**SME Expected Questions:**

1. **Indexing**

Indexes are **special lookup tables** that the database search engine can use to speed up data retrieval. Simply put, an index is a pointer to data in a table. An index in a database is very similar to an index in the back of a book.

For example, if you want to reference all pages in a book that discusses a certain topic, you first refer to the index, which lists all the topics alphabetically and are then referred to one or more specific page numbers.

An index helps to speed up **SELECT** queries and **WHERE** clauses, but it slows down data input, with the **UPDATE** and the **INSERT** statements. Indexes can be created or dropped with no effect on the data.

<https://www.sqlshack.com/what-is-the-difference-between-clustered-and-non-clustered-indexes-in-sql-server/>

1. **Stream API(Examples)**

Stream is a new abstract layer introduced in Java 8.

Stream represents a sequence of objects from a source, which supports aggregate operations. Following are the characteristics of a Stream –

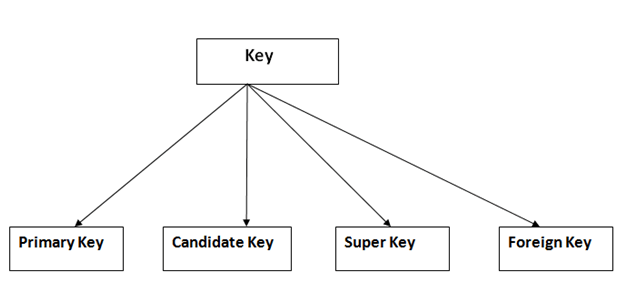
* **Sequence of elements** − A stream provides a set of elements of specific type in a sequential manner. A stream gets/computes elements on demand. It never stores the elements.
* **Source** − Stream takes Collections, Arrays, or I/O resources as input source.
* **Aggregate operations** − Stream supports aggregate operations like filter, map, limit, reduce, find, match, and so on.
* **Pipelining** − Most of the stream operations return stream itself so that their result can be pipelined. These operations are called intermediate operations and their function is to take input, process them, and return output to the target. collect() method is a terminal operation which is normally present at the end of the pipelining operation to mark the end of the stream.
* **Automatic iterations** − Stream operations do the iterations internally over the source elements provided, in contrast to Collections where explicit iteration is required.

<https://winterbe.com/posts/2014/07/31/java8-stream-tutorial-examples/>

<https://www.tutorialspoint.com/java8/java8_streams.htm>

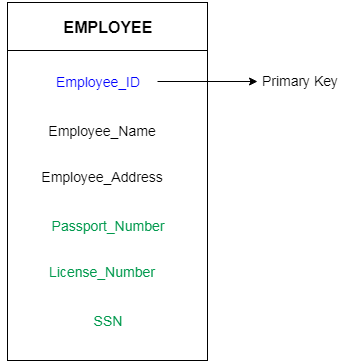
1. **keys in dbms**

* Keys play an important role in the relational database.
* It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables.



1. Primary key

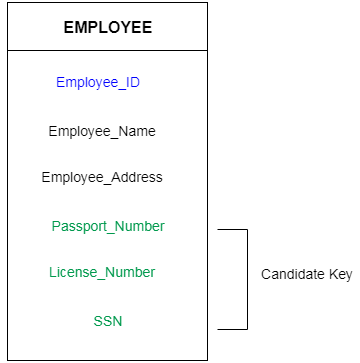
* It is the first key which is used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys as we saw in PERSON table. The key which is most suitable from those lists become a primary key.
* In the EMPLOYEE table, ID can be primary key since it is unique for each employee. In the EMPLOYEE table, we can even select License\_Number and Passport\_Number as primary key since they are also unique.
* For each entity, selection of the primary key is based on requirement and developers.



2. Candidate key

* A candidate key is an attribute or set of an attribute which can uniquely identify a tuple.
* The remaining attributes except for primary key are considered as a candidate key. The candidate keys are as strong as the primary key.

**For example:** In the EMPLOYEE table, id is best suited for the primary key. Rest of the attributes like SSN, Passport\_Number, and License\_Number, etc. are considered as a candidate key.



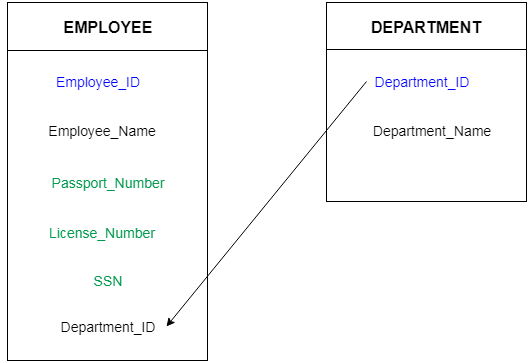
### 3. Super Key

Super key is a set of an attribute which can uniquely identify a tuple. Super key is a superset of a candidate key.

**For example:** In the above EMPLOYEE table, for(EMPLOEE\_ID, EMPLOYEE\_NAME) the name of two employees can be the same, but their EMPLYEE\_ID can't be the same. Hence, this combination can also be a key.

The super key would be EMPLOYEE-ID, (EMPLOYEE\_ID, EMPLOYEE-NAME), etc.

4. Foreign key

* Foreign keys are the column of the table which is used to point to the primary key of another table.
* In a company, every employee works in a specific department, and employee and department are two different entities. So we can't store the information of the department in the employee table. That's why we link these two tables through the primary key of one table.
* We add the primary key of the DEPARTMENT table, Department\_Id as a new attribute in the EMPLOYEE table.
* Now in the EMPLOYEE table, Department\_Id is the foreign key, and both the tables are related.
* 

1. **OOPS**

**Object** means a real-world entity such as a pen, chair, table, computer, watch, etc. **Object-Oriented Programming** is a methodology or paradigm to design a program using classes and objects. It simplifies software development and maintenance by providing some concepts:

* [Object](https://www.javatpoint.com/object-and-class-in-java)
* Class
* [Inheritance](https://www.javatpoint.com/inheritance-in-java)
* [Polymorphism](https://www.javatpoint.com/runtime-polymorphism-in-java)
* [Abstraction](https://www.javatpoint.com/abstract-class-in-java)
* [Encapsulation](https://www.javatpoint.com/encapsulation)



<https://www.javatpoint.com/java-oops-concepts>

1. **Super key**

A super key is a set of one or more attributes (columns), which can uniquely identify a row in a table. Often [DBMS beginners](https://beginnersbook.com/2015/04/dbms-tutorial/) get confused between super key and [candidate key](https://beginnersbook.com/2015/04/candidate-key-in-dbms/), so we will also discuss candidate key and its relation with super key in this article.

## How candidate key is different from super key?

Answer is simple – Candidate keys are selected from the set of super keys, the only thing we take care while selecting candidate key is: It should not have any redundant attribute. That’s the reason they are also termed as minimal super key.

## Super key vs Candidate Key

1. First you have to understand that all the candidate keys are super keys. This is because the candidate keys are chosen out of the super keys.  
2. How we choose candidate keys from the set of super keys? We look for those keys from which we cannot remove any fields. In the above example, we have not chosen {Emp\_SSN, Emp\_Name} as candidate key because {Emp\_SSN} alone can identify a unique row in the table and Emp\_Name is redundant.

1. **Array list (adv and disadv)**

 the drawbacks of arrays.

* Arrays are of fixed length. You can not change the size of the arrays once they are created.
* You can not accommodate an extra element in an array after they are created.
* Memory is allocated to an array during it’s creation only, much before the actual elements are added to it.

Here are some advantages of using ArrayList over arrays.

1. You can define ArrayList as **re-sizable array**. Size of the ArrayList is not fixed. ArrayList can grow and shrink dynamically.
2. Elements can be inserted at or deleted from a particular position.
3. ArrayList class has many methods to manipulate the stored objects.
4. If generics are not used, ArrayList can hold any type of objects
5. Many are of the assumption that multiple insertion and removal operations on ArrayList will decrease the performance of an application. But, there will be no significant change in the performance of an application if you use ArrayList instead of arrays.
6. You can traverse an ArrayList in both the directions – forward and backward using ListIterator.
7. ArrayList can hold multiple null elements.
8. ArrayList can hold duplicate elements.

