**Horizontal scaling vs Vertical scaling:**

Horizontal scaling means that you scale by adding more machines into your pool of resources whereas Vertical scaling means that you scale by adding more power (CPU, RAM) to an existing machine.

**SQL vs NOSQL which to choose and when:**

SQL database is a table-based structure with fixed schema for storing data. SQL databases are well-suited for applications that require complex queries, transactions, and structured data.

NOSQL databases are non-relational databases which use various data models to store and retrieve data. Can handle semi structured and unstructured data more efficiently than SQL. NoSQL databases are better for applications that require scalability and flexibility in handling unstructured data.

SQL databases are better suited for traditional business applications such as finance, banking, and accounting, while NoSQL databases are better suited for applications such as social media, e-commerce, and gaming, which require handling large volumes of data and high scalability.

**Lambda Expressions:** Lambda expressions can be used to facilitate functional programming and simplify development a lot.

**Functional Programming:** Defining the body of the method at the time of developing the application. Lambda expressions provide functional programming.

**Functional interface:** Java interface that has only one abstract method, known as the functional method.

**Stream:** It is the new abstract layer which represents a sequence of objects. We can use Stream API to process data in declarative way, like SQL statements.

**monolithic vs microservices:** small deployable units in microservices whereas bundle of modules in monolithic.

**Garbage collector – finalize:**

The garbage collector in Java is responsible for freeing up memory used by objects that are no longer referenced by the program.  
The "finalize" method is a method that can be defined in a class, and it is automatically called by the garbage collector before the object is garbage collected.

**public static void main [string[] args]:**

Public – accessible for all classes, Static – method belongs only to class not for instances, Void – return type, main – main method entry point searched by jvm , string[] args- array of strings that can be passed as command line arguments.

**static vs final:**

Static means it belongs only to class not for instances This means that a static variable or method can be accessed without creating an object of the class. FINAL is used to define constants, meaning variables that cannot be changed after they are initialized.

**singleton vs prototype:**

Singleton pattern is used to ensure that only one instance of a class is created and that the same instance is used throughout the application. The Singleton pattern is implemented by creating a class with a private constructor and a static method that returns the same instance of the class every time it is called.

Prototype allows for the creation of multiple instances based on a prototype object. Uses clone() method to copy the object.

**String vs StringBuilder vs StringBuffer:**

String is an immutable class, which means that once a String object is created, it cannot be modified. Less efficient when more modifications are required.

StringBuilder is a mutable class that can be used to modify strings. mutable, more efficient than StringBuffer in single-threaded applications, not thread-safe.

StringBuffer is a mutable class that can be used to modify strings. mutable, thread-safe, less efficient than StringBuilder in single-threaded applications

**Database indexing:** a technique used to improve the performance of database queries. Indexing is the process of creating a data structure that allows for faster data retrieval**.**

**Default method:** Is a method defined in an interface that provides a default implementation for that method.

**Comparable Vs Comparator:** Comparable interface compares “this” reference with the object specified and Comparator in Java compares two different class objects provided.

**Dependency Injection Principle: is** nothing but being able to pass (inject) the dependencies when required instead of initializing the dependencies. Dependency injection makes it easier to test and maintain code by reducing the amount of coupling between classes.

**Serialization:** the process of converting an object into a stream of bytes, so that it can be easily stored in a file, database, or transmitted over a network**.** It is achieved by implementing the Serializable interface. Any class that implements the Serializable interface can be serialized and deserialized.

Serialize – FileOutputStream, ObjectOutputStream – writeObject()

Deserialize – FileInputStream, ObjectInputStream – readObject()

**Marker Interface:** a interface which does not have any methods or fields. It is used to mark the class with a property or behavior. E.g.: serializable, Cloneable.

**Abstraction:** the process of hiding the implementation details of a system and only exposing the essential features and functionality to the user. In object-oriented programming, abstraction is implemented using abstract classes and interfaces.

An **abstract class** is a class that cannot be instantiated on its own and must be subclassed to be used.

100% Abstraction: using Interfaces, abstract classes, encapsulation, (SOLID principles and design patterns)

**.equals vs ==:**

== is same object in memory.

.equals() method two objects for content equality.