, so we can focus on business logic.
2. Hibernate framework provides support for XML as well as JPA annotations, that makes our code implementation independent.
3. Hibernate provides a powerful query language (HQL) that is similar to SQL. However, HQL is fully object-oriented and understands concepts like inheritance, polymorphism, and association.
4. Hibernate is an open source project from Red Hat Community and used worldwide. This makes it a better choice than others because learning curve is small and there are tons of online documentation and help is easily available in forums.
5. Hibernate is easy to integrate with other Java EE frameworks, it’s so popular that Spring Framework provides built-in support for integrating hibernate with Spring applications.
6. Hibernate supports lazy initialization using proxy objects and perform actual database queries only when it’s required.
7. Hibernate cache helps us in getting better performance.
8. For database vendor specific feature, hibernate is suitable because we can also execute native sql queries.

Overall hibernate is the best choice in current market for ORM tool, it contains all the features that you will ever need in an ORM tool.

3. Explain Hibernate architecture.

Hibernate has a layered architecture which helps the user to operate without having to know the underlying APIs. Hibernate makes use of the database and configuration data to provide persistence services (and persistent objects) to the application. It includes many objects such as persistent object, session factory, transaction factory, connection factory, session, transaction etc.



The Hibernate architecture is categorized in four layers.

- Java application layer
- Hibernate framework layer
- Backhand API layer
- Database layer

4. What are the differences between get and load methods?

The differences between get() and load() methods are given below.

No.	get()	load()
1)	Returns null if object is not found.	Throws ObjectNotFoundException if an object is not found.
2)	get() method always hit the database.	load() method doesn't hit the database.
3)	It returns a real object, not a proxy.	It returns a proxy object.
4)	It should be used if you are not sure about the existence of instance.	It should be used if you are sure that the instance exists.

5. What are the advantages of Hibernate over JDBC?

Some of the important advantages of Hibernate framework over JDBC are:

1. Hibernate removes a lot of boiler-plate code that comes with JDBC API, the code looks cleaner and readable.
2. Hibernate supports inheritance, associations, and collections. These features are not present with JDBC API.
3. Hibernate implicitly provides transaction management, in fact, most of the queries can't be executed outside transaction. In JDBC API, we need to write code for transaction management using commit and rollback.
4. JDBC API throws SQLException that is a checked exception, so we need to write a lot of try-catch block code. Most of the times it's redundant in every JDBC call and used for transaction management. Hibernate wraps JDBC exceptions and throw JDBCException or HibernateException un-checked exception, so we don't need to write code to handle it. Hibernate built-in transaction management removes the usage of try-catch blocks.
5. Hibernate Query Language (HQL) is more object-oriented and close to Java programming language. For JDBC, we need to write native SQL queries.
6. Hibernate supports caching that is better for performance, JDBC queries are not cached hence performance is low.
7. Hibernate provides option through which we can create database tables too, for JDBC tables must exist in the database.
8. Hibernate configuration helps us in using JDBC like connection as well as JNDI DataSource for the connection pool. This is a very important feature in enterprise application and completely missing in JDBC API.

9. Hibernate supports JPA annotations, so the code is independent of the implementation and easily replaceable with other ORM tools. JDBC code is very tightly coupled with the application.

JSP – Java Interview Questions

1. What are the life-cycle methods for a jsp?

Methods	Description
public void jspInit()	It is invoked only once, same as init method of servlet.
public void _jspService(ServletRequest request,ServletResponse)throws ServletException,IOException	It is invoked at each request, same as service() method of servlet.
public void jspDestroy()	It is invoked only once, same as destroy() method of servlet.

2. What are the JSP implicit objects?

JSP provides 9 implicit objects by default. They are as follows:

Object	Type
1) out	JspWriter
2) request	HttpServletRequest
3) response	HttpServletResponse
4) config	ServletConfig
5) session	HttpSession
6) application	ServletContext
7) pageContext	PageContext
8) page	Object
9) exception	Throwable

3. What are the differences between include directive and include action?

include directive	include action
The include directive includes the content at page translation time.	The include action includes the content at request time.
The include directive includes the original content of the page so page size increases at runtime.	The include action doesn't include the original content rather invokes the include() method of Vendor provided class.
It's better for static pages.	It's better for dynamic pages.

4. How to disable caching on back button of the browser?

```
<%
response.setHeader("Cache-Control","no-store");
response.setHeader("Pragma","no-cache");
response.setHeader ("Expires", "0");                    //prevents caching at the
```

proxy server

%>

5. What are the different tags provided in JSTL?

There are 5 type of JSTL tags.

