Java interview preparation:

**Q-01 : Differences between JDK, JRE and JVM**

### **JDK**

JDK is an acronym for Java Development Kit. The Java Development Kit (JDK) is a software development environment which is used to develop Java applications and [applets](https://www.javatpoint.com/java-applet). It physically exists. It contains JRE + development tools.

The JDK contains a private Java Virtual Machine (JVM) and a few other resources such as an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc), etc. to complete the development of a Java Application.

### **JRE**

JRE is an acronym for Java Runtime Environment. It is also written as Java RTE. The Java Runtime Environment is a set of software tools which are used for developing Java applications. It is used to provide the runtime environment. It is the implementation of JVM. It physically exists. It contains a set of libraries + other files that JVM uses at runtime.

The implementation of JVM is also actively released by other companies besides Sun Micro Systems.

### **JVM**

JVM (Java Virtual Machine) is an abstract machine. It is called a virtual machine because it doesn't physically exist. It is a specification that provides a runtime environment in which Java byte code can be executed. It can also run those programs which are written in other languages and compiled to Java bytecode.

JVMs are available for many hardware and software platforms. JVM, JRE, and JDK are platform dependent because the configuration of each [OS](https://www.javatpoint.com/os-tutorial) is different from each other. However, Java is platform independent. There are three notions of the JVM: specification, implementation, and instance.

The JVM performs the following main tasks:

* Loads code
* Verifies code
* Executes code
* Provides runtime environment

Learn more : <https://www.javatpoint.com/jvm-java-virtual-machine>

### **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.

### **2. What is Java?**

Java is a programming language and computing platform first released by Sun Microsystems in 1995. There are lots of applications and websites that will not work unless you have Java installed, and more are created every day. Java is fast, secure, and reliable. From laptops to datacenters, game consoles to scientific supercomputers, cell phones to the Internet, Java is everywhere!

Java program which prints Hello World!



|  |  |
| --- | --- |
| 1  2  3  4  5 | public class MyClass {    public static void main(String args[]) {      System.out.println("Hello World!");    }  } |

### **3. Mention some features of Java?**

Some of the features which play an important role in the popularity of java are as follows:

* Simple: Java is easy to learn. Even though Java is based on C++ , it was developed by eliminating poor programming practices of C++.
* Object-Oriented: Java is an object-oriented programming language. Everything in Java is an Object.

[Learn more here.](https://www.softwaretestingmaterial.com/oops-concept-in-java/)

* **Portable:** Java run time environment uses a bytecode verification process to make sure that code loaded over the network doesn’t violate Java security constraints.
* **Platform independent:** Java is platform-independent. Java is a write once, run anywhere language. Without any modifications, we can use a program on different platforms.
* **Secured:** Java is well known for its security. It delivers virus-free systems.
* **High Performance:** Java enables high performance with the use of JIT (Just-In-Time) compilers
* **Multithreaded:** Java Multithreaded features allows us to write programs that can perform many tasks simultaneously. The multithreading concept of Java shares a common memory area. It doesn’t occupy memory for each thread.

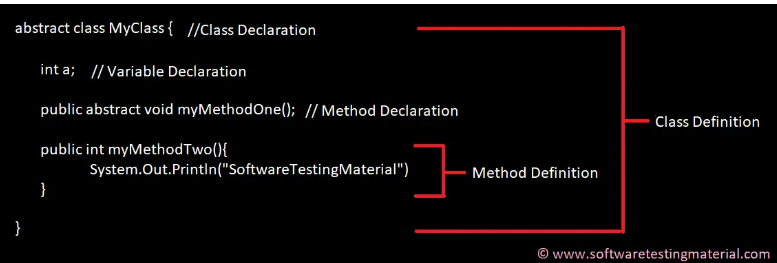
### **6. What is the difference between Declaration and Definition in Java?**

**Declaration:** If you just declare a class or method/function or variable without mentioning anything about what that class or method/function or variable looks like is called a declaration in Java.

**Definition:** If you define how a class or method/function or variable is implemented then it is called definition in Java.

When we create an interface or abstract class, we simply declare a method/function but not define it.

For a clear understanding, check the below image



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

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

Both equals() method and the == operator are used to compare two objects in Java. == is an operator and equals() is method. But == operator compares reference or memory location of objects in a heap, whether they point to the same location or not.  
Whenever we create an object using the operator new, it will create a new memory location for that object. So we use the == operator to check memory location or address of two objects are the same or not.

In general, both equals() and “==” operators in[Java](https://www.geeksforgeeks.org/java-tutorial/) are used to compare objects to check equality, but here are some of the differences between the two:

1. The main difference between the [.equals() method](https://www.geeksforgeeks.org/method-class-equals-method-in-java/) and ==[operator](https://www.geeksforgeeks.org/operators-in-java/) is that one is a method, and the other is the operator.
2. 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.

### **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. |

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

The Non-Serialized attribute can be used to prevent member variables from being serialized.  
You should also make an object that potentially contains security-sensitive data non-serializable if possible. Apply the Non-Serialized attribute to certain fields that store sensitive data if the object must be serialized. If you don’t exclude these fields from serialization, the data they store will be visible to any programs 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-

### Programming & Frameworks Training

|  |  |
| --- | --- |
| 1  2  3  4 | 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.

**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.

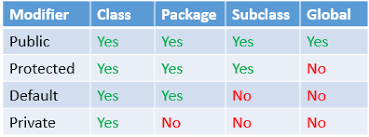
### **Q24. What is JIT compiler in Java?**

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



### **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.

|  |  |
| --- | --- |
| 1  2  3 | 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**

|  |  |
| --- | --- |
| 1  2  3  4 | 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.

|  |  |
| --- | --- |
| 1  2  3  4 | 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 |

In case you are facing any challenges with these Core Java interview questions, please comment on your problems in the section below.

What is the difference between a function and method?

**A function is a set of instructions or procedures to perform a specific task, and a method is a set of instructions that are associated with an object**.

### **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:***

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | for (int i = 0; i < 5; i++)  {  if (i == 3)  {  break;  }  System.out.println(i);  } |

***Example continue:***

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | 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:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | 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.  }  } |

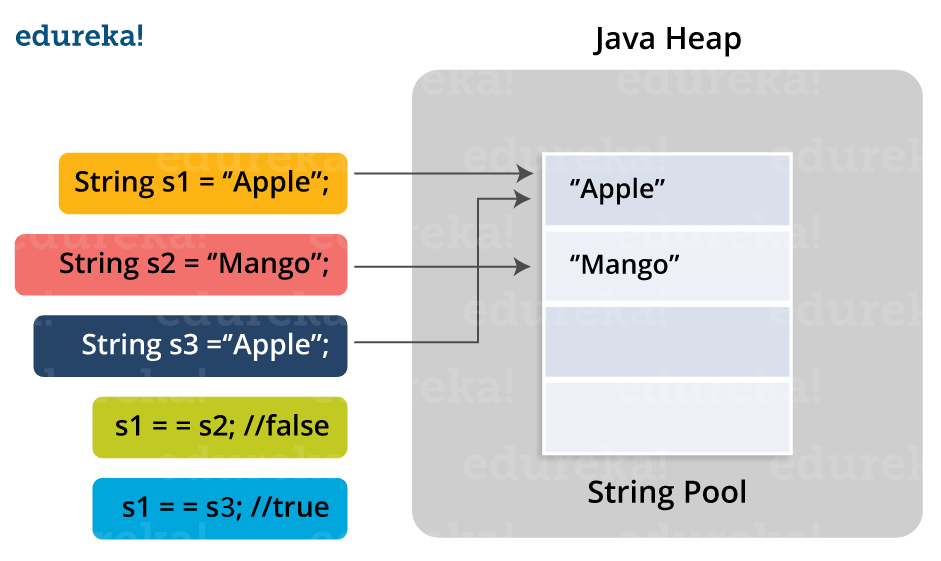
### **Q35. What is the difference between this() and super() in Java?**

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 |

### **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 |

If you think this article on Java Interview Questions is helpful, you can check out Edureka’s [Ja](https://www.edureka.co/java-j2ee-training-course-chennai)

### **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 |

If you think this article on Java Interview Questions is helpful, you can check out Edureka’s [Java Training in Chennai](https://www.edureka.co/java-j2ee-training-course-chennai) as well.

### **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.



Want to upskill yourself to get ahead in Career? Check out this video

Core Java Interview Quesions And Answers for Freshers and Experienced

### **13. What is Local Variable and Instance Variable?**

**Local Variable:**

A local variable is a variable that we declare inside a Method. A method will often store its temporary state in local variables.