<https://www.quora.com/What-are-the-advantages-and-disadvantages-of-ArrayList-in-collection>

1. **Hashmap,treeset,hashset,hashtable,linkedlist, linked hash map, vector, arraylistlinkedset,features and comparsion**

[**https://dzone.com/articles/hashmap-vs-treemap-vs**](https://dzone.com/articles/hashmap-vs-treemap-vs)

1. **JAVA8 features**

[**https://www.javatpoint.com/java-8-features**](https://www.javatpoint.com/java-8-features)

1. **Lambda Expressions(in detail)**

Lambda expression is a new and important feature of Java which was included in Java SE 8. It provides a clear and concise way to represent one method interface using an expression. It is very useful in collection library. It helps to iterate, filter and extract data from collection.

The Lambda expression is used to provide the implementation of an interface which has functional interface. It saves a lot of code. In case of lambda expression, we don't need to define the method again for providing the implementation. Here, we just write the implementation code.

Java lambda expression is treated as a function, so compiler does not create .class file.

## **Functional Interface**

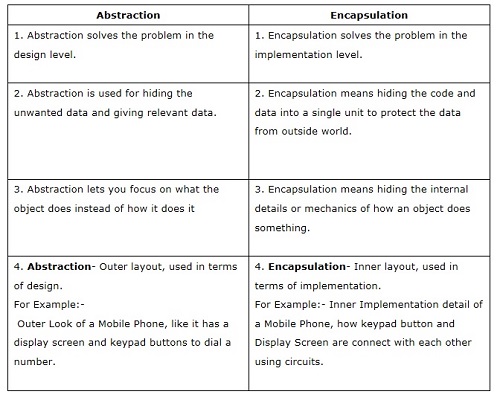
Lambda expression provides implementation of functional interface. An interface which has only one abstract method is called functional interface. Java provides an anotation @FunctionalInterface, which is used to declare an interface as functional interface.

## **Why use Lambda Expression**

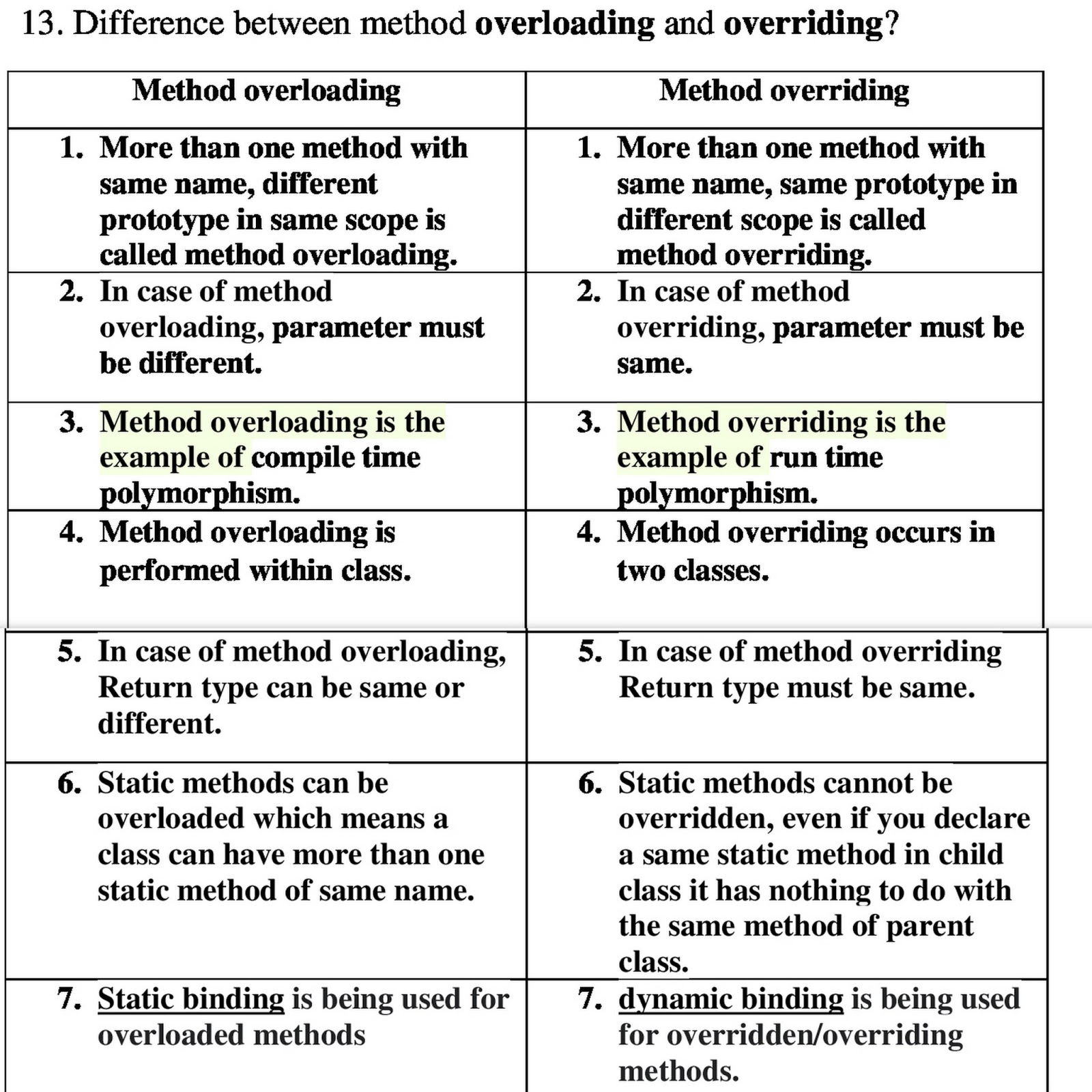
1. To provide the implementation of Functional interface.
2. Less coding.
3. **diff between final , finally, finalize()**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **final** | **finally** | **finalize** |
| 1) | Final is used to apply restrictions on class, method and variable. Final class can't be inherited, final method can't be overridden and final variable value can't be changed. | Finally is used to place important code, it will be executed whether exception is handled or not. | Finalize is used to perform clean up processing just before object is garbage collected. |
| 2) | Final is a keyword. | Finally is a block. | Finalize is a method. |

1. There are many differences between final, finally and finalize. A list of differences between final, finally and finalize are given below:
2. **diff b/w abstraction and encapsulation**



1. **overloading and overriding**



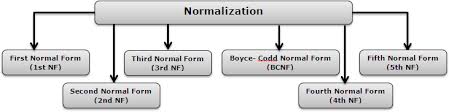
[**https://www.programcreek.com/2009/02/overriding-and-overloading-in-java-with-examples/#:~:text=Overloading%20occurs%20when%20two%20or,(i.e.%2C%20method%20signature)**](https://www.programcreek.com/2009/02/overriding-and-overloading-in-java-with-examples/#:~:text=Overloading%20occurs%20when%20two%20or,(i.e.%2C%20method%20signature))**.**

1. **Normalization and types of normalization**

# **Normalization**

* Normalization is the process of organizing the data in the database.
* Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate the undesirable characteristics like Insertion, Update and Deletion Anomalies.
* Normalization divides the larger table into the smaller table and links them using relationship.
* The normal form is used to reduce redundancy from the database table.

|  |  |
| --- | --- |
| **Normal Form** | **Description** |
| [1NF](https://www.javatpoint.com/dbms-first-normal-form) | A relation is in 1NF if it contains an atomic value. |
| [2NF](https://www.javatpoint.com/dbms-second-normal-form) | A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key. |
| [3NF](https://www.javatpoint.com/dbms-third-normal-form) | A relation will be in 3NF if it is in 2NF and no transition dependency exists. |
| [4NF](https://www.javatpoint.com/dbms-forth-normal-form) | A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency. |
| [5NF](https://www.javatpoint.com/dbms-fifth-normal-form) | A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless. |