1. core tags
2. sql tags
3. xml tags
4. internationalization tags
5. functions tags

6. How to disable session in JSP?

1. <%@ page session=“false” %>

7. How to delete a Cookie in a JSP?

The following code explains how to delete a Cookie in a JSP :

```
Cookie mycook = new Cookie("name1","value1");

response.addCookie(mycook1);

Cookie killmycook = new
Cookie("mycook1","value1");

killmycook . set MaxAge ( 0 );

killmycook . set Path ("/");

killmycook . addCookie ( killmycook 1 );
```

8. Explain the jspDestroy() method.

jspDestry() method is invoked from **javax.servlet.jsp.JspPage** interface whenever a JSP page is about to be destroyed. Servlets destroy methods can be easily overridden to perform cleanup, like when closing a database connection.

9. How is JSP better than Servlet technology?

JSP is a technology on the server’s side to make content generation simple. They are document-centric, whereas servlets are programs. A Java server page

can contain fragments of Java program, which execute and instantiate Java classes. However, they occur inside an HTML template file. It provides the framework for the development of a Web Application.

10. Why should we not configure JSP standard tags in web.xml?

We don’t need to configure JSP standard tags in web.xml because when container loads the web application and find TLD files, it automatically configures them to be used directly in the application JSP pages. We just need to include it in the JSP page using taglib directive.

11. How will you use JSP EL in order to get the HTTP method name?

Using pageContext JSP EL implicit object you can get the request object reference and make use of the dot operator to retrieve the HTTP method name in the JSP page. The JSP EL code for this purpose will look like ${pageContext.request.method}.

Exception and Thread Java Interview Questions

Q1. What is the difference between Error and Exception?

An error is an irrecoverable condition occurring at runtime. Such as OutOfMemory error. These JVM errors you cannot repair them at runtime. Though error can be caught in the catch block but the execution of application will come to a halt and is not recoverable.

While exceptions are conditions that occur because of bad input or human error etc. e.g. FileNotFoundException will be thrown if the specified file does not exist. Or a NullPointerException will take place if you try using a null reference. In most of the cases it is possible to recover from an exception (probably by giving the user feedback for entering proper values etc.

Q2. How can you handle Java exceptions?

There are five keywords used to handle exceptions in Java:

1. try

2. catch
3. finally
4. throw
5. throws

Q3. What are the differences between Checked Exception and Unchecked Exception?

Checked Exception

- The classes that extend Throwable class except RuntimeException and Error are known as checked exceptions.
- Checked exceptions are checked at compile-time.
- Example: IOException, SQLException etc.

Unchecked Exception

- The classes that extend RuntimeException are known as unchecked exceptions.
- Unchecked exceptions are not checked at compile-time.
- Example: ArithmeticException, NullPointerException etc.

Q4. What are the different ways of thread usage?

There are two ways to create a thread:

- Extending Thread class

This creates a thread by creating an instance of a new class that extends the Thread class. The extending class must override the run() function, which is the thread's entry point.

- Implementing Runnable interface

This is the easiest way to create a thread, by creating a class that implements the runnable interface. After implementing the runnable interface, the class must implement the public void run() method ()

The run() method creates a parallel thread in your programme. When run() returns, the thread will come to an end.

The run() method creates a parallel thread in your programme. When run() returns, the thread will come to an end.

Within the run() method, you must specify the thread's code.

Like any other method, the run() method can call other methods, use other classes, and define variables.

Java works as "pass by value" or "pass by reference" phenomenon?

Java is always pass-by-value. This means that it creates a copy of the contents of the parameter in memory. In Java, object variables always refer to the memory heap's real object.

Q5. Will the finally block get executed when the return statement is written at the end of try block and catch block as shown below?

The finally block always gets executed even hen the return statement is written at the end of the try block and the catch block. It always executes , whether there is an exception or not. There are only a few situations in which the finally block does not execute, such as VM crash, power failure, software crash, etc. If you don't want to execute the finally block, you need to call the System.exit() method explicitly in the finally block.

Q6. How does an exception propagate in the code?

If an exception is not caught, it is thrown from the top of the stack and falls down the call stack to the previous procedure. If the exception isn't caught there, it falls back to the previous function, and so on, until it's caught or the call stack reaches the bottom. The term for this is Exception propagation.