It can be accessible only inside a block, function, or constructor.



|  |  |
| --- | --- |
| 1  2  3  4  5 | public void website() {  String websiteName;  double websiteLoadTime;  int webisteAge;  } |

String websiteName, double websiteLoadTime, int websiteAge are Local variables in above example.

**Instance Variable (Non-static):**

An instance variable is a variable that is declared inside a Class but outside a Method. We don’t declare this variable as Static because these variables are non-static variables.

It can be accessible by all the methods in the class.



|  |  |
| --- | --- |
| 1  2  3  4  5 | class website() {  public String websiteName;  public double websiteLoadTime;  public int webisteAge;  } |

websiteName, websiteLoadTime, websiteAge are Instance variables in above example.

### **26. Why are strings immutable in Java?**

In Java, String is immutable to make sure that the string value doesn’t change. String literals are usually shared between multiple clients. If the value of the string changes (from “STM” to “stm”), it will affect all reference variables and cause severe discrepancies.  
Hence, strings are immutable in Java. Making string immutable enhances security, caching, synchronization, and performance of the application.

### **42. How to read a file line by line in Java?**

We can read a file line by line in Java in two ways.

1. BufferedReader Class  
2. Scanner Class

**Using BufferedReader Class:**

BufferedReader Class belongs to java.io package and it provides readLine() method to read a file line by line in Java.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | package softwareTestingMaterial;    import java.io.BufferedReader;  import java.io.FileReader;  import java.io.IOException;    public class ReadLineByProgram {    public static void main(String[] args) {  BufferedReader reader;  try {  reader = new BufferedReader(new FileReader(  "/Users/Rajkumar/Downloads/STM.txt"));  String line = reader.readLine();  while (line != null) {  System.out.println(line);  // read next line  line = reader.readLine();  }  reader.close();  } catch (IOException e) {  e.printStackTrace();  }  }  } |

**Using Scanner Class:**

Java Scanner class provides the nextLine() method to facilitates line by line of file’s content.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | package softwareTestingMaterial;    import java.io.File;  import java.io.FileNotFoundException;  import java.util.Scanner;    public class ReadLineByProgram {    public static void main(String[] args) {  try {  Scanner scanner = new Scanner(new File("/Users/Rajkumar/Downloads/STM.txt"));  while (scanner.hasNextLine()) {  System.out.println(scanner.nextLine());  }  scanner.close();  } catch (FileNotFoundException e) {  e.printStackTrace();  }  }  } |

### **46. Difference between static binding and dynamic binding?**

1. Static binding is also known as early binding whereas dynamic binding is also known as late binding.  
2. Determining the type of an object at compile time is Static binding whereas determining the type of an object at run time is dynamic binding  
3. Java uses static binding for overloaded methods and dynamic binding for overridden methods.

To know more about this you have to go through [Method Overloading](https://www.softwaretestingmaterial.com/method-overloading-in-java/) and [Method Overriding](https://www.softwaretestingmaterial.com/method-overriding-in-java/).

# Autoboxing and Unboxing in Java

n Java, primitive data types are treated differently so do there comes the introduction of [wrapper classes](https://www.geeksforgeeks.org/wrapper-classes-java/) where two components play a role namely Autoboxing and Unboxing. [Autoboxing](https://www.geeksforgeeks.org/autoboxing-unboxing-java/) refers to the conversion of a primitive value into an object of the corresponding wrapper class is called autoboxing. For example, converting int to Integer class. The Java compiler applies autoboxing when a primitive value is:

* Passed as a parameter to a method that **expects an object** of the corresponding wrapper class.
* Assigned to a variable of the corresponding **wrapper class**.

**Unboxing**on the other hand refers to converting an object of a wrapper type to its corresponding primitive value. For example conversion of Integer to int. The Java compiler applies to unbox when an object of a wrapper class is:

* Passed as a parameter to a method that **expects a value** of the corresponding primitive type.
* Assigned to a variable of the corresponding **primitive type**.

<https://www.geeksforgeeks.org/autoboxing-unboxing-java/>

# Java Reflection API

**Java Reflection** is a *process of examining or modifying the run time behavior of a class at run time*.

There are 3 ways to get the instance of Class class. They are as follows:

* forName() method of Class class
* getClass() method of Object class
* the .class syntax

https://www.geeksforgeeks.org/reflection-in-java/

# Java Lambda Expressions

It provides a clear and concise way to represent one method interface using an expression.

The Lambda expression is used to provide the implementation of an interface which has functional interface

## **Functional Interface**

Lambda expression provides implementation of functional interface. An interface which has only one abstract method is called functional interface. Java provides an anotation @FunctionalInterface, which is used to declare an interface as functional interface.

## **Why use Lambda Expression**

1. To provide the implementation of Functional interface.

### **49. What are the differences between throw and throws in Java?**

**throw keyword**

* The throw keyword is used to explicitly throw an exception in the program inside a function or inside a block of code.
* The checked exceptions cannot be propagated with throw only.
* The throw keyword is followed by an instance.
* The throw keyword is used within the method.
* You cannot throw multiple exceptions.

**throws keyword**

* The throws keyword is used in the method signature to declare an exception which might get thrown by the function while executing the code.
* The checked exception can be propagated with throws
* The throws keyword is followed by class.
* The throws keyword is used with the method signature.
* You can declare multiple exceptions, e.g., public void method()throws IOException, SQLException.

Error Handing in Java

[Error](https://www.geeksforgeeks.org/types-of-errors-in-java-with-examples/) refers to an illegal operation performed by the user

It is of three types:

* Compile-time
* Run-time
* Logical

Whereas [exceptions in java](https://www.geeksforgeeks.org/exceptions-in-java/) refer to an unwanted or unexpected event, which occurs during the execution of a program

* [Checked exceptions](https://www.geeksforgeeks.org/checked-vs-unchecked-exceptions-in-java/)
* [Unchecked exceptions](https://www.geeksforgeeks.org/checked-vs-unchecked-exceptions-in-java/)

### **Differences between Checked Exception and Unchecked Exception:**

* **Checked Exception**

1. Checked exceptions occur at compile time.
2. Here, the JVM needs the exception to catch and handle.

* File Not Found Exception
* No Such Field Exception
* Interrupted Exception
* No Such Method Exception
* Class Not Found Exception
* **Un-Checked Exception**

1. Un-Checked exceptions occur at run time.
2. Here, the JVM does not require the exception to catch and handle

* No Such Element Exception
* Undeclared Throwable Exception
* Empty Stack Exception
* Arithmetic Exception
* Null Pointer Exception
* Array Index Out of Bounds Exception
* Security Exception

# Java Reflection API

**Java Reflection** is a *process of examining or modifying the run time behavior of a class at run time*.

There are 3 ways to get the instance of Class class. They are as follows:

* forName() method of Class class
* getClass() method of Object class
* the .class syntax

# Java Lambda Expressions

It provides a clear and concise way to represent one method interface using an expression.

The Lambda expression is used to provide the implementation of an interface which has functional interface

## **Functional Interface**

Lambda expression provides implementation of functional interface. An interface which has only one abstract method is called functional interface. Java provides an anotation @FunctionalInterface, which is used to declare an interface as functional interface.

## **Why use Lambda Expression**

1. To provide the implementation of Functional interface.

AOP(Aspect oriented programming)

# AOP(Aspect Oriented Programming)

## **AOP**

AOP **(Aspect-Oriented Programming)** is a programming pattern that increases modularity by allowing the separation of the **cross-cutting concern**. These cross-cutting concerns are different from the main business logic. We can add additional behavior to existing code without modification of the code itself.

Spring's AOP framework helps us to implement these cross-cutting concerns.

Using AOP, we define common functionality in one place. We are free to define how and where this functionality is applied without modifying the class to which we are applying the new feature. The cross-cutting concern can now be modularized into special classes, called **aspect**.

There are **two** benefits of aspects:

* First, the logic for each concern is now in one place instead of scattered all over the codebase.
* Second, the business modules only contain code for their primary concern. The secondary concern has been moved to the **aspect**.

The aspects have the responsibility that is to be implemented, called **advice**. We can implement an aspect's functionality into a program at one or more join points.