****

1. **Polymorphism real example**

**Real life example** of **polymorphism**: A person at the same time can have different characteristic. Like a man at the same time is a father, a husband, an employee. ... This is called **polymorphism**. **Polymorphism** is considered as one of the important features of Object Oriented Programming

[**https://www.studytonight.com/post/polymorphism-view-from-a-real-world**](https://www.studytonight.com/post/polymorphism-view-from-a-real-world)

1. **what is ideal class and ideal access specifier**

**A picture containing green, bus, lot, grass

Description automatically generated**

[**https://dzone.com/articles/rule-30-%E2%80%93-when-method-class-or#:~:text=a)%20Methods%20should%20not%20have,up%20to%2027%2C000%20code%20lines**](https://dzone.com/articles/rule-30-%E2%80%93-when-method-class-or#:~:text=a)%20Methods%20should%20not%20have,up%20to%2027%2C000%20code%20lines)**.**

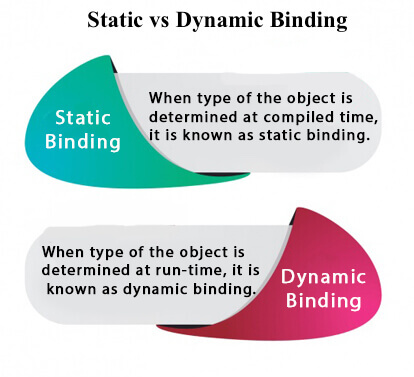
1. **Static and dynamic binding explanation**

Association of method call to the method body is known as binding. There are two types of binding: **Static Binding** that happens at compile time and **Dynamic Binding** that happens at runtime. Before I explain static and dynamic binding in java, lets see few terms that will help you understand this concept better.

## Static Binding vs Dynamic Binding

## Lets discuss the **difference between static and dynamic binding in Java**.

1. Static binding happens at compile-time while dynamic binding happens at runtime.
2. Binding of private, static and final methods always happen at compile time since these methods cannot be overridden. When the method overriding is actually happening and the reference of parent type is assigned to the object of child class type then such binding is resolved during runtime.
3. The binding of [overloaded methods](https://beginnersbook.com/2013/05/method-overloading/) is static and the binding of overridden methods is dynamic.



1. **need for default constructors nd are constructor mandatory?**

Java doesn't require a constructor when we create a class.   However, it's important to know what happens under the hood when no constructors are explicitly defined The compiler automatically provides a public no-argument constructor for any class without constructors.   This is called the *default constructor*.If we do explicitly declare a constructor of *any* form, then this automatic insertion by the compiler won't occur.

1. **worm long form in java**

**Worm: A self-replicating program**

* The major difference between a virus and a worm is that worm does not attach itself to other existing program as viruses do .
* Worms spread across networks due to poor security of the infected computers.
* Asthis type of infection runs by itself it can have devastating impacts.
* **Worm Viruses Include:**lovgate.F, sobig.D ,trile. C

[**https://www.geeksforgeeks.org/worms-viruses-and-beyond/**](https://www.geeksforgeeks.org/worms-viruses-and-beyond/)

1. **benefit of platform independency in java**

One of the most significant advantages of Java is its ability to move easily from one computer system to another. The ability to run the same program on many different systems is crucial to World Wide Web software, and Java succeeds at this by being platform-independent at both the source and binary levels.

[**https://www.artima.com/insidejvm/ed2/platindepP.html**](https://www.artima.com/insidejvm/ed2/platindepP.html)

1. **joins in details and diff in different joins nd where we use joins**

## **Different Types of SQL JOINs**

Here are the different types of the JOINs in SQL:

* **(INNER) JOIN**: Returns records that have matching values in both tables
* **LEFT (OUTER) JOIN**: Returns all records from the left table, and the matched records from the right table
* **RIGHT (OUTER) JOIN**: Returns all records from the right table, and the matched records from the left table
* **FULL (OUTER) JOIN**: Returns all records when there is a match in either left or right table

1. **why java do not have pointers**

**Java doesn’t have pointers; Java has references.**

**Reference:** A reference is a variable that refers to something else and can be used as an alias for that something else

Java doesn’t have pointers (in the C/C++ sense) because it doesn’t need them for general purpose OOP programming. Furthermore, adding pointers to Java would undermine security and robustness and make the language more complex.

[**https://www.geeksforgeeks.org/is-there-any-concept-of-pointers-in-java/#:~:text=So%20overall%20Java%20doesn't,make%20the%20language%20more%20complex**](https://www.geeksforgeeks.org/is-there-any-concept-of-pointers-in-java/#:~:text=So%20overall%20Java%20doesn't,make%20the%20language%20more%20complex)**.**

1. **predefined oracle exception**

[**https://www.tutorialspoint.com/plsql/plsql\_exceptions.htm**](https://www.tutorialspoint.com/plsql/plsql_exceptions.htm)

1. **what is jdbc(read in detail)**

[**https://www.javatpoint.com/java-jdbc**](https://www.javatpoint.com/java-jdbc)

1. **DDL,DML,CRUD**

**DDL(Data Definition Language) :**DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database.

**Examples of DDL commands:**

* [**CREATE**](https://www.geeksforgeeks.org/sql-create/) – is used to create the database or its objects (like table, index, function, views, store procedure and triggers).
* [**DROP**](https://www.geeksforgeeks.org/sql-drop-truncate/) – is used to delete objects from the database.
* [**ALTER**](https://www.geeksforgeeks.org/sql-alter-add-drop-modify/)-is used to alter the structure of the database.
* [**TRUNCATE**](https://www.geeksforgeeks.org/sql-drop-truncate/)–is used to remove all records from a table, including all spaces allocated for the records are removed.
* [**COMMENT**](https://www.geeksforgeeks.org/sql-comments/) –is used to add comments to the data dictionary.
* [**RENAME**](https://www.geeksforgeeks.org/sql-alter-rename/)–is used to rename an object existing in the database

**DML(Data Manipulation Language) :**The SQL commands that deals with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements.

**Examples of DML:**

* [**INSERT**](https://www.geeksforgeeks.org/sql-insert-statement/) – is used to insert data into a table.
* [**UPDATE**](https://www.geeksforgeeks.org/sql-update-statement/) – is used to update existing data within a table.
* [**DELETE**](https://www.geeksforgeeks.org/sql-delete-statement/) – is used to delete records from a database table.

1. **what should be data type of primary key?**

Because MySQL works faster with integers, the [data type](https://www.mysqltutorial.org/mysql-data-types.aspx) of the primary key column should be the integer e.g., [INT](https://www.mysqltutorial.org/mysql-int/), BIGINT. And you should ensure sure that value ranges of the integer type for the primary key are sufficient for storing all possible rows that the table may have.

A primary key column often has the [AUTO\_INCREMENT](https://www.mysqltutorial.org/mysql-sequence/) attribute that automatically generates a sequential integer whenever you [insert a new row](https://www.mysqltutorial.org/mysql-insert-statement.aspx) into the table.

When you define a primary key for a table, MySQL automatically [creates an index](https://www.mysqltutorial.org/mysql-index/mysql-create-index/) called PRIMARY.

1. **what are getters and setters**

## Getter and setter method

While defining a POJO/Bean object (or, encapsulating variables of a class) we generally,

* Declare all the variables of the as private.
* Provide public methods to modify and view their values (since you cannot access them directly).

The method that is used to set/modify the value of a private instance variable of a class is known as a setter method and, the method that is used to retrieve the value of a private instance variable is known as a getter method.

**Q.28 Types of statements and types of result set in jdbc and explain with example**

There are three types of statements in JDBC namely, Statement, Prepared Statement, Callable statement. The Statement interface represents the static SQL statement. It helps you to create a general purpose SQL statements using Java.

**Creating a statement**

You can create an object of this interface using the **createStatement()** method of the **Connection** interface.

Create a statement by invoking the **createStatement()** method as shown below.

Statement stmt = null;

try {

stmt = conn.createStatement( );

. . .

}

catch (SQLException e) {

. . .