Q7. Can you explain the Java thread lifecycle?

The java thread lifecycle has the following states-

New-

When a thread is created, and before the program starts the thread, it is in the new state. It is also referred to as a born thread.

Runnable

When a thread is started, it is in the Runnable state. In this state, the thread is executing its task.

Waiting

Sometimes, a thread goes to the waiting state, where it remains idle because another thread is executing. When the other thread has finished, the waiting thread again comes into the running state.

Timed Waiting

In timed waiting, the thread goes to waiting state. But, it remains in waiting state for only a specified interval of time after which it starts executing.It remains waiting either till the time interval ends or till the other thread has finished.

Terminated

A thread is said to be in this state once it terminates. It may be because the thread has completed its task or due to any other reason.

Q8. What purpose do the keywords final, finally, and finalize fulfill?

Final:

Final is used to apply restrictions on class, method, and variable. A final class can’t be inherited, final method can’t be overridden and final variable value can’t be changed. Let’s take a look at the example below to understand it better.

```
class FinalVarExample {
public static void main( String args[])
{
final int a=10;   // Final variable
a=50;           //Error as value can't be
changed
}
```

Finally

Finally is used to place important code, it will be executed whether the exception is handled or not. Let's take a look at the example below to understand it better.

```
class FinallyExample {
public static void main(String args[]){
try {
int x=100;
}
catch(Exception e) {
System.out.println(e);
}
finally {
System.out.println("finally block is
executing");}
}}
}
```

Finalize

Finalize is used to perform clean up processing just before the object is garbage collected. Let's take a look at the example below to understand it better.

```
class FinalizeExample {
public void finalize() {
System.out.println("Finalize is called");
}
public static void main(String args[])
{
FinalizeExample f1=new FinalizeExample();
FinalizeExample f2=new FinalizeExample();
f1= NULL;
f2=NULL;
System.gc();
}
}
```

Q9. What are the differences between throw and throws?

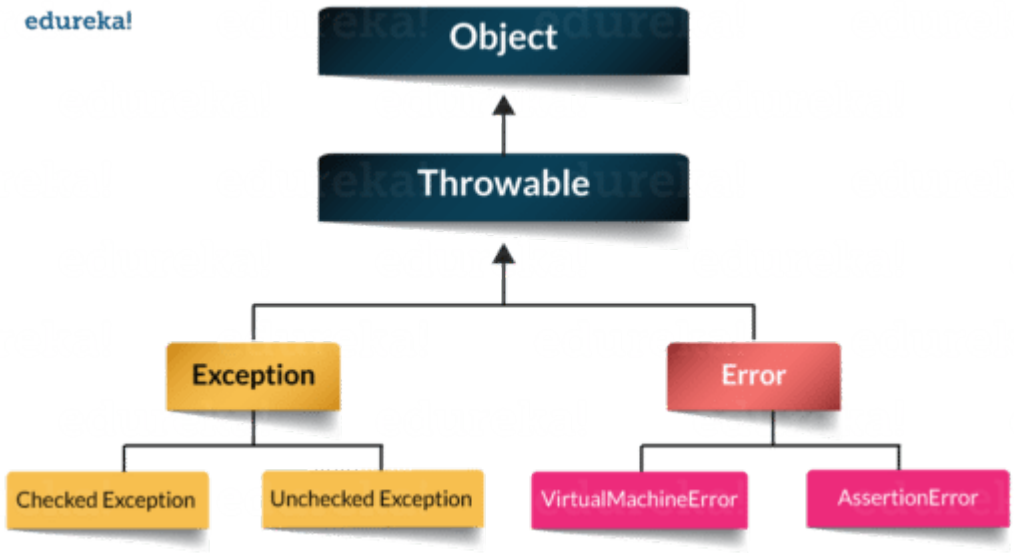
throw keyword	throws keyword
Throw is used to explicitly throw an exception.	Throws is used to declare an exception.
Checked exceptions can not be propagated with throw only.	Checked exception can be propagated with throws.
Throw is followed by an instance.	Throws is followed by class.
Throw is used within the method.	Throws is used with the method signature.
You cannot throw multiple exception	You can declare multiple exception e.g. public void

method()throws IOException,SQLException.

Q10. What is exception hierarchy in java?

The hierarchy is as follows:

Throwable is a parent class of all Exception classes. There are two types of Exceptions: Checked exceptions and UncheckedExceptions or RunTimeExceptions. Both type of exceptions extends Exception class whereas errors are further classified into Virtual Machine error and Assertion error.