## **Benefits of AOP**

* It is implemented in pure Java.
* There is no requirement for a special compilation process.
* It supports only method execution Join points.
* Only run time weaving is available.
* Two types of AOP proxy is available: **JDK dynamic proxy** and **CGLIB proxy.**

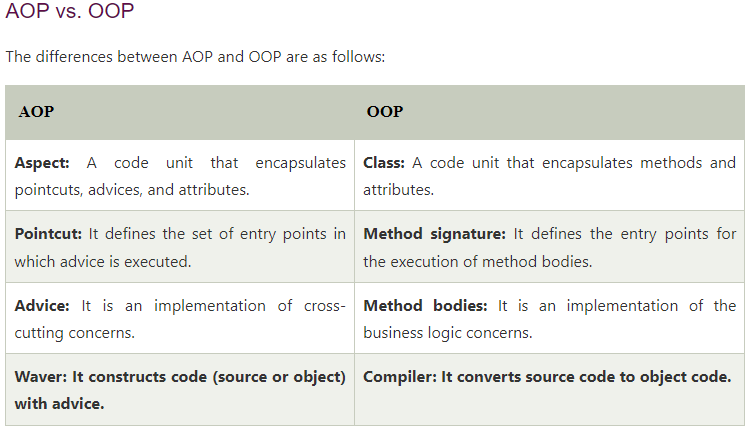
## **Cross-cutting concern**

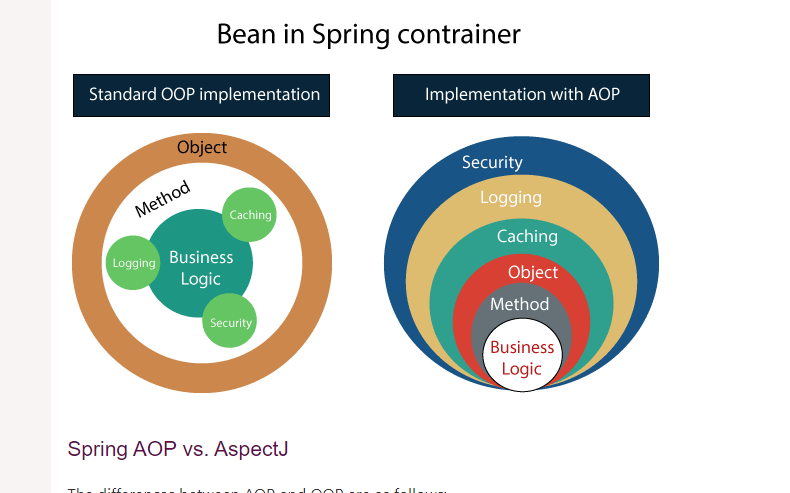
The cross-cutting concern is a concern that we want to implement in multiple places in an application. It affects the entire application. Ex **Logging, Security, validation, caching,** etc

## **AOP Terminology**

* **Aspect:** An aspect is a module that encapsulates **advice** and **pointcuts** and provides **cross-cutting** An application can have any number of aspects. We can implement an aspect using regular class annotated with **@Aspect** annotation.
* **Pointcut:** A pointcut is an expression that selects one or more join points where advice is executed. We can define pointcuts using **expressions** or **patterns**. It uses different kinds of expressions that matched with the join points. In Spring Framework, **AspectJ** pointcut expression language is used.
* **Join point:** A join point is a point in the application where we apply an **AOP aspect**. Or it is a specific execution instance of an advice. In AOP, join point can be a **method execution, exception handling, changing object variable value**, etc.
* **Advice:** The advice is an action that we take either **before** or **after** the method execution. The action is a piece of code that invokes during the program execution. There are **five** types of advices in the Spring AOP framework: **before, after, after-returning, after-throwing,**and **around advice.**Advices are taken for a particular **join point.**We will discuss these advices further in this section.
* **Target object:** An object on which advices are applied, is called the **target object**. Target objects are always a **proxied** It means a subclass is created at run time in which the target method is overridden, and advices are included based on their configuration.
* **Weaving:** It is a process of **linking aspects** with other application types. We can perform weaving at **run time, load time,** and **compile time**.

**Proxy:** It is an object that is created after applying advice to a target object is called **proxy**. The Spring AOP implements the **JDK dynamic proxy** to create the proxy classes with target classes and advice invocations. These are called AOP proxy classes





## **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.

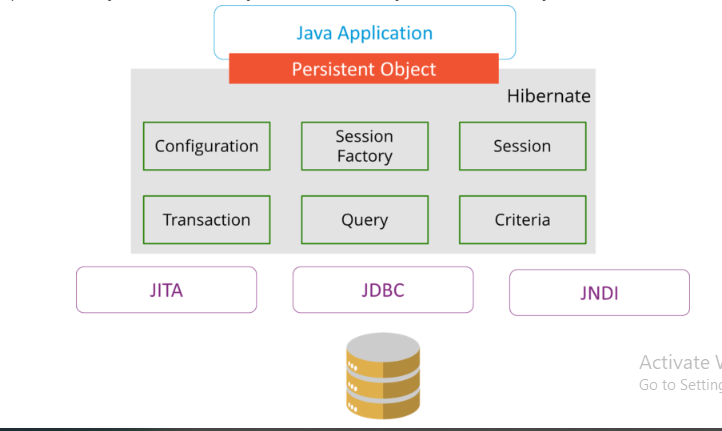
### **2. What are the important benefits of using Hibernate Framework?**

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.

In case you are facing any challenges with these Java interview questions, please comment on your problems in the section below. Apart from this Java Interview Questions Blog, if you want to get trained from professionals on this technology, you can opt for structured training from edureka!

### 1) What is hibernate?

Hibernate is an open-source and lightweight ORM tool that is used to store, manipulate, and retrieve data from the database.

[more details...](https://www.javatpoint.com/hibernate-tutorial)

### 2) What is ORM?

ORM is an acronym for Object/Relational mapping. It is a programming strategy to map object with the data stored in the database. It simplifies data creation, data manipulation, and data access.

### 3) Explain hibernate architecture?

Hibernate architecture comprises of many interfaces such as Configuration, SessionFactory, Session, Transaction, etc.

[more details...](https://www.javatpoint.com/hibernate-architecture) 

### 4) What are the core interfaces of Hibernate?

The core interfaces of Hibernate framework are:

* Configuration
* SessionFactory
* Session
* Query
* Criteria
* Transaction

### 5) Mention some of the advantages of using ORM over JDBC.

ORM has the following advantages over JDBC:

* Application development is fast.
* Management of transaction.
* Generates key automatically.
* Details of SQL queries are hidden.

### 6) Define criteria in terms of Hibernate.

The objects of criteria are used for the creation and execution of the object-oriented criteria queries.

### 7) List some of the databases supported by Hibernate.

Some of the databases supported by Hibernate are:

* DB2
* MySQL
* Oracle
* Sybase SQL Server
* Informix Dynamic Server
* HSQL
* PostgreSQL
* FrontBase

### 8) List the key components of Hibernate.

Key components of Hibernate are:

* Configuration
* Session
* SessionFactory
* Criteria
* Query
* Transaction

### 9) Mention two components of Hibernate configuration object.

Database Connection

Class Mapping Setup

### 10) How is SQL query created in Hibernate?

The SQL query is created with the help of the following syntax:

Session.createSQLQuery

### 11) What does HQL stand for?

Hibernate Query Language

### 12) How is HQL query created?

The HQL query is created with the help of the following syntax:

Session.createQuery

### 13) How can we add criteria to a SQL query?

A criterion is added to a SQL query by using the Session.createCriteria.

### 14) Define persistent classes.

Classes whose objects are stored in a database table are called as persistent classes.

### 15) What is SessionFactory?

SessionFactory provides the instance of Session. It is a factory of Session. It holds the data of second level cache that is not enabled by default.

[more details...](https://www.javatpoint.com/hibernate-architecture)

### 16) Is SessionFactory a thread-safe object?

Yes, SessionFactory is a thread-safe object, many threads cannot access it simultaneously.

### 17) What is Session?

It maintains a connection between the hibernate application and database.

It provides methods to store, update, delete or fetch data from the database such as persist(), update(), delete(), load(), get() etc.

It is a factory of Query, Criteria and Transaction i.e. it provides factory methods to return these instances.

[more details...](https://www.javatpoint.com/hibernate-architecture)

### 18) Is Session a thread-safe object?

No, Session is not a thread-safe object, many threads can access it simultaneously. In other words, you can share it between threads.