}

finally {

. . .

}

**Executing the Statement object**

Once you have created the statement object you can execute it using one of the execute methods namely, execute(), executeUpdate() and, executeQuery().

* **execute():** This method is used to execute SQL DDL statements, it returns a boolean value specifying whether the ResultSet object can be retrieved.
* **executeUpdate():** This method is used to execute statements such as insert, update, delete. It returns an integer value representing the number of rows affected.
* **executeQuery():** This method is used to execute statements that returns tabular data (example SELECT statement). It returns an object of the class ResultSet.

## Prepared Statement

The **PreparedStatement** interface extends the Statement interface. It represents a precompiled SQL statement which can be executed multiple times. This accepts parameterized SQL quires and you can pass 0 or more parameters to this query.

Initially, this statement uses place holders “**?**” instead of parameters, later on, you can pass arguments to these dynamically using the **setXXX()** methods of the **PreparedStatement** interface.

You can create an object of the **PreparedStatement** (interface) using the **prepareStatement()** method of the Connection interface. This method accepts a query (parameterized) and returns a PreparedStatement object.

String query="INSERT INTO Employee(Name, Salary, Location)VALUES(?, ?, ?)";

Statement pstmt = con.prepareStatement(query);

## Executing the Prepared Statement

Once you have created the **PreparedStatement** object you can execute it using one of the **execute()** methods of the **PreparedStatement** interface namely, execute(), executeUpdate() and, executeQuery().

* **execute():** This method executes normal static SQL statements in the current prepared statement object and returns a boolean value.
* **executeQuery():** This method executes the current prepared statement and returns a **ResultSet** object.
* **executeUpdate():** This method executes SQL DML statements such as insert update or delete in the current Prepared statement. It returns an integer value representing the number of rows affected.

## CallableStatement

The **CallableStatement** interface provides methods to execute stored procedures. Since the JDBC API provides a stored procedure SQL escape syntax, you can call stored procedures of all RDBMS in a single standard way.

## Creating a CallableStatement

You can create an object of the **CallableStatement** (interface) using the **prepareCall()** method of the **Connection** interface.

This method accepts a string variable representing a query to call the stored procedure and returns a **CallableStatement** object.

CallableStatement cstmt = con.prepareCall("{call myProcedure(?, ?, ?)}");

Once you have created the CallableStatement object you can execute it using one of the **execute()** method.

**TYPE\_FORWARD\_ONLY:**  
  
The result set cannot be scrolled; its cursor moves forward only, from before the first row to after the last row. The rows contained in the result set depend on how the underlying database generates the results. That is, it contains the rows that satisfy the query at either the time the query is executed or as the rows are retrieved.  
  
**TYPE\_SCROLL\_INSENSITIVE:**  
  
The result can be scrolled; its cursor can move both forward and backward relative to the current position, and it can move to an absolute position. The result set is insensitive to changes made to the underlying data source while it is open. It contains the rows that satisfy the query at either the time the query is executed or as the rows are retrieved.  
  
**TYPE\_SCROLL\_SENSITIVE:**  
  
The result can be scrolled; its cursor can move both forward and backward relative to the current position, and it can move to an absolute position. The result set reflects changes made to the underlying data source while the result set remains open.  
  
The default ResultSet type is TYPE\_FORWARD\_ONLY.

**Q29.difference in class and objects**

Classes and objects from the essential part of Object-oriented programming, where a class can be considered as a construct that encapsulates a group of variables and methods; whereas, an object acts as member or instance of that class.

A class doesn't take any memory spaces when a programmer creates one.

An object takes memory when a programmer creates one.

The class has to be declared only once.

Objects can be declared several times depending on the requirement.

The class has to be declared only once. Objects can be declared several times depending on the requirement.

**Q30. Indexing (refer Q1)**

**Q31. Constraint on index**

## **DROP an INDEX Constraint**

ALTER TABLE CUSTOMERS

DROP INDEX idx\_age;

## **Q32. What is multithreading**

## <https://www.geeksforgeeks.org/multithreading-in-java/#:~:text=Multithreading%20is%20a%20Java%20feature,weight%20processes%20within%20a%20process>.

## **Q33. Exception Handling**

## <https://beginnersbook.com/2013/04/java-exception-handling/#:~:text=Exception%20handling%20ensures%20that%20the,the%20program%20will%20terminate%20abruptly>.

## **Q. 34 and 35**

## https://2.bp.blogspot.com/-X6SuJoObHkE/V3Z2h03YqAI/AAAAAAAAe54/D77IdpgiYQsJYuXUJe9OcjnKMYy066BMwCLcB/s1600/99.jpg

**Q36 Types of variables**

**Local Variables:** These types of Variables are defined within a block, a method, or a constructor of the program.

**Instance Variable:** They are variables that can be declared in a class, outside a block, a method, or a constructor. They are non-static.

These variables are created when an **object of a class is created** and destroyed when the **object is destroyed**.

Instance Variables have default values, **0** for numbers, **false** for Boolean, and **null** for object references.

**Static Variables:** They are similar in nature to Instance Variables. The major difference is that they are declared using the static keyword and only a single copy of a static variable per class is allowed.

* They are also known as Class Variables.
* Only a single copy of a static variable per class is allowed, irrespective of the number of objects created.
* These variables are created at the start of the program and are automatically destroyed when the execution of the program is completed.
* The default values of the static variables are the same as the instance variables.

**Q37. Steps in JDBC**

The fundamental steps involved in the process of connecting to a database and executing a query consist of the following:

* Import JDBC packages.
* Load and register the JDBC driver.
* Open a connection to the database.
* Create a statement object to perform a query.
* Execute the statement object and return a query resultset.
* Process the resultset.
* Close the resultset and statement objects.
* Close the connection.

For more info follow below link

<https://www.informit.com/articles/article.aspx?p=26251&seqNum=3>

**Q38. Blockings in jdbc**

Didn’t get proper answer check by yourself

**Q39. Case manipulation in sql**

There are three case manipulation functions in SQL, namely:

* LOWER: This function returns the string in lowercase. It takes a string as an argument and returns it by converting it into lower case. Syntax:

LOWER(‘string’)

* UPPER: This function returns the string in uppercase. It takes a string as an argument and returns it by converting it into uppercase. Syntax:

UPPER(‘string’)

* INITCAP: This function returns the string with the first letter in uppercase and rest of the letters in lowercase. Syntax:

INITCAP(‘string’)

