**Day 9: 28-10-2025:**

**Java 8:**

1. From Java 8 onward interface can contains method with body. But we need to use default or static keyword with those methods.
2. if interface contains default as well as static method with body. You no need to override those method means no need to provide the body. If you want you can override default method but you can’t override static methods.
3. If more than interface contains same method body with different logic and the class which implements that interface need to override mandatory.
4. Functional interface: the interface which contains only one abstract method. it can contain more than one static as well as default it must be contains only one abstract method that type of interface is known as functional interface.
5. Marker interface: the interface doesn’t contains any method means no method that type of interface is known as marker interface. Example Serializable or Cloneable or Remote etc.
6. Lambda expression : Before Java 8 Java was known as object oriented and procedure language. But From Java8 onward with help of lambda concept we can say Java also functional programming language

(the language which support nested function as well as function chaining concept is known as functional programming language).

Using lambda expression is like an anonymous method.

With help of lambda we can provide shorted format of code for method. if you want to achieve the lambda expression we need interface with only one abstract method ie functional interface.

Lambda syntax

InterfaceName referenceName = (parameterList)->expression;

It return the result or expression without return keyword with method signature.

Inner classes: class within another class

1. Non static inner class or instance type property (number of copies of object)
2. Static inner class or static type property (one copy
3. Anonymous inner class
4. Local method inner class

Command functional interfaces

1. **Predicate :** this interface contains one abstract method is test() it take T parameter and return type is boolean fixed.
2. **Consumer :** it contains accept() method pass T parameter but no return type.
3. **Supplier :** it contains get() method no passing parameter but return T value.
4. **Function :** method is apply() takes T as parameter and return R value.

These all functional interfaces part of function package and function package is sub package of util package.

**Stream API:** Stream API is a Java 8 features. Using steam API we can process collection of data from array or collection classes in a functional and declarative way.

Stream API allow us to load the data from collection or array on demand and achieve business requirements.

It allow developer or programmer to perform operation like filtering, mapping, sorting and searching without modifying the origin data present in collection or array.

Stream doesn’t hold data.

In simple stream, flow of data or sequence of element from source to destination.

Declarative : focuses on what to do rather than how to do.

Function programming style : using lambda expression or method reference.

Non – storage : steam doesn’t store the data. It hold the data for temporary purpose.

Collection is storage. Means it hold the data.

Lazy evaluation : intermediate operator do the operation once you connect to terminal operator.

Stream divided into 3 parts

Source Data Stream

Array

Or -🡪 Stream 🡪 Inter mediate operator or method1,2,3,

Collection -🡪 Terminal operator

Source Intermediate operator

List, Set, Map, File filter forEach()

Database map collect()

Or Array sorted findAny()

Skip findAll()

count()

Intermediate operator return type is stream itself. These intermediate operator takes functional interface reference as parameter like Function, Consumer, Predicate and Supplier etc. which help to do specific task.

Terminal operator: return type is non stream is void, float, Boolean, long etc.

Intermediate operator

Filter

Map

Flatmap

Skip

Limit

Terminal operator

forEach : void return type

count :long

anyMatch : Boolean

allMatch : Boolean

noneMatch : Boolean

reduce : long

reduce function mainly use to combine all elements of a stream into a single result or output.

Example : sum of all numbers

Number of products or number customer etc.

collect : collect takes Collectors class reference which contains static method which help to convert to back to source like set, list etc.

Collectors utilities classes methods.

1. groupBy :