**Day 9:**

Java Features

**Java Version 1.8**

1. From Java 8 onward interface can contains method without body. But we need to **use default or static** keyword with methods.
2. If we write method with default keyword we can write method with body. The class which implements that interface not mandatory need to provide the body for default method. If we want, we can override default method. But static method can’t override.
3. Functional interface: the interface contains only one abstract method. it can more than one default or static but only one abstract method is known as functional interface.
4. To check the interface is functional interface or not we can use @FuntionalInterface annotation.
5. Lambda expression : using lambda expression in java we can achieve functional programming concept. Using lambda expression we can write method or function in anonymous function or methods style code. Using lambda we can provide the body for functional interface abstract method in anonymous style.
6. Syntax

Interfacename refereneName = (parameterList)->body;

By default it return expression without return keyword.

1. **Stream API** : Stream is flow of data. Using stream api in java we can achieve **declarative and function style** to access the data from collection or array through a pile line of stream operation. Stream is use to hold the data for temporary purpose. On demand we retrieve data from collection or array and apply some business logic on those data.

While using steam concept we need to use pre defined functional interface

**Ie Function, Consumer, Predicate and Supplier etc.**

Pre defined functional interface.

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

Function : public abstract R apply(T); it takes T as parameter and return R value.

Supplier : public abstract T get(); : no passing parameter but return type is T.

Predicate : public abstract **boolean** test(T); : takes T as parameter and return boolean value.

Consumer : public abstract **void accept(T);** :it takes T as a parameter but no return type.

Stream API:

Collection

Or --🡪Stream 🡪IO1🡪IO2🡪Ion--🡪TO

Array

First we need to convert to stream, then we can use more than one intermediate operator. Those operator return type itself is stream and then we can use terminal operator. Terminal operator return type is non stream ie void or int, float, double, String etc

**Intermediate operator**

Filter()

Map()

Sort()

Skip()

Limit()

Etc

**Terminal operator**

**forEach()**

collect()

anyMatch()

allMatch()

reduce()

count();