**Q 40 loops in sql**

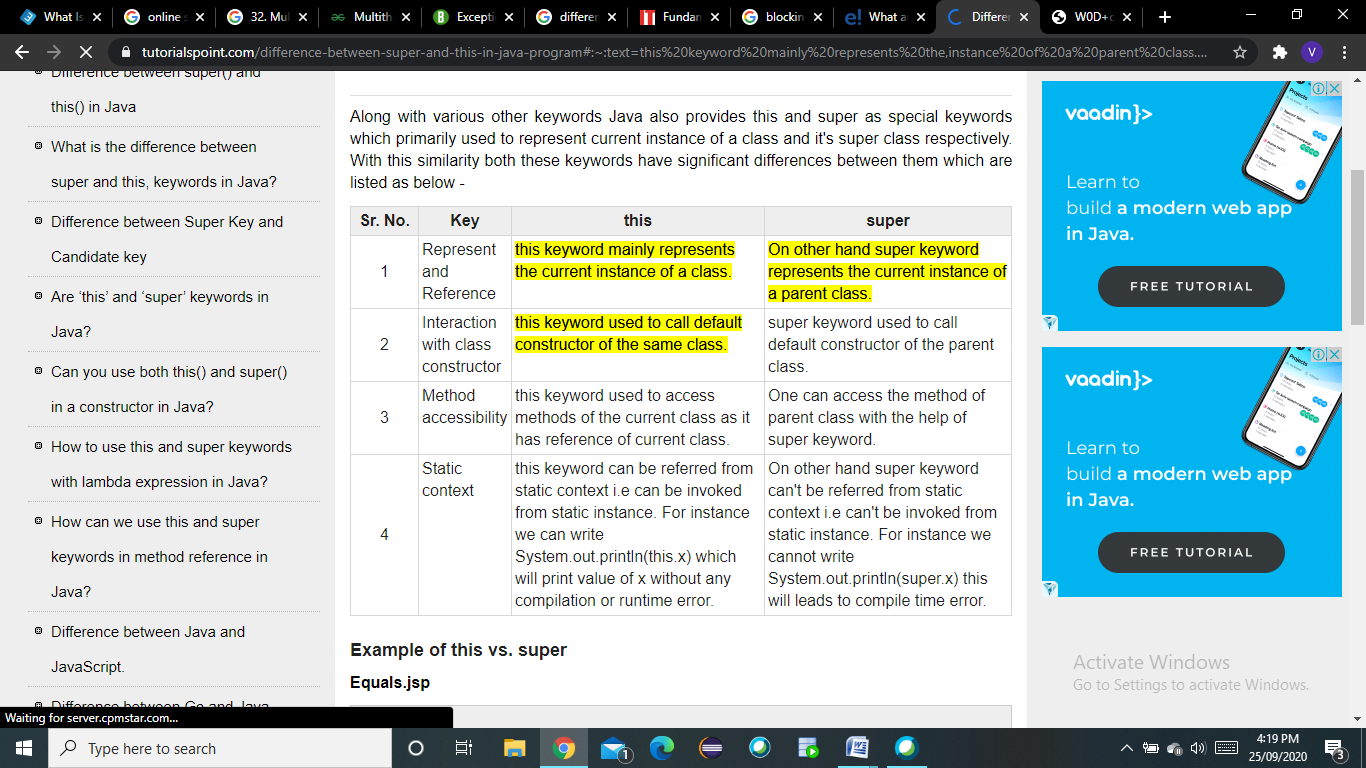
Sql server is involved here in java we just know oracle so check it on your own It might be from PL/SQL

**Q 41. error vs exception(refer 33)**

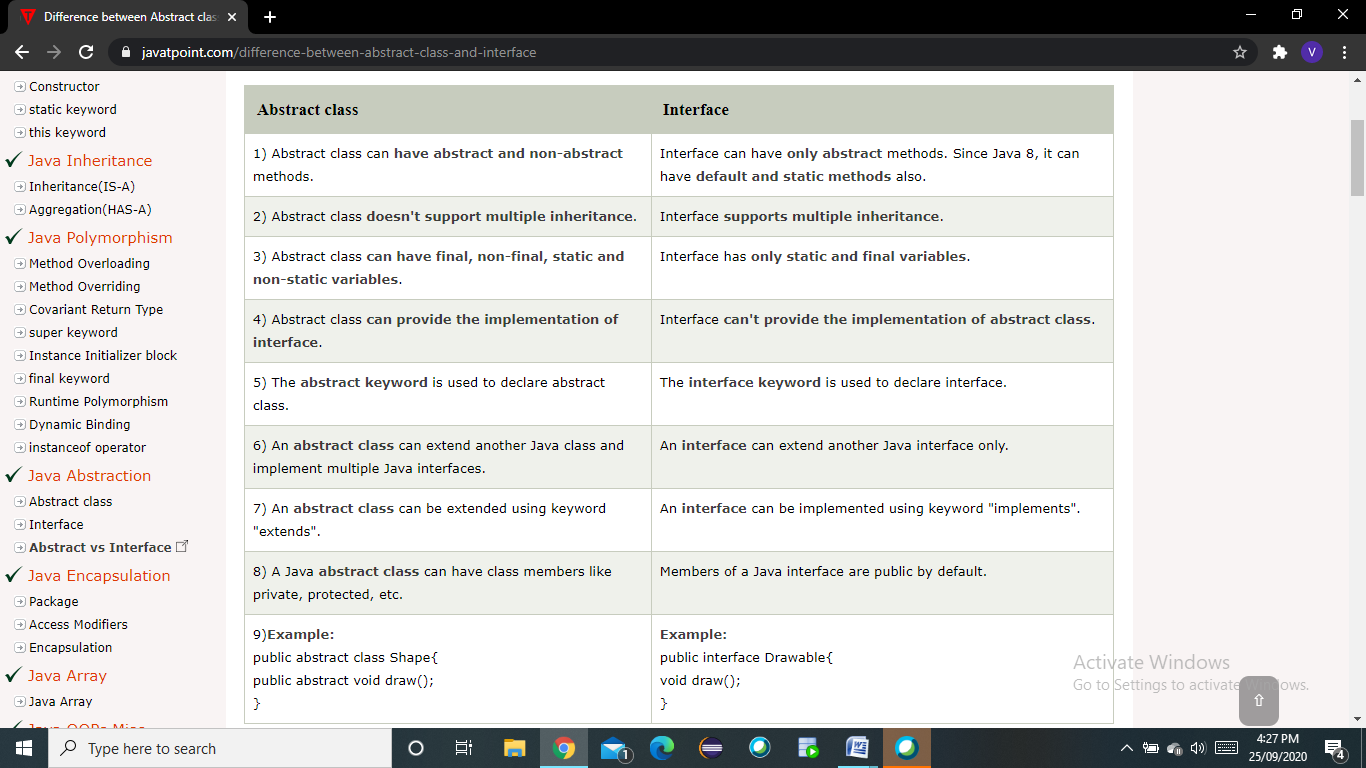
**Q 42. diff b/w throw throws and throwable**

<https://www.edureka.co/blog/throw-throws-throwable/>

**Q 43. diff between this and super**



**Q 44. Diff b/w abstract class and interface**



**Q.45 diff in static and final**

<https://stackoverflow.com/questions/13772827/difference-between-static-and-final#:~:text=static%20means%20there%20is%20only,the%20value%20of%20the%20variable>.

**Q.46 static and dynamic binding**

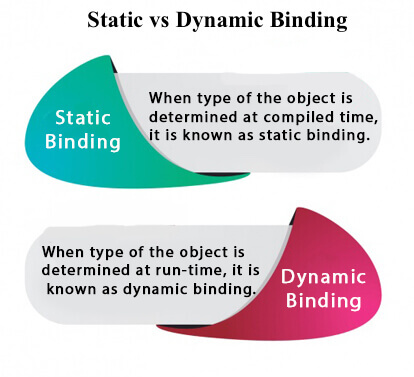
# **Static Binding and Dynamic Binding**



Connecting a method call to the method body is known as binding.

There are two types of binding

1. Static Binding (also known as Early Binding).
2. Dynamic Binding (also known as Late Binding).



### Understanding Type

Let's understand the type of instance.

#### 1) variables have a type

Each variable has a type, it may be primitive and non-primitive.

1. **int** data=30;

Here data variable is a type of int.

#### 2) References have a type

1. **class** Dog{
2. **public** **static** **void** main(String args[]){
3. Dog d1;//Here d1 is a type of Dog
4. }
5. }

#### 3) Objects have a type

|  |
| --- |
| An object is an instance of particular java class,but it is also an instance of its superclass. |

1. **class** Animal{}
3. **class** Dog **extends** Animal{
4. **public** **static** **void** main(String args[]){
5. Dog d1=**new** Dog();
6. }
7. }

|  |
| --- |
| Here d1 is an instance of Dog class, but it is also an instance of Animal. |

### static binding

When type of the object is determined at compiled time(by the compiler), it is known as static binding.

If there is any private, final or static method in a class, there is static binding.

### Example of static binding

1. **class** Dog{
2. **private** **void** eat(){System.out.println("dog is eating...");}
4. **public** **static** **void** main(String args[]){
5. Dog d1=**new** Dog();
6. d1.eat();
7. }
8. }

### Dynamic binding

When type of the object is determined at run-time, it is known as dynamic binding.