### 19) What is the difference between session.save() and session.persist() method?

|  |  |  |
| --- | --- | --- |
| **No.** | **save()** | **persist()** |
| 1) | returns the identifier (Serializable) of the instance. | Return nothing because its return type is void. |
| 2) | Syn: public Serializable save(Object o) | Syn: public void persist(Object o) |

### 20) What is the difference between get and load method?

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

|  |  |  |
| --- | --- | --- |
| **No.** | **get()** | **load()** |
| 1) | Returns **null** if an 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 the real object, not the proxy. | It returns **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 instance exists. |

### 21) What is the difference between update and merge method?

The differences between update() and merge() methods are given below.

|  |  |  |
| --- | --- | --- |
| **No.** | **The update() method** | **merge() method** |
| 1) | Update means to edit something. | Merge means to combine something. |
| 2) | update() should be used if the session doesn't contain an already persistent state with the same id. It means an update should be used inside the session only. After closing the session, it will throw the error. | merge() should be used if you don't know the state of the session, means you want to make the modification at any time. |

Let's try to understand the difference by the example given below:

1. ...
2. SessionFactory factory = cfg.buildSessionFactory();
3. Session session1 = factory.openSession();
5. Employee e1 = (Employee) session1.get(Employee.class, Integer.valueOf(101));//passing id of employee
6. session1.close();
8. e1.setSalary(70000);
10. Session session2 = factory.openSession();
11. Employee e2 = (Employee) session1.get(Employee.class, Integer.valueOf(101));//passing same id
13. Transaction tx=session2.beginTransaction();
14. session2.merge(e1);
16. tx.commit();
17. session2.close();

After closing session1, e1 is in detached state. It will not be in the session1 cache. So if you call update() method, it will throw an error.

Then, we opened another session and loaded the same Employee instance. If we call merge in session2, changes of e1 will be merged in e2.

### 22) What are the states of the object in hibernate?

There are 3 states of the object (instance) in hibernate.

1. **Transient**: The object is in a transient state if it is just created but has no primary key (identifier) and not associated with a session.
2. **Persistent**: The object is in a persistent state if a session is open, and you just saved the instance in the database or retrieved the instance from the database.
3. **Detached**: The object is in a detached state if a session is closed. After detached state, the object comes to persistent state if you call lock() or update() method.

### 23) What are the inheritance mapping strategies?

There are 3 ways of inheritance mapping in hibernate.

1. Table per hierarchy
2. Table per concrete class
3. Table per subclass

[more details...](https://www.javatpoint.com/hibernate-inheritance-mapping-tutorial)

### 24) How to make an immutable class in hibernate?

If you mark a class as mutable="false", the class will be treated as an immutable class. By default, it is mutable="true".

### 25) What is automatic dirty checking in hibernate?

The automatic dirty checking feature of Hibernate, calls update statement automatically on the objects that are modified in a transaction.

Let's understand it by the example given below:

1. ...
2. SessionFactory factory = cfg.buildSessionFactory();
3. Session session1 = factory.openSession();
4. Transaction tx=session2.beginTransaction();
6. Employee e1 = (Employee) session1.get(Employee.class, Integer.valueOf(101));
8. e1.setSalary(70000);
10. tx.commit();
11. session1.close();

Here, after getting employee instance e1 and we are changing the state of e1.

After changing the state, we are committing the transaction. In such a case, the state will be updated automatically. This is known as dirty checking in hibernate.

### 26) How many types of association mapping are possible in hibernate?

There can be 4 types of association mapping in hibernate.

1. One to One
2. One to Many
3. Many to One
4. Many to Many

### 27) Is it possible to perform collection mapping with One-to-One and Many-to-One?

No, collection mapping can only be performed with One-to-Many and Many-to-Many.

### 28) What is lazy loading in hibernate?

Lazy loading in hibernate improves the performance. It loads the child objects on demand.

Since Hibernate 3, lazy loading is enabled by default, and you don't need to do lazy="true". It means not to load the child objects when the parent is loaded.

### 29) What is HQL (Hibernate Query Language)?

Hibernate Query Language is known as an object-oriented query language. It is like a structured query language (SQL).

The main advantage of HQL over SQL is:

1. You don't need to learn SQL
2. Database independent
3. Simple to write a query

### 30) What is the difference between first level cache and second level cache?

|  |  |  |
| --- | --- | --- |
| **No.** | **First Level Cache** | **Second Level Cache** |
| 1) | First Level Cache is **associated with Session**. | Second Level Cache is associated with **SessionFactory**. |
| 2) | It is **enabled** by default. | It is **not enabled** by default. |

## **Hibernate Interview Questions for beginners**

### **Q1. What is Hibernate?**

[*Hibernate*](https://www.edureka.co/blog/what-is-hibernate-in-java/) is one of the most popular [*Java frameworks*](https://www.edureka.co/blog/java-frameworks/) that simplify the development of Java application to interact with the database. It is an Object-relational mapping (ORM) tool. Hibernate also provides a reference implementation of Java API.

It is referred as a framework which comes with an abstraction layer and also handles the implementations internally. The implementations include tasks like writing a query for [*CRUD*](https://www.edureka.co/blog/node-js-mysql-tutorial/) operations or establishing a connection with the databases, etc.

Hibernate develops persistence logic, which stores and processes the data for longer use. It is a lightweight tool and most importantly **open-sourced**which gives it an edge over other frameworks.

### **Q2. What are the major advantages of Hibernate Framework?**

* It is open-sourced and lightweight.
* Performance of Hibernate is very fast.
* Helps in generating database independant queries.
* Provides facilities to automatically create a table.
* It provides query statistics and database status.

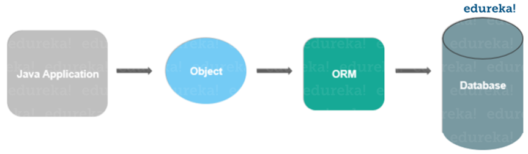
### **Q3. What are the advantages of using Hibernate over JDBC?**

Major advantages of using Hibernate over JDBC are:

1. Hibernate eliminates a lot of boiler-plate code that comes with [*JDBC API*](https://www.edureka.co/blog/connect-mysql-database-in-java/), the code looks cleaner and readable.
2. This Java framework supports [*inheritance*](https://www.edureka.co/blog/inheritance-in-java/), associations, and collections. These features are actually not present in JDBC.
3. HQL (Hibernate Query Language) is more object-oriented and close to Java. But for JDBC, you need to write native SQL queries.
4. Hibernate implicitly provides transaction management whereas, in JDBC API, you need to write code for transaction management using commit and rollback.
5. JDBC throws SQLException that is a checked exception, so you have to write a lot of try-catch block code. Hibernate wraps JDBC exceptions and throw JDBCException or HibernateException which are the unchecked exceptions, so you don’t have to write code to handle it has built-in transaction management which helps in removing the usage of try-catch blocks.

### **Q4. What is an ORM tool?**

It is basically a technique that maps the object that is stored in the database. An ORM tool helps in simplifying data creation, manipulation, and access. It internally uses the Java API to interact with the databases.



### **Q5. Why use Hibernate Framework?**

Hibernate overcomes the shortcomings of other technologies like [*JDBC*](https://www.edureka.co/blog/connect-mysql-database-in-java).

* It overcomes the database dependency faced in the JDBC.
* Changing of the databases cost a lot working on JDBC, hibernate overcomes this problem with flying colors.
* Code portability is not an option while working on JDBC. This is easily handled by Hibernate.
* Hibernate strengthens the object level relationship.
* It overcomes the [*exception-handling*](https://www.edureka.co/blog/java-exception-handling) part which is mandatory while working on JDBC.
* It reduces the length of code with increased readability by overcoming the boilerplate problem.

### **Q6. What are the different functionalities supported by Hibernate?**

* Hibernate is an ORM tool.
* Hibernate uses Hibernate Query Language(HQL) which makes it database-independent.
* It supports auto DDL operations.
* This Java framework also has an Auto Primary Key Generation support.
* Supports cache memory.
* Exception handling is not mandatory in the case of Hibernate.

### **Q7. What are the technologies that are supported by Hibernate?**

Hibernate supports a variety of technologies, like:

* XDoclet Spring
* [*Maven*](https://www.edureka.co/blog/create-selenium-maven-project/)
* Eclipse Plug-ins
* J2EE

### **Q8. What is HQL?**

HQL is the acronym of Hibernate Query Language. It is an Object-Oriented Query Language and is independent of the database.

### **Q9. How to achieve mapping in Hibernate?**

Association mappings are one of the key features of Hibernate. It supports the same associations as the relational database model. They are:

* One-to-One associations
* Many-to-One associations
* Many-to-Many associations

You can map each of them as a uni- or bidirectional association.

