Java 8 Features Day 1 : 20-12-2022

Interface : Till Java 7 interface is known as 100% pure abstract class. Which contains only abstract method. From Java8 onward interface can contains method with body. But the method is must with default or static keyword.

Interface contains default method with default implementation then the class which implements that interface no need to provide the body. If we need to override we can override default method also.

Static method we can’t override. If interface contains method with body with keyword as static we can’t call that interface through object of that class. The class which implements that interface. Static method which belong to interface we have to class through interface name.

Bank interface

withdraw

deposit

default gst 12%

static home loan : 8%

HDFC implements Bank

Provided body for withdraw and deposit

HSBC implements Bank

Provided body for withdraw and deposit

SBI implements Bank

Provided body for withdraw and deposit

Functional interface.

The interface contains only one abstract method is known as functional interface. It can contains more than one default as well as static method but only one abstract method.

Marker interface The interface contains zero method or no method is known as marker interface. Example : Clonenable or Serializable.

Java 8 provided pre defined annotation @FuntionalInterface. This annotation we can use on interface level to check is that interface is functional interface or not.

Inner class :

Class within another class is known as inner class.

1. Non static inner class
2. Static inner class
3. Anonymous inner class
4. Local method class

Lambda Expression : Lambda is a Greek word. Using lambda expression Java also become functional programming language.

Function within another function or method within another method.

void display() {

void display2() {

}

}

From Java8 onward using lambda expression we can achieve functional programming concept. Lambda expression is anonymous function or method.

If we can lambda expression for those method. The method must be part of interface and that interface must be functional interface.

Day 2 : 22-12-2022

Lambda Expression

Syntax

We can apply lambda expression for only those interface which is type of functional interface (which contains only one abstract method).

InterfaceName refereceName = (parameterList)-> body of interface.

If we want to write only one line statement then curly braces not required.

If we write only one statement body of that interface method in lambda syntax return output without return keyword.

In lambda expression if we want to write multi line statement then we have to use curly braces and if that method signature return the value then at the end we need to return the value using return keyword.

Pre defined functional interfaces.

Top four functional interface and these four interfaces are of function package and it is sub package of util package.

javap java.util.function.Function

Function : public abstract R apply(T); it contains one abstract method with name as apply. We need to pass T parameter and it return R value.

T mean type and R means return.

We can pass any value and it return any value.

Supplier : it contains one abstract method with name as Get. It doesn’t take any parameter but return R value. R means any value.

Consumer : it contains one abstract method with name accept. It take T parameter and no return type.

Predicate : it contains one abstract method with name test. It take T parameter and return boolean value.

Stream API : Stream mean flow of data.

Collection Framework. Collection provide data structure. Data structure classes and interface are in memory data.

If our collection framework class like ArrayList hold huge data may be type of primitive or user defined object.

List<String> ll = new ArrayList();

We store huge names ll object.

If we want to apply some condition on those values.

We need to iterate one by one using Iterator or ListIterator or Using for loop or for each loop. In Iterator or ListIterator or for we do any changes base upon our requirement our actual value get effected.

List<Employee> listOfEmp = new ArrayList();

Iterator<Employee> li = listOfEmp.Iterator();

while(li.hasNext()) {

Employee emp = li.next();

If(emp.designation(“Manager”)) {

emp.setSalary(emp.getSalary()+5000);

}

}

Stream : Flow of data. On demand we can load the data from container and apply the logic. Stream hold data temporary. After business logic stream get destroy.

If we do any changes in Stream actual collection doesn’t get effect.

Flow of the stream

Collection Classes or Container --🡪 Stream --🡪 Intermediate Operator1-🡪 Intermediate Operator2-> Intermediate Operator3-> Terminal operator.

Intermediate operator return type is stream itself.

Terminal operator return type is non stream mean void, int, float, double but not stream.

String api intermediate and terminal operator take lambda expression as a parameter of pre defined function interface of type Function, Consumer, Predicate, Supplier.