### Example of dynamic binding

1. **class** Animal{
2. **void** eat(){System.out.println("animal is eating...");}
3. }
5. **class** Dog **extends** Animal{
6. **void** eat(){System.out.println("dog is eating...");}
8. **public** **static** **void** main(String args[]){
9. Animal a=**new** Dog();
10. a.eat();
11. }
12. }

Output:dog is eating...

|  |
| --- |
| In the above example object type cannot be determined by the compiler, because the instance of Dog is also an instance of Animal.So compiler animal so compiler doesn't know its type, only its base type.  **Q47. diff in collection and collections**  The differences between the Collection and Collections are given below.  The Collection is an interface whereas Collections is a class.  The Collection interface provides the standard functionality of data structure to List,  Set, and Queue. However, Collections class is to sort and synchronize the collection elements.  The Collection interface provides the methods that can be used for data structure whereas  Collections class provides the static methods which can be used for various operation on a collection.  https://3.bp.blogspot.com/-KPg0n_OCB6Q/VZvNW6yYALI/AAAAAAAAHYQ/muLFv4zk6BM/s1600/Java%2B-%2BCollection%2BFramework%2B-%2BCollection%2BVs%2BCollections.jpg  **Q48. access modifiers scope**  https://media.geeksforgeeks.org/wp-content/cdn-uploads/Access-Modifiers-in-Java.png |

**Q 49.Reflection API**

<https://www.geeksforgeeks.org/reflection-in-java/#:~:text=Reflection%20is%20an%20API%20which,reflect%20package.&text=Through%20reflection%20we%20can%20invoke,access%20specifier%20used%20with%20them>.

**Q50. JVM JDK JIT JRE**





* **Java Virtual Machine (JVM)** is an abstract computing machine.
* **Java Runtime Environment (JRE)** is an implementation of the JVM.
* **Java Development Kit (JDK)**contains JRE along with various development tools like Java libraries, Java source compilers, Java debuggers, bundling and deployment tools.
* **Just In Time compiler (JIT)** is runs after the program has started executing, on the fly. It has access to runtime information and makes optimizations of the code for better performance.

**Q51. Diff b/w String string buffer and string builder**

Refer q7 (Sme practice)

**Q.52 Polymorphism**

[**https://www.javatpoint.com/runtime-polymorphism-in-java**](https://www.javatpoint.com/runtime-polymorphism-in-java)

[**https://www.geeksforgeeks.org/polymorphism-in-java/**](https://www.geeksforgeeks.org/polymorphism-in-java/)

**Q53. Why strings are immutable**

Because java uses the concept of string literal.Suppose there are 5 reference variables,all referes to one object "sachin".If one reference variable changes the value of the object, it will be affected to all the reference variables. That is why string objects are immutable in java.

**Q54. serialization and de serialization**

<https://www.geeksforgeeks.org/serialization-in-java/#:~:text=Serialization%20is%20a%20mechanism%20of,actual%20Java%20object%20in%20memory>.

**Q55. rowtype use**

The %ROWTYPE attribute provides a record type that represents a row in a database table. The record can store an entire row of data selected from the table or fetched from a cursor or cursor variable. Fields in a record and corresponding columns in a row have the same names and datatypes.You can use the %ROWTYPE attribute in variable declarations as a datatype specifier. Variables declared using %ROWTYPE are treated like those declared using a datatype name.

Q 56. what is thread

Thread is the smallest unit of processing that can performed in an OS

Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU. Each part of such program is called a thread. So, threads are light-weight processes within a process.  
  
Threads can be created by using two mechanisms :  
1. Extending the Thread class  
2. Implementing the Runnable Interface

Q.57 how do we create user defined exception

In java we can create our own exception class and throw that exception using throw keyword. These exceptions are known as **user-defined** or **custom** exceptions.

1. User-defined exception must extend Exception class.  
2. The exception is thrown using throw keyword.

Q.58 when we use interface and when we use class before java 8

Interfaces prior to Java 8 used to contain only abstract methods and constants, and any class that implemented an Interface must implement all its methods or be declared abstract. Well, this was before Java 8. With Java 8, that was upgraded.

Interfaces can, post-Java 7, contain methods with implementations. Such methods that have an implementation right from the interface were called default methods. These allow for a default implementation of a method by all classes that implement an interface and provides the opportunity for the implementing classes to override the method, if they so wish.

With this introduction, the major concern on the mind of every Java developer is, "How did this concept manage multiple inheritances from different interfaces or if a class and an interface have the same method?"

Well, such fear was anticipated, so the following rules applied when more than one interface had the same method and a class implements the two of them.

1. A class implementation of a method takes precedence over a default method. So if the class already has the same method as an Interface, then the default method from the implemented Interface does not take effect.
2. However, if two interfaces implement the same default method, then there is a conflict.
3. In cases where one interface inherits another interface and both of them implement a default method, the default method of the child interface would be used by an implementing class.
4. Also, an explicit call to an interface default method can be made from inside an implementing class using super. For example Interface.super.defaultMethod()

Also, Java 8 introduced static methods in Interfaces. So interfaces could create static methods and reference these static methods from anywhere simply by calling the Interface name followed by the method name. For example: InterfaceName.staticMethodName()

Q.59 what is entity in dbms

An **entity** can be a real-world object, either animate or inanimate, that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as **entities**. ... An **entity** set is a collection of similar types of **entities**.

Q.60 drawback of array

<https://www.faceprep.in/data-structures/advantages-and-disadvantages-of-arrays/>

advantages are given as well

Q.61 what will u prefer to implement runnable interface or extend thread class

If your class is *extending the Thread class* then it becomes a single thread which inherits the properties *Thread class*, so it'll be heavy. (When *extending Thread class* each of the threads creates unique object and associate with it, but when *implementing Runnable*, it shares the same object to multiple Threads).

If your class is *Implementing the Runnable interface* then you only override the run() .So this instance creates a separate Thread and every individual Thread runs separately but not as a single heavy Thread in your program. Another thing, Since Java does not support *multiple inheritance*, if you *implement the Runnable* you'll avoid problems of multiple extending, so if you implement *Runnable interface* you can extend any class that you are required other than *Thread* class.

Q.62 what is dynamic dispatch and how to get dynamic dispatch

Dynamic dispatch is basically run time polymorphism

<https://www.geeksforgeeks.org/dynamic-method-dispatch-runtime-polymorphism-java/>

explanation with code given

Q.63 sql query to get second and third highest salary

**select** \* from( **select** ename, sal, dense\_rank() over(order by sal desc)r from Employee) where r=&n; To **find** to **the** 2nd highest sal set n = 2 To **find 3rd** highest sal set n = **3** and so on

Q.64 diff between java 6 vs java 7

<https://www.freelancinggig.com/blog/2018/08/28/whats-the-differences-between-java-6-java-7-java-8-and-java-9/#:~:text=The%20difference%20between%20Java%206%20and%20Java%207%20are%20in,memory%20gets%20free%20as%20well>.

Java 6 to java 9 is given

Q. 65 what is the use of interfaces before java 8

--same as 4

Q.66 How iterate the elements of array list

<https://crunchify.com/how-to-iterate-through-java-list-4-way-to-iterate-through-loop/>

methods given

Q.67 Diff between comparable and comparator

<https://www.javatpoint.com/difference-between-comparable-and-comparator#:~:text=1)%20Comparable%20provides%20a%20single,%2C%20name%2C%20and%20price%20etc>.

diff table given

Q.68 Aggregate functions in sql

**COUNT** – counts the number of elements in the group defined

**SUM** – calculates the sum of the given attribute/expression in the group defined

**AVG** – calculates the average value of the given attribute/expression in the group defined

**MIN** – finds the minimum in the group defined

**MAX** – finds the maximum in the group defined

Q.69 Diff between delete truncate and drop

**DELETE:**

* Removes rows from a table. Delete rows one at a time & records an entry in the transaction log for each deleted row.
* Deleted data can be rollback.
* DML command
* When the DELETE statement is executed using a row lock, each row in the table is locked for deletion.

**TRUNCATE:**

* TRUNCATE removes **all rows** from a table.
* TRUNCATE TABLE always locks the table and page but not each row.
* If we truncate a table, then truncate table statement cannot be rolled back in some of the database.
* Truncate table statement is a Data Definition Language.