### **Q10. Name some of the important interfaces of Hibernate framework?**

Hibernate interfaces are:

* **SessionFactory** (org.hibernate.SessionFactory)
* **Session** (org.hibernate.Session)
* **Transaction** (org.hibernate.Transaction)

### **Q11. What is One-to-One association in Hibernate?**

In this type of mapping,  you only need to model the system for the entity for which you want to navigate the relationship in your query or domain model. You need an entity attribute that represents the association, so annotate it with an @OneToOne annotation.

### **Q12. What is One-to-Many association in Hibernate?**

In this type of association, one object can be associated with multiple/different objects. Talking about the mapping, the One-to-Many mapping is implemented using a [*Set Java*](https://www.edureka.co/blog/sets-in-java/) collection that does not have any redundant element. This One-to-Many element of the set indicates the relation of one object to multiple objects.

### **Q13. What is Many-to-Many association in Hibernate?**

Many-to-Many mapping requires an entity attribute and a @ManyToMany annotation. It can either be unidirectional and bidirectional. In **Unidirectional**, the attributes model the association and you can use it to navigate it in your domain model or JPQL queries. The annotation tells Hibernate to map a Many-to-Many association. The **bidirectional** relationship, mapping allows you to navigate the association in both directions.

### **Q14. How to integrate Hibernate and Spring?**

[*Spring*](https://www.edureka.co/blog/spring-tutorial/) is also one of the most commonly used Java frameworks in the market today. Spring is a JavaEE Framework and Hibernate is the most popular ORM framework. This is why Spring Hibernate combination is used in a lot of enterprise applications.

Following are the steps you should follow to integrate Spring and Hibernate.

1. Add Hibernate-entity manager, Hibernate-core and Spring-ORM dependencies.
2. Create Model classes and corresponding DAO implementations for database operations. The DAO classes will use SessionFactory that will be injected by the Spring Bean configuration.
3. Note that you don’t need to use Hibernate Transaction Management, as you can leave it to the Spring declarative transaction management using @Transactional annotation.

### **Q15. What do you mean by Hibernate Configuration File?**

Hibernate Configuration File mainly contains database-specific configurations and are used to initialize SessionFactory. Some important parts of the Hibernate Configuration File are Dialect information, so that hibernate knows the database type and mapping file or class details.

## **Hibernate Interview Questions for intermediate**

### **Q16. Mention some important annotations used for Hibernate mapping?**

Hibernate supports JPA annotations. Some of the major annotations are:

1. **javax.persistence.Entity:** This is used with model classes to specify they are entity beans.
2. **javax.persistence.Table:** It is used with entity beans to define the corresponding table name in the database.
3. **javax.persistence.Access:** Used to define the access type, field or property. The default value is field and if you want Hibernate to use the getter/setter methods then you need to set it to a property.
4. **javax.persistence.Id:** Defines the primary key in the entity bean.
5. **javax.persistence.EmbeddedId:** It defines a composite primary key in the entity bean.
6. **javax.persistence.Column:** Helps in defining the column name in the database table.
7. **javax.persistence.GeneratedValue:**It defines the strategy to be used for the generation of the primary key. It is also used in conjunction with javax.persistence.GenerationType enum.

### **Q17. What is Session in Hibernate and how to get it?**

Hibernate Session is the interface between Java application layer and Hibernate. It is used to get a physical connection with the database. The Session object created is lightweight and designed to be instantiated each time an interaction is needed with the database. This Session provides methods to create, read, update and delete operations for a constant object. To get the Session, you can execute HQL queries, SQL native queries using the Session object.

### **Q18. What is Hibernate SessionFactory?**

SessionFactory is the factory class that is used to get the Session objects. The SessionFactory is a heavyweight object so usually, it is created during application startup and kept for later use. This SessionFactory is a thread-safe object which is used by all the threads of an application. If you are using multiple databases then you would have to create multiple SessionFactory objects.

### **Q19. What is the difference between openSession and getCurrentSession?**

This getCurrentSession() method returns the session bound to the context and for this to work, you need to configure it in Hibernate configuration file. Since this session object belongs to the context of Hibernate, it is okay if you don’t close it. Once the SessionFactory is closed, this session object gets closed.

[[](https://www.edureka.co/java-j2ee-training-course?utm_source=blogbanner&utm_campaign=curriculum)](https://www.edureka.co/java-j2ee-training-course?utm_source=blogbanner&utm_campaign=curriculum" \t "_blank)

### [Java Certification Training Course](https://www.edureka.co/java-j2ee-training-course?utm_source=blogbanner&utm_campaign=curriculum" \t "_blank)

* *[Instructor-led Sessions](https://www.edureka.co/java-j2ee-training-course?utm_source=blogbanner&utm_campaign=curriculum" \t "_blank)*
* *[Real-life Case Studies](https://www.edureka.co/java-j2ee-training-course?utm_source=blogbanner&utm_campaign=curriculum" \t "_blank)*
* *[Assignments](https://www.edureka.co/java-j2ee-training-course?utm_source=blogbanner&utm_campaign=curriculum" \t "_blank)*
* *[Lifetime Access](https://www.edureka.co/java-j2ee-training-course?utm_source=blogbanner&utm_campaign=curriculum" \t "_blank)*

[Explore Curriculum](https://www.edureka.co/java-j2ee-training-course?utm_source=blogbanner&utm_campaign=curriculum" \t "_blank)

openSession() method helps in opening a new session. You should close this session object once you are done with all the database operations. And also, you should open a new session for each request in a multi-threaded environment.

### **Q20. What do you mean by Hibernate configuration file?**

The following steps help in configuring Hibernate file:

1. First, identify the POJOs (Plain Old Java Objects) that have a database representation.
2. Identify which properties of POJOs need to be continued.
3. Annotate each of the POJOs in order to map the Java objects to columns in a database table.
4. Create a database schema using the schema export tool which uses an existing database, or you can create your own database schema.
5. Add Hibernate Java libraries to the application’s classpath.
6. Create a Hibernate XML configuration file that points to the database and the mapped classes.
7. In the Java application, you can create a Hibernate Configuration object that refers to your XML configuration file.
8. Also, build a Hibernate SessionFactory object from the Configuration object.
9. Retrieve the Hibernate Session objects from the SessionFactory and write down the data access logic for your application (create, retrieve, update, and delete).

### **Q21. What are the key components of a Hibernate configuration object?**

The configuration provides 2 key components, namely:

* Database Connection: This is handled by one or more configuration files.
* Class Mapping setup: It helps in creating the connection between Java classes and database tables.

### **Q22. Discuss the Collections in Hibernate**

Hibernate provides the facility to persist the Collections. A [*Collection*](https://www.edureka.co/blog/java-collections/) basically can be a List, Set, Map, Collection, Sorted Set, Sorted Map. java.util.List, java.util.Set, java.util.Collection, etc, are some of the real interface types to declared the persistent collection-value fields. Hibernate injects persistent Collections based on the type of interface. The collection instances generally behave like the types of value behavior.

### **Q23. What are the collection types in Hibernate?**

There are five collection types in hibernate used for one-to-many relationship mappings.

* Bag
* Set
* List
* Array
* Map

### **Q24. What is a Hibernate Template class?**

When you integrate Spring and Hibernate, Spring ORM provides two helper classes – HibernateDaoSupport and HibernateTemplate. The main reason to use them was to get two things, the Session from Hibernate and Spring Transaction Management. However, from Hibernate 3.0.1, you can use the SessionFactory getCurrentSession() method to get the current session. The major advantage of using this Template class is the **exception translation** but that can be achieved easily by using @Repository annotation with service classes.

### **Q25. What are the benefits of using Hibernate template?**

The following are the benefits of using this Hibernate template class:

* Automated Session closing ability.
* The interaction with the Hibernate Session is simplified.
* Exception handling is automated.

### **Q26. Which are the design patterns that are used in Hibernate framework?**

There are a few design patterns used in Hibernate Framework, namely:

* Domain Model Pattern: An object model of the domain that incorporates both behavior as well as data.
* Data Mapper: A layer of the map that moves data between objects and a database while keeping it independent of each other and the map itself.
* Proxy Pattern: It is used for lazy loading.
* Factory Pattern: Used in SessionFactory.

### **Q27. Define Hibernate Validator Framework**

Data validation is considered as an integral part of any application. Also, data validation is used in the presentation layer with the use of Javascript and the server-side code before processing. It occurs before persisting it in order to make sure it follows the correct format. Validation is a cross-cutting task, so we should try to keep it apart from the business logic. This Hibernate Validator provides the reference implementation of bean validation specs.

### **Q28. What is Dirty Checking in Hibernate?**

Hibernate incorporates Dirty Checking feature that permits developers and users to avoid time-consuming write actions. This Dirty Checking feature changes or updates fields that need to be changed or updated, while keeping the remaining fields untouched and unchanged.

