**JAVA 8**

-java 8 mainly created for **concise code(small code).**

-How they acchive that by enabling the **functional programming** as java is oops lang.

-**Lamda Expression**

Suppose if you want to create a method which take i/p as int and giving o/p as squr. Of that no then how we code normally upto 1.7 java version.

Class test

{

Public static int squrIt(int n)

{

Return n\*n;

}

Public static void main(String[] args)

{

Sop(squrIt(5));

Sop(squrIt(4));

}

}

BUT INSTEAD OF THIS YOU CAN DO

**import** java.util.function.Function;

**public** **class** Test

{

**public** **static** **void** main(String[] args)

{

Function<Integer,Integer>f=i->i\*i;

System.***out***.println(f.apply(5));

System.***out***.println(f.apply(4));}}

Imp Update from JAVA 8

1)Lambda Expression

2)Functional Interface.

3)Default Method and Static Method

4)Predefined Functional Interface

i)Predicate

ii)Function

iii)Consumer

iv)Supplier

5)Double Colon Operator(::)

METHOD REFERECE PEURPOSE

CONSTRUCTOR reference

6)Streams API

7)Date and Time API

8)Optional class

9)Nashron javaScript engine.

1)Lambda Expression:

1930

LISP

What is lambda expression?

- **Lambda expression provides implementation of functional interface. An interface which has only one abstract method is called functional interface.**

-it is an anonymous function

Nameless function

- Lambda expression helps us to write our code in functional style. It provides a clear and concise way to implement SAM interface(Single Abstract Method) by using an expression. It is very useful in collection library in which it helps to iterate, filter and extract data.

How we can write nameless function

Without name ,without return type,without modifier

If input is only one so you don’t need bracket and data

Type handle by compiler and return also not required

If you remove bracket for one line statement

EX: We have to write program to calculate length of input string

Normal way

**public** **class** Test

{

**public** **static** **int** length(String s)

{

**return** s.length();

}

**public** **static** **void** main(String[] args)

{

System.***out***.println("Length of String is "+*length*("SHUBHAM"));

}

}

BY using lambda rules

s-> s.length();

**2)Functional interface:**

-