**DROP:**

The DROP in SQL command removes a table from the database

Q.70 Query to find number of distinct records in a table

**SELECT COUNT**(**DISTINCT** column) FROM **table**

Q.71 What are functional interfaces

A functional interface is an interface that contains only one abstract method. They can have only one functionality to exhibit. From Java 8 onwards, [lambda expressions](https://www.geeksforgeeks.org/lambda-expressions-java-8/) can be used to represent the instance of a functional interface. A functional interface can have any number of default methods. ***Runnable***, ***ActionListener***,***Comparable*** are some of the examples of functional interfaces.  
Before Java 8, we had to create anonymous inner class objects or implement these interfaces

Q.72 Hashset Vs treeset

[HashSet](https://beginnersbook.com/2013/12/hashset-class-in-java-with-example/" \t "_blank" \o "HashSet Class in Java with example) gives better performance (faster) than [TreeSet](https://beginnersbook.com/2013/12/treeset-class-in-java-with-example/" \o "TreeSet Class in Java with example" \t "_blank) for the operations like add, remove, contains, size etc. HashSet offers constant time cost while TreeSet offers log(n) time cost for such operations.

HashSet does not maintain any order of elements while TreeSet elements are sorted in ascending order by default.

Q.73 Diff. b/w string pool and heap memory

**Heap Memory**

* The heap memory is a run time data area from which the memory for all java class instances and arrays is allocated.
* The heap is created when the JVM starts up and may increase or decrease in size while the application runs.
* The size of the heap can be specified using –Xms VM option. The heap can be of fixed size or variable size depending on the garbage collection strategy. Maximum heap size can be set using –Xmx option.
* By default, the maximum heap size is set to 64 MB.

**String Constant Pool**

* String uses a special memory location to reuse of String objects called String Constant Pool.
* String objects created without the use of new keyword are stored in the String Constant Pool part of the heap.
* One of the important characteristics of String constant pool is that it does not create the same String object if there is already String constant in the pool.

Q.74 Difference between JDK,JRE and JVM.

<https://www.guru99.com/difference-between-jdk-jre-jvm.html#:~:text=JDK%20is%20a%20software%20development%20kit%20whereas%20JRE%20is%20a,JVM%20is%20Java%20Virtual%20Machine>.

Everything is given about JDK, JRE & JVM

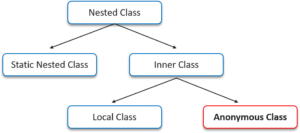
Q.75 Diff. between compile time and runtime polymorphism.

**Compile Time Polymorphism:** Whenever an object is bound with their functionality at the compile-time, this is known as the compile-time polymorphism. At compile-time, java knows which method to call by checking the method signatures. So this is called compile-time polymorphism or static or early binding. Compile-time polymorphism is achieved through [method overloading](https://www.geeksforgeeks.org/overloading-in-java/). Method Overloading says you can have more than one function with the same name in one class having a different prototype. Function overloading is one of the ways to achieve polymorphism but it depends on technology that which type of polymorphism we adopt. In java, we achieve function overloading at run time. The following is an example where compile-time polymorphism can be observed.

**Run-Time Polymorphism:** Whenever an object is bound with the functionality at run time, this is known as runtime polymorphism. The runtime polymorphism can be achieved by [method overriding](https://www.geeksforgeeks.org/overriding-in-java/). [Java virtual machine](https://www.geeksforgeeks.org/jvm-works-jvm-architecture/) determines the proper method to call at the runtime, not at the compile time. It is also called dynamic or late binding. Method overriding says child class has the same method as declared in the parent class. It means if child class provides the specific implementation of the method that has been provided by one of its parent class then it is known as method overriding. The following is an example where runtime polymorphism can be observed.

Q.76 What is Anonymous class?

In simple words, the anonymous inner class is a class without names and only one object is created.



Anonymous class is useful when we have to create an instance of the object with overloading methods of a class or interface without creating a subclass of class.

Anonymous can be created in two ways:

* Class(can also be Abstract)
* Interface

In the anonymous class we can declare the following:

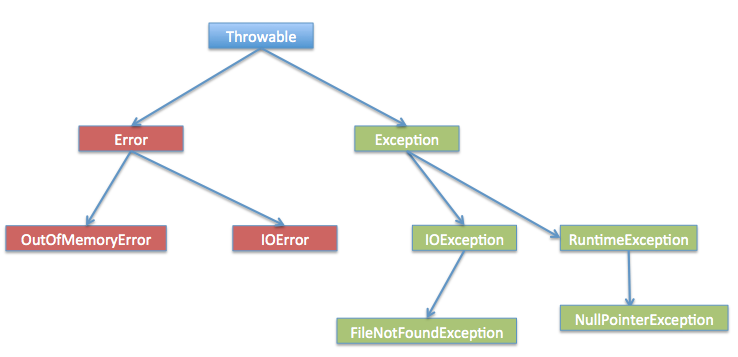
* Fields
* Extra methods
* Instance Initializers
* Local classes

**Syntax of Anonymous Class in Java**

The syntax of an anonymous class is just like constructor except that there is a class definition in the block as shown in the snippet below:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | // AnonymousClass = interface,abstract/concrete class.  AnonymousClass t = newAnonymousClass()  {    // methods and fields    publicvoidsomeMethod()    {       //code goes here     }  } |

Q.77 Explain the exception hierarchy.



Q.78 What is the default size of Map?

The capacity of an *HashMap* is the number of buckets in the hash table. The initial capacity is the capacity of an *HashMap* at the time of its creation. The default initial capacity of the *HashMap* is 24 i.e 16. The capacity of the *HashMap* is doubled each time it reaches the threshold. i.e the capacity is increased to 25=32, 26=64, 27=128….. when the threshold is reached.

Q.79 Diff. between Comparable and Comparator. –same as 13

Q.80 Diff. b/w ArrayList and List.

|  |  |
| --- | --- |
| **LIST** | **ARRAYLIST** |
| List is an Interface. | ArrayList is a Class. |
| List interface extends the [Collection framework](https://www.geeksforgeeks.org/collections-in-java-2/). | ArrayList extends [AbstractList class](https://www.geeksforgeeks.org/abstractlist-in-java-with-examples/) and implements [List interface](https://www.geeksforgeeks.org/list-interface-java-examples/). |
| List cannot be instantiated. | ArrayList can be instantiated. |
| List interface is used to create a list of elements(objects) which are associated with their index numbers. | ArrayList class is used to create a dynamic array that contains objects. |
| List interface creates a collection of elements that are stored in a sequence and they are identified and accessed using the index. | ArrayList creates an array of objects where the array can grow dynamically. |

Q.81 Diff between vector and arraylist.

Vector and ArrayList both uses Array internally as data structure. They are dynamically resizable. Difference is in the way they are internally resized. By default, Vector doubles the size of its array when its size is increased. But, ArrayList increases by half of its size when its size is increased.

Therefore as per Java API the only main difference is, Vector’s methods are synchronized and ArrayList’s methods are not synchronized.

**Q.82 Vector or ArrayList? Which is better to use in java?**

In general, executing a ‘synchronized’ method results in costlier performance than a unsynchronized method. Keeping the difference in mind, using Vector will incur a performance hit than the ArrayList. But, when there is a certain need for thread-safe operation Vector needs to be used.

Q.83 Diff. b/w List and Set.

 List is an ordered collection it maintains the insertion order, which means upon displaying the list content it will display the elements in the same order in which they got inserted into the list.

Set is an unordered collection, it doesn’t maintain any order. There are few implementations of Set which maintains the order such as LinkedHashSet (It maintains the elements in insertion order).

2) List allows duplicates while Set doesn’t allow duplicate elements. All the elements of a Set should be unique if you try to insert the duplicate element in Set it would replace the existing value.

3) List implementations: [ArrayList](https://beginnersbook.com/2013/12/java-arraylist/" \o "ArrayList in java with example programs – Collections Framework" \t "_blank), [LinkedList](https://beginnersbook.com/2013/12/linkedlist-in-java-with-example/" \t "_blank" \o "LinkedList in Java with Example) etc.