Q11. How to create a custom Exception?

To create you own exception extend the Exception class or any of its subclasses.

- class New1Exception extends Exception { } // this will create Checked Exception
- class NewException extends IOException { } // this will create Checked exception
- class NewException extends NullPonterExcpetion { } // this will create UnChecked exception

Q12. What are the important methods of Java Exception Class?

Exception and all of it's subclasses doesn't provide any specific methods and all of the methods are defined in the base class Throwable.

1. **String getMessage()** – This method returns the message String of Throwable and the message can be provided while creating the exception through it's constructor.
2. **String getLocalizedMessage()** – This method is provided so that subclasses can override it to provide locale specific message to the calling program. Throwable class implementation of this method simply use getMessage() method to return the exception message.
3. **Synchronized Throwable getCause()** – This method returns the cause of the exception or null id the cause is unknown.
4. **String toString()** – This method returns the information about Throwable in String format, the returned String contains the name of Throwable class and localized message.
5. **void printStackTrace()** – This method prints the stack trace information to the standard error stream, this method is overloaded and we can pass PrintStream or PrintWriter as an argument to write the stack trace information to the file or stream.

Q13. What are the differences between processes and threads?

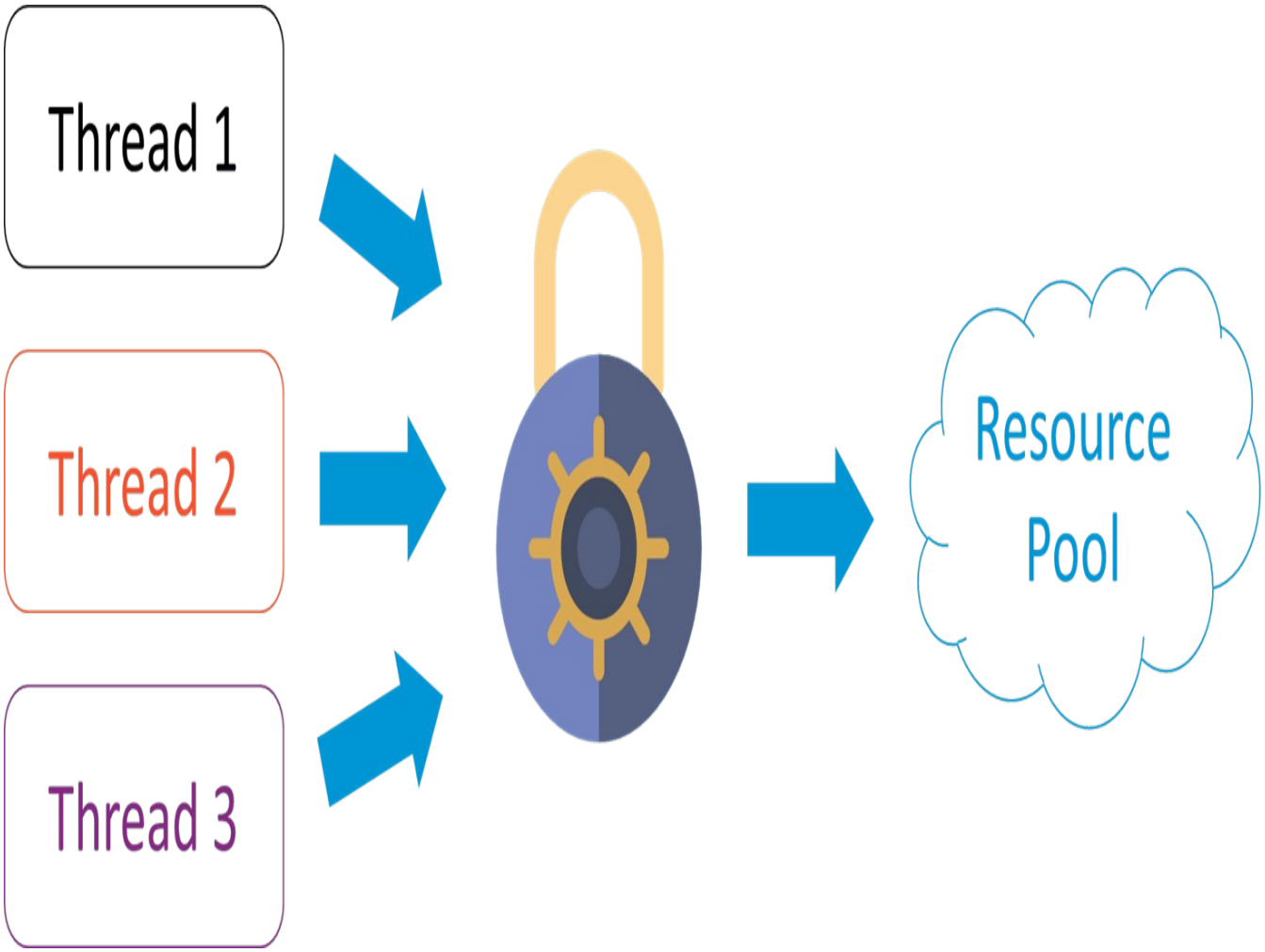
	Process	Thread
Definition	An executing instance of a program is called a process.	A thread is a subset of the process.
Communication	Processes must use inter-process communication to communicate with sibling processes.	Threads can directly communicate with other threads of its process.
Control	Processes can only exercise control over child processes.	Threads can exercise considerable control over threads of the same process.
Changes	Any change in the parent process does not affect child processes.	Any change in the main thread may affect the behavior of the other threads of the process.
Memory	Run in separate memory spaces.	Run in shared memory spaces.
Controlled by	Process is controlled by the operating system.	Threads are controlled by programmer in a program.
Dependence	Processes are independent.	Threads are dependent.

Q14. What is a finally block? Is there a case when finally will not execute?

Finally block is a block which always executes a set of statements. It is always associated with a try block regardless of any exception that occurs or not. Yes, finally will not be executed if the program exits either by calling System.exit() or by causing a fatal error that causes the process to abort.

Q15. What is synchronization?

Synchronization refers to multi-threading. A synchronized block of code can be executed by only one thread at a time. As Java supports execution of multiple threads, two or more threads may access the same fields or objects. Synchronization is a process which keeps all concurrent threads in execution to be in sync. Synchronization avoids memory consistency errors caused due to inconsistent view of shared memory. When a method is declared as synchronized the thread holds the monitor for that method's object. If another thread is executing the synchronized method the thread is blocked until that thread releases the monitor.



Q16. Can we write multiple catch blocks under single try block?

Yes we can have multiple catch blocks under single try block but the approach should be from specific to general. Let's understand this with a programmatic example.

```
public class Example {
public static void main(String args[]) {
try {
int a[]= new int[10];
a[10]= 10/0;
}
catch(ArithmeticException e)
{
System.out.println("Arithmetic exception in first catch block");
}
catch(ArrayIndexOutOfBoundsException e)
{
System.out.println("Array index out of bounds in second catch
block");
}
catch(Exception e)
{
System.out.println("Any exception in third catch block");
}
}
```

Q17. What are the important methods of Java Exception Class?

Methods are defined in the base class Throwable. Some of the important methods of Java exception class are stated below.

1. **String getMessage()** – This method returns the message String about the exception. The message can be provided through its constructor.
2. **public StackTraceElement[] getStackTrace() –** This method returns an array containing each element on the stack trace. The element at index 0 represents the top of the call stack whereas the last element in the array represents the method at the bottom of the call stack.
3. **Synchronized Throwable getCause()** – This method returns the cause of the exception or null id as represented by a Throwable object.
4. **String toString()** – This method returns the information in String format. The returned String contains the name of Throwable class and localized message.
5. **void printStackTrace()** – This method prints the stack trace information to the standard error stream.

Q18. What is OutOfMemoryError in Java?

OutOfMemoryError is the subclass of java.lang.Error which generally occurs when our JVM runs out of memory.

Q19. What is a Thread?

A thread is the smallest piece of programmed instructions which can be executed independently by a scheduler. In Java, all the programs will have at least one thread which is known as the main thread. This main thread is created by the JVM when the program starts its execution. The main thread is used to invoke the main() of the program.

Q20. What are the two ways to create a thread?

In Java, threads can be created in the following two ways:-

- By implementing the Runnable interface.
- By extending the Thread

Q21. What are the different types of garbage collectors in Java?

Garbage collection in Java a program which helps in implicit memory management. Since in Java, using the new keyword you can create objects dynamically, which once created will consume some memory. Once the job is done and there are no more references left to the object, Java using garbage collection destroys the object and relieves the memory occupied by it. Java provides four types of garbage collectors:

- Serial Garbage Collector
- Parallel Garbage Collector
- CMS Garbage Collector
- G1 Garbage Collector