### **Q29. How can you share your views on mapping description files?**

* Mapping description files are used by the Hibernate to configure functions.
* These files have the **\*.hbm** extension, which facilitates the mapping between database tables and Java class.
* Whether to use mapping description files or not this entirely depends on business entities.

**Q30. What is meant by Light Object Mapping?**

The means that the syntax is hidden from the business logic using specific design patterns. This is one of the valuable levels of ORM quality and this Light Object Mapping approach can be successful in case of applications where there are very fewer entities, or for applications having data models that are metadata-driven.

## **Hibernate Interview Questions for experienced**

### **Q31. What is meant by Hibernate tuning?**

Optimizing the performance of Hibernate applications is known as Hibernate tuning.

The performance tuning strategies for Hibernate are:

1. SQL Optimization
2. Session Management
3. Data Caching

### **Q32. What is Transaction Management in Hibernate? How does it work?**

Transaction Management is a property which is present in the Spring framework. Now, what role does it play in Hibernate?

Transaction Management is a process of managing a set of commands or statements. In hibernate, Transaction Management is done by transaction interface. It maintains abstraction from the transaction implementation (JTA, JDBC). A transaction is associated with Session and is instantiated by calling session.beginTransaction().

**Q33. How do you integrate Hibernate with Struts2 or Servlet web applications?**

You can integrate any Struts application with Hibernate. There are no extra efforts required.

1. Register a custom ServletContextListener.
2. In the ServletContextListener class, first, initialize the Hibernate Session, store it in the servlet context.
3. Action class helps in getting the Hibernate Session from the servlet context, and perform other Hibernate task as normal.

**Q34. What are the different states of a persistent entity?**

It may exist in one of the following 3 states:

* Transient: This is not associated with the Session and has no representation in the database.
* Persistent: You can make a transient instance persistent by associating it with a Session.
* Detached: If you close the Hibernate Session, the persistent instance will become a detached instance.

**Q35. How can the primary key be created by using Hibernate?**

A Primary key is a special relational database table column designated to uniquely identify all table records. It is specified in the configuration file hbm.xml. The generator can also be used to specify how a Primary key can be created in the database.

|  |  |
| --- | --- |
| 1  2  3  4 | <id name="ClassID" type="string" >  <column name= "columnID" length="10" >  <generator/>  </id> |

**Q36. Explain about Hibernate Proxy and how it helps in Lazy loading?**

* Hibernate uses a proxy object in order to support Lazy loading.
* When you try loading data from tables, Hibernate doesn’t load all the mapped objects.
* After you reference a child object through getter methods, if the linked entity is not present in the session cache, then the proxy code will be entered to the database and load the linked object.
* It uses Java assist to effectively and dynamically generate sub-classed implementations of your entity objects.

**Q37. How can we see Hibernate generated SQL on console?**

In order to view the SQL on a console, you need to add following in Hibernate configuration file to enable viewing SQL on the console for debugging purposes:

|  |  |
| --- | --- |
| 1 | <property name="show\_sql">true</property> |

**Q38. What is Query Cache in Hibernate?**

Hibernate implements a separate cache region for queries resultset that integrates with the Hibernate second-level cache. This is also an optional feature and requires a few more steps in code.

### Programming & Frameworks Training

### [FULL STACK WEB DEVELOPMENT INTERNSHIP PROGRAM](https://www.edureka.co/full-stack-web-development?utm_source=blogbanner&utm_campaign=trending_course" \o "Full Stack Web Development Internship Program" \t "_blank)

### [Full Stack Web Development Internship Program](https://www.edureka.co/full-stack-web-development?utm_source=blogbanner&utm_campaign=trending_course" \o "Full Stack Web Development Internship Program" \t "_blank)

*[Reviews](https://www.edureka.co/full-stack-web-development?utm_source=blogbanner&utm_campaign=trending_course" \o "Full Stack Web Development Internship Program" \t "_blank)*

**[5](https://www.edureka.co/full-stack-web-development?utm_source=blogbanner&utm_campaign=trending_course" \o "Full Stack Web Development Internship Program" \t "_blank)**[(4161)](https://www.edureka.co/full-stack-web-development?utm_source=blogbanner&utm_campaign=trending_course" \o "Full Stack Web Development Internship Program" \t "_blank)

### [JAVA CERTIFICATION TRAINING COURSE](https://www.edureka.co/java-j2ee-training-course?utm_source=blogbanner&utm_campaign=trending_course" \o "Java Certification Training Course" \t "_blank)

### [Java Certification Training Course](https://www.edureka.co/java-j2ee-training-course?utm_source=blogbanner&utm_campaign=trending_course" \o "Java Certification Training Course" \t "_blank)

*[Reviews](https://www.edureka.co/java-j2ee-training-course?utm_source=blogbanner&utm_campaign=trending_course" \o "Java Certification Training Course" \t "_blank)*

**[4](https://www.edureka.co/java-j2ee-training-course?utm_source=blogbanner&utm_campaign=trending_course" \o "Java Certification Training Course" \t "_blank)**[(66096)](https://www.edureka.co/java-j2ee-training-course?utm_source=blogbanner&utm_campaign=trending_course" \o "Java Certification Training Course" \t "_blank)

### [NODE.JS CERTIFICATION TRAINING COURSE](https://www.edureka.co/nodejs-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "Node.js Certification Training Course" \t "_blank)

### [Node.js Certification Training Course](https://www.edureka.co/nodejs-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "Node.js Certification Training Course" \t "_blank)

*[Reviews](https://www.edureka.co/nodejs-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "Node.js Certification Training Course" \t "_blank)*

**[5](https://www.edureka.co/nodejs-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "Node.js Certification Training Course" \t "_blank)**[(8499)](https://www.edureka.co/nodejs-certification-training?utm_source=blogbanner&utm_campaign=trending_course" \o "Node.js Certification Training Course" \t "_blank)

### [PYTHON SCRIPTING CERTIFICATION TRAINING](https://www.edureka.co/python-scripting?utm_source=blogbanner&utm_campaign=trending_course" \o "Python Scripting Certification Training" \t "_blank)

### [Python Scripting Certification Training](https://www.edureka.co/python-scripting?utm_source=blogbanner&utm_campaign=trending_course" \o "Python Scripting Certification Training" \t "_blank)

*[Reviews](https://www.edureka.co/python-scripting?utm_source=blogbanner&utm_campaign=trending_course" \o "Python Scripting Certification Training" \t "_blank)*

**[5](https://www.edureka.co/python-scripting?utm_source=blogbanner&utm_campaign=trending_course" \o "Python Scripting Certification Training" \t "_blank)**[(12779)](https://www.edureka.co/python-scripting?utm_source=blogbanner&utm_campaign=trending_course" \o "Python Scripting Certification Training" \t "_blank)

### [ADVANCED JAVA CERTIFICATION TRAINING](https://www.edureka.co/advanced-java-sp?utm_source=blogbanner&utm_campaign=trending_course" \o "Advanced Java Certification Training" \t "_blank)

### [Advanced Java Certification Training](https://www.edureka.co/advanced-java-sp?utm_source=blogbanner&utm_campaign=trending_course" \o "Advanced Java Certification Training" \t "_blank)

*[Reviews](https://www.edureka.co/advanced-java-sp?utm_source=blogbanner&utm_campaign=trending_course" \o "Advanced Java Certification Training" \t "_blank)*

**[5](https://www.edureka.co/advanced-java-sp?utm_source=blogbanner&utm_campaign=trending_course" \o "Advanced Java Certification Training" \t "_blank)**[(5229)](https://www.edureka.co/advanced-java-sp?utm_source=blogbanner&utm_campaign=trending_course" \o "Advanced Java Certification Training" \t "_blank)

### [SPRING FRAMEWORK CERTIFICATION COURSE](https://www.edureka.co/spring-certification-course?utm_source=blogbanner&utm_campaign=trending_course" \o "Spring Framework Certification Course" \t "_blank)

### [Spring Framework Certification Course](https://www.edureka.co/spring-certification-course?utm_source=blogbanner&utm_campaign=trending_course" \o "Spring Framework Certification Course" \t "_blank)

*[Reviews](https://www.edureka.co/spring-certification-course?utm_source=blogbanner&utm_campaign=trending_course" \o "Spring Framework Certification Course" \t "_blank)*

**[5](https://www.edureka.co/spring-certification-course?utm_source=blogbanner&utm_campaign=trending_course" \o "Spring Framework Certification Course" \t "_blank)**[(10178)](https://www.edureka.co/spring-certification-course?utm_source=blogbanner&utm_campaign=trending_course" \o "Spring Framework Certification Course" \t "_blank)

### [PYTHON DJANGO CERTIFICATION TRAINING COURSE](https://www.edureka.co/python-django?utm_source=blogbanner&utm_campaign=trending_course" \o "Python Django Certification Training Course" \t "_blank)

### [Python Django Certification Training Course](https://www.edureka.co/python-django?utm_source=blogbanner&utm_campaign=trending_course" \o "Python Django Certification Training Course" \t "_blank)

*[Reviews](https://www.edureka.co/python-django?utm_source=blogbanner&utm_campaign=trending_course" \o "Python Django Certification Training Course" \t "_blank)*

**[5](https://www.edureka.co/python-django?utm_source=blogbanner&utm_campaign=trending_course" \o "Python Django Certification Training Course" \t "_blank)**[(7472)](https://www.edureka.co/python-django?utm_source=blogbanner&utm_campaign=trending_course" \o "Python Django Certification Training Course" \t "_blank)

### [PHP & MYSQL WITH MVC FRAMEWORKS CERTIFICATION TRAINING](https://www.edureka.co/php-mysql-self-paced?utm_source=blogbanner&utm_campaign=trending_course" \o "PHP & MySQL with MVC Frameworks Certification Training" \t "_blank)

### [PHP & MySQL with MVC Frameworks Certification Training](https://www.edureka.co/php-mysql-self-paced?utm_source=blogbanner&utm_campaign=trending_course" \o "PHP & MySQL with MVC Frameworks Certification Training" \t "_blank)

*[Reviews](https://www.edureka.co/php-mysql-self-paced?utm_source=blogbanner&utm_campaign=trending_course" \o "PHP & MySQL with MVC Frameworks Certification Training" \t "_blank)*

**[5](https://www.edureka.co/php-mysql-self-paced?utm_source=blogbanner&utm_campaign=trending_course" \o "PHP & MySQL with MVC Frameworks Certification Training" \t "_blank)**[(4447)](https://www.edureka.co/php-mysql-self-paced?utm_source=blogbanner&utm_campaign=trending_course" \o "PHP & MySQL with MVC Frameworks Certification Training" \t "_blank)

### [MICROSOFT .NET FRAMEWORK CERTIFICATION TRAINING](https://www.edureka.co/microsoft-dotnet-framework-self-paced?utm_source=blogbanner&utm_campaign=trending_course" \o "Microsoft .NET Framework Certification Training" \t "_blank)

### [Microsoft .NET Framework Certification Training](https://www.edureka.co/microsoft-dotnet-framework-self-paced?utm_source=blogbanner&utm_campaign=trending_course" \o "Microsoft .NET Framework Certification Training" \t "_blank)

*[Reviews](https://www.edureka.co/microsoft-dotnet-framework-self-paced?utm_source=blogbanner&utm_campaign=trending_course" \o "Microsoft .NET Framework Certification Training" \t "_blank)*

**[5](https://www.edureka.co/microsoft-dotnet-framework-self-paced?utm_source=blogbanner&utm_campaign=trending_course" \o "Microsoft .NET Framework Certification Training" \t "_blank)**[(5132)](https://www.edureka.co/microsoft-dotnet-framework-self-paced?utm_source=blogbanner&utm_campaign=trending_course" \o "Microsoft .NET Framework Certification Training" \t "_blank)

Next

***Note:*** This is only useful for queries that are run frequently with the same parameters.

**Q39. What is the benefit of Native SQL query support in Hibernate?**

Hibernate provides an option to execute Native SQL queries through the use of the [SQLQuery](https://www.edureka.co/blog/insert-query-sql/)object. For normal scenarios, it is however not the recommended approach because you might lose other benefits like Association and Hibernate first-level caching.

Native SQL Query comes handy when you want to execute database-specific queries that are not supported by Hibernate API such query hints or the Connect keyword in Oracle Database.

**Q40. What is Named SQL Query?**

Hibernate provides another important feature called Named Query using which you can define at a central location and use them anywhere in the code.

You can create named queries for both HQL as well as for Native SQL. These Named Queries can be defined in Hibernate mapping files with the help of JPA annotations @NamedQuery and @NamedNativeQuery.

**Q41. When do you use merge() and update() in Hibernate?**

This is one of the tricky Hibernate Interview Questions asked.

update(): If you are sure that the Hibernate Session does not contain an already persistent instance with the same id .  
*merge():*  Helps in merging your modifications at any time without considering the state of the Session.

**Q42. Difference between get() vs load() method in Hibernate?**

This is one of the most frequently asked Hibernate Interview Questions. The key difference between the get() and load() method is:

load(): It will throw an exception if an object with an ID passed to them is not found.  
get():  Will return null.

load(): It can return proxy without hitting the database unless required.  
get(): It always goes to the database.

So sometimes using load() can be faster than the get() method.

**Q43. Difference between the first and second level cache in Hibernate?**

The first-level cache is maintained at Session level while the second level cache is maintained at a SessionFactory level and is shared by all sessions.

**Q44. Difference between Session and SessionFactory in Hibernate?**

This is yet another popular Hibernate Interview Question asked.

* *A Session* is a single-threaded, short-lived object. It provides the first-level cache.
* SessionFactory is immutable and shared by all Session. It also lives until the Hibernate is running. It also provides the second-level cache.

**Q45. Difference between save() and saveOrUpdate() method of Hibernate?**

Even though save() and saveOrUpdate() method is used to store an object into Database, the key difference between them is that save() can only **Insert** records but saveOrUpdate() can either Insert or Update records.

**Q46. Difference between sorted and ordered collection in Hibernate?**

sorted collection sort the data in JVM’s heap memory using Java’s collection framework sorting methods. The ordered collection is sorted using order by clause in the database itself.

***Note:*** A sorted collection is more suited for small dataset but for a large dataset, it’s better to use ordered collection to avoid

**Q47. Difference between the transient, persistent and detached state in Hibernate?**

**Transient state:** New objects are created in the Java program but are not associated with any Hibernate Session.

**Persistent state:** An object which is associated with a Hibernate session is called Persistent object. While an object which was earlier associated with Hibernate session but currently it’s not associate is known as a detached object. You can call save() or persist() method to store those object into the database and bring them into the Persistent state.

[[](https://www.edureka.co/java-j2ee-training-course?utm_source=blogbanner&utm_campaign=batches)](https://www.edureka.co/java-j2ee-training-course?utm_source=blogbanner&utm_campaign=batches" \t "_blank)

### [Java Certification Training Course](https://www.edureka.co/java-j2ee-training-course?utm_source=blogbanner&utm_campaign=batches" \t "_blank)

[Weekday / Weekend BatchesSee Batch Details](https://www.edureka.co/java-j2ee-training-course?utm_source=blogbanner&utm_campaign=batches" \t "_blank)

**Detached state**: You can re-attach a detached object to Hibernate sessions by calling either update() or saveOrUpdate() method.

**Q48. Difference between managed associations and Hibernate associations?**

**Managed associations:** Relate to container management persistence and are bi-directional.

**Hibernate Associations:** These associations are unidirectional.

V

Database

# Indexing in Databases:

## What is Indexing?

**Indexing** is a data structure technique which allows you to quickly retrieve records from a database file. An Index is a small table having only two columns. The first column comprises a copy of the primary or candidate key of a table. Its second column contains a set of [pointers](https://www.guru99.com/c-pointers.html) for holding the address of the disk block where that specific key value stored.

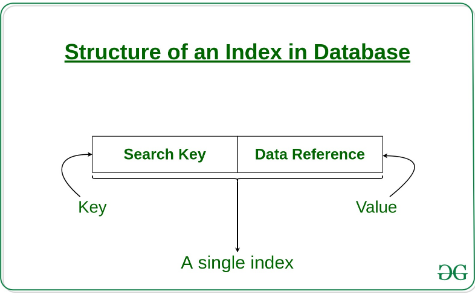
An index –

* Takes a search key as input
* Efficiently returns a collection of matching records.

# It is a data structure technique which is used to quickly locate and access the data in a database

Indexes are created using a few database columns.