Set implementations: [HashSet](https://beginnersbook.com/2013/12/hashset-class-in-java-with-example/" \t "_blank" \o "HashSet Class in Java with example), [LinkedHashSet](https://beginnersbook.com/2013/12/linkedhashset-class-in-java-with-example/" \o "LinkedHashSet Class in Java with Example" \t "_blank), [TreeSet](https://beginnersbook.com/2013/12/treeset-class-in-java-with-example/" \o "TreeSet Class in Java with example" \t "_blank) etc.

4) List allows any number of null values. Set can have only a single null value at most.

5) [ListIterator](https://beginnersbook.com/2014/06/listiterator-in-java-with-examples/" \o "ListIterator in Java with examples" \t "_blank) can be used to traverse a List in both the directions(forward and backward) However it can not be used to traverse a Set. We can use [Iterator](https://beginnersbook.com/2014/06/java-iterator-with-examples/" \t "_blank" \o "Java Iterator with examples) (It works with List too) to traverse a Set.

6) List interface has one legacy class called [Vector](https://beginnersbook.com/2013/12/vector-in-java/)whereas Set interface does not have any legacy class.

Q.84 What is unboxing?

**Autoboxing:**Converting a primitive value into an object of the corresponding [wrapper class](https://www.geeksforgeeks.org/wrapper-classes-java/) is called autoboxing. For example, converting int to [Integer class](https://www.geeksforgeeks.org/wrapper-classes-java/). 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:** Converting an object of a wrapper type to its corresponding primitive value is called unboxing. For example conversion of [Integer](https://www.geeksforgeeks.org/wrapper-classes-java/) to int. The Java compiler applies unboxing 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**.

Q.85 Is HashMap synchronized?

No.

Q.86 Internal Working of HashSet and HashMap.

<https://javaconceptoftheday.com/how-hashset-works-internally-in-java/#:~:text=HashSet%20uses%20HashMap%20internally%20to,with%20it%20is%20also%20created.&text=The%20elements%20you%20add%20into,keys%20will%20be%20a%20constant>.

Q.87 Why main method is static

Java **main()**method is always static, so that compiler can call it without the creation of an object or before the creation of an object of the class.

* In any Java program, the **main()** method is the starting point from where compiler starts program execution. So, the compiler needs to call the main() method.
* If the**main()** is allowed to be non-static, then while calling the **main()** method JVM has to instantiate its class.
* While instantiating it has to call the constructor of that class, There will be ambiguity if the constructor of that class takes an argument.
* Static method of a class can be called by using the class name only without creating an object of a class.
* The **main()**method in Java must be declared **public**, **static**and **void**. If any of these are missing, the Java program will compile but a runtime error will be thrown.

Q.88 How to create an immutable class?

Q.89 Diff. b/s == and equals() method.

Q.90 Read about cursors,functions, triggers ,sequqnces, procedures

Q.91 diff in function and procedures

**SME Practice(**

1. How JVM works?

JVM(Java Virtual Machine) acts as a run-time engine to run Java applications. JVM is the one that actually calls the **main** method present in a java code. JVM is a part of JRE(Java Runtime Environment).

Java applications are called WORA (Write Once Run Anywhere). This means a programmer can develop Java code on one system and can expect it to run on any other Java enabled system without any adjustment. This is all possible because of JVM. First, Java code is complied into bytecode. This bytecode gets interpreted on different machinesBetween host system and Java source, Bytecode is an intermediary language.JVM is responsible for allocating memory space.

2. diff in & and &&

& is bitwise. && is logical.

& evaluates both sides of the operation.  
&& evaluates the left side of the operation, if it's true, it continues and evaluates the right side.

3. Hash code Contract:

contract between equals() and hashCode() is:  
1) If two objects are equal, then they must have the same hash code.  
2) If two objects have the same hash code, they may or may not be equal.

If two objects are equal according to the equals(Object) method, then calling the hashCode method on each of the two objects must produce the same integer result.

4. What is Serialization and deserialization?

[Seriazlization](https://www.geeksforgeeks.org/serialization-in-java/#:~:text=Serialization%20is%20a%20mechanism%20of,actual%20Java%20object%20in%20memory.)

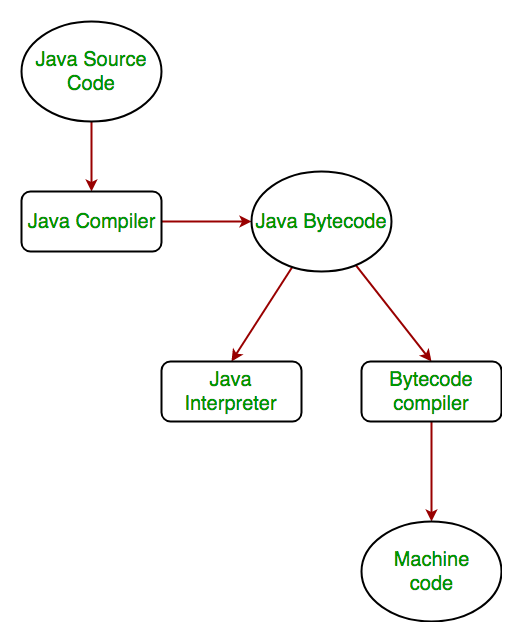
5. What will happen if we remove equals and hashcode methods from Object class?

You must override hashCode() in every class that overrides equals(). Failure to do so will result in a violation of the general contract for Object.hashCode(), which will prevent your class from functioning properly in conjunction with all hash-based collections, including HashMap, HashSet, and Hashtable.

6. What will happen if we remove equals and hashcode methods from Object class?

The meaning of platform-independent is that the java compiled code(byte code) can run on all operating systems.  
A program is written in a language that is a human-readable language.

* The result of the JAVA compiler is the **.class file or the bytecode** and not the machine native code (unlike C compiler).
* The bytecode generated is a non-executable code and needs an interpreter to execute on a machine. This interpreter is the JVM and thus the Bytecode is executed by the JVM.



7. Diff b/w string string buffer and string builder

String

String is immutable ( once created can not be changed )object . The object created as a String is stored in the Constant String Pool. Every immutable object in Java is thread safe ,that implies String is also thread safe . String can not be used by two threads simultaneously. String once assigned can not be changed.

String Buffer

String Buffer is mutable means one can change the value of the object . The object created through String Buffer is stored in the heap. String Buffer has the same methods as the String Builder , but each method in String Buffer is synchronized that is String Buffer is thread safe . Due to this it does not allow two threads to simultaneously access the same method . Each method can be accessed by one thread at a time .

But being thread safe has disadvantages too as the performance of the String Buffer hits due to thread safe property . Thus String Builder is faster than the String Buffer when calling the same methods of each class

String Builder:

String Builder is same as the String Buffer , that is it stores the object in heap and it can also be modified . The main difference between the String Buffer and String Builder is that String Builder is also not thread safe. String Builder is fast as it is not thread safe .

Q. 8 . Can a class be static or final or abstract?

Class can be final or abstract

For static :

The answer to this question is both **Yes**and **No**, depending on whether you are talking about a top-level class or a nested class in Java. You cannot make a top-level class static in Java, the compiler will not allow it, but you can make a nested class static in Java.

Q.9 What is Volatile?

The Java volatile keyword is used to mark a Java variable as "being stored in main memory". More precisely that means, that every read of a volatile variable will be read from the computer's main memory, and not from the CPU cache, and that every write to a volatile variable will be written to main memory, and not just to the CPU cache.

Volatile in java is different For Java, “volatile” tells the compiler that the value of a variable must never be cached as its value may change outside of the scope of the program itself. Thread safe means that a method or class instance can be used by multiple threads at the same time without any problem.

Q.10 what is Tostring method?

If you want to represent any object as a string, **toString() method** comes into existence.

The toString() method returns the string representation of the object.

If you print any object, java compiler internally invokes the toString() method on the object. So overriding the toString() method, returns the desired output, it can be the state of an object etc. depends on your implementation.