* The first column is the **Search key** that contains a copy of the primary key or candidate key of the table. These values are stored in sorted order so that the corresponding data can be accessed quickly.   
  *Note: The data may or may not be stored in sorted order.*
* The second column is the **Data Reference** or **Pointer** which contains a set of pointers ho

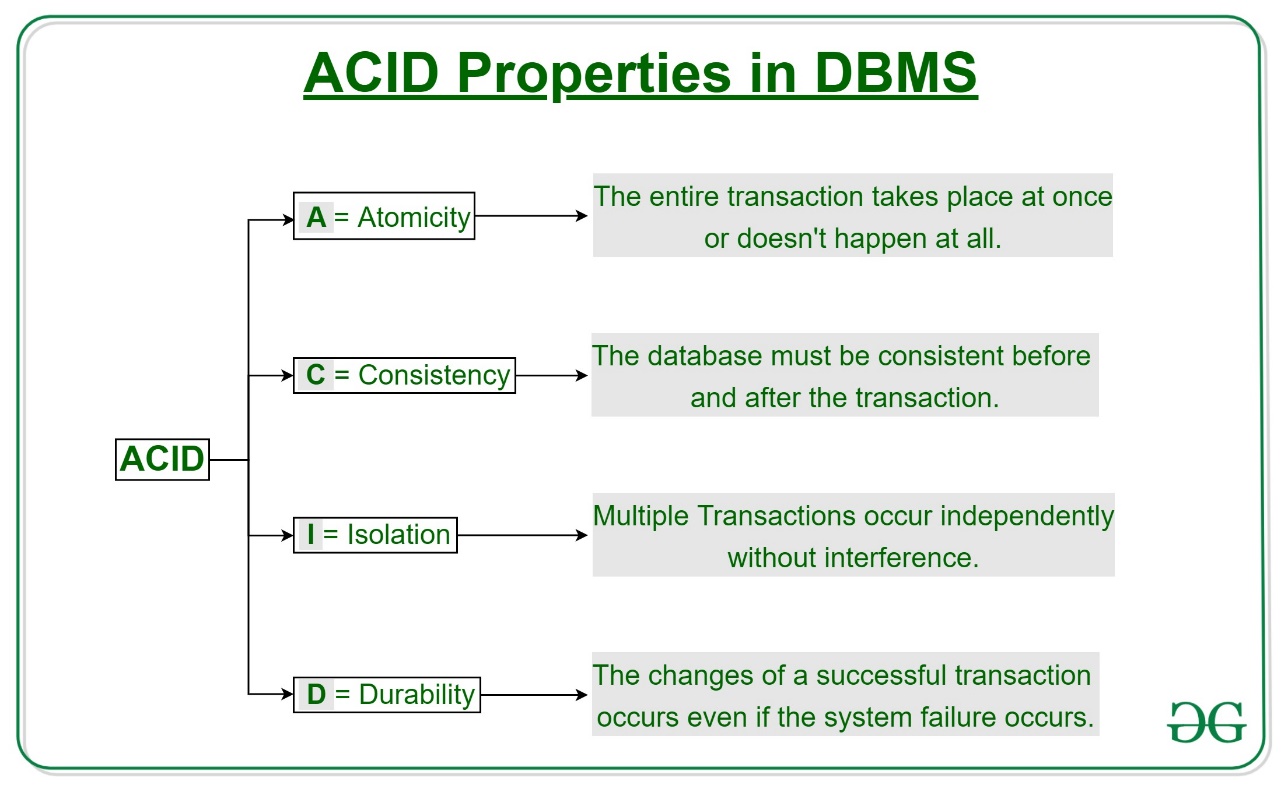


The indexing has various attributes:

* **Access Types**: This refers to the type of access such as value based search, range access, etc.
* **Access Time**: It refers to the time needed to find particular data element or set of elements.
* **Insertion Time**: It refers to the time taken to find the appropriate space and insert a new data.
* **Deletion Time**: Time taken to find an item and delete it as well as update the index structure.
* **Space Overhead**: It refers to the additional space required by the index

# ACID Properties in DBMS

ACID Properties in SQL Server ensure Data Integrity during a transaction. The ACID is an acronym for Atomicity, Consistency, Isolation, and Durability.



**Some important points:**

| **Property** | **Responsibility for maintaining properties** |
| --- | --- |
| Atomicity | Transaction Manager |
| Consistency | Application programmer |
| Isolation | Concurrency Control Manager |
| Durability | Recovery Manager |

**1. Relational Database :**   
RDBMS stands for Relational Database Management Systems. It is most popular database. In it, data is store in the form of row that is in the form of tuple. It contain numbers of table and data can be easily accessed because data is store in the table. This Model was proposed by E.F. Codd.

**2. NoSQL :**   
NoSQL Database stands for a non-SQL database. NoSQL database doesn’t use table to store the data like relational database. It is used for storing and fetching the data in database and generally used to store the large amount of data. It supports query language and provides better performance.

**Difference between Relational database and NoSQL :**

|  |  |
| --- | --- |
| Relational Database | NoSQL |
| It is used to handle data coming in low velocity. | It is used to handle data coming in high velocity. |
| It gives only read scalability. | It gives both read and write scalability. |
| It manages structured data. | It manages all type of data. |
| Data arrives from one or few locations. | Data arrives from many locations. |
| It supports complex transactions. | It supports simple transactions. |
| It has single point of failure. | No single point of failure. |
| It handles data in less volume. | It handles data in high volume. |
| Transactions written in one location. | Transactions written in many locations. |
| support ACID properties compliance | doesn’t support ACID properties |
| Its difficult to make changes in database once it is defined | Enables easy and frequent changes to database |
| schema  is mandatory to store the data | schema design is not required |
| Deployed in vertical fashion. | Deployed in Horizontal fashion. |

### Features of relational databases

* They work with structured data.
* Relationships in the system have constraints, which promotes a high level of data integrity.
* There are limitless indexing capabilities, which results in faster query response times.
* They are excellent at keeping data transactions secure.
* They provide the ability to write complex SQL queries for data analysis and reporting.
* Their models can ensure and enforce business rules at the data layer adding a level of data integrity not found in a non-relational database.
* They are table and row oriented.
* They Use SQL (structured query language) for shaping and manipulating data, which is very powerful.
* SQL database examples: [MySql](https://www.pluralsight.com/paths/mysql), [Oracle](https://www.pluralsight.com/courses/oracle-database-12c-fundamentals), Sqlite, Postgres and MS-SQL. NoSQL database examples: MongoDB, [BigTable](https://www.pluralsight.com/courses/google-bigtable-architecting-big-data-solutions), Redis, RavenDb, Cassandra, Hbase, Neo4j and CouchDb.
* SQL databases are best fit for heavy duty transactional type applications.

### Features of non-relational databases

* They have the ability to store large amounts of data with little structure.
* They provide scalability and flexibility to meet changing business requirements.
* They provide schema-free or schema-on-read options.
* They have the ability to capture all types of data “Big Data” including unstructured data.
* They are document oriented.
* NoSQL or non-relational databases examples: MongoDB, Apache Cassandra, Redis, Couchbase and Apache HBase.
* They are best for Rapid Application Development. NoSQL is the best selection for flexible data storage with little to no structure limitations.
* They provide flexible data model with the ability to easily store and combine data of any structure without the need to modify a schema.

SOLID

They all serve the same purpose:

"To create understandable, readable, and testable code that many developers can collaboratively work on."

Let's look at each principle one by one. Following the SOLID acronym, they are:

* The **S**ingle Responsibility Principle
* The **O**pen-Closed Principle
* The **L**iskov Substitution Principle
* The **I**nterface Segregation Principle
* The **D**ependency Inversion Principle

## The Single Responsibility Principle

The Single Responsibility Principle states that **a class should do one thing and therefore it should have only a single reason to change**.

## Open-Closed Principle

The Open-Closed Principle requires that **classes should be open for extension and closed to modification.**

## Liskov Substitution Principle

The Liskov Substitution Principle states that subclasses should be substitutable for their base classes.

This means that, given that class B is a subclass of class A, we should be able to pass an object of class B to any method that expects an object of class A and the method should not give any weird output in that case

## Interface Segregation Principle

Segregation means keeping things separated, and the Interface Segregation Principle is about separating the interfaces.

The principle states that many client-specific interfaces are better than one general-purpose interface. Clients should not be forced to implement a function they do no need.

## Dependency Inversion Principle

The Dependency Inversion principle states that our classes should depend upon interfaces or abstract classes instead of concrete classes and functions.

In his [article](https://fi.ort.edu.uy/innovaportal/file/2032/1/design_principles.pdf)(2000), Uncle Bob summarizes this principle as follows:

"If the OCP states the goal of OO architecture, the DIP states the primary mechanism".

These two principles are indeed related and we have applied this pattern before while we were discussing the Open-Closed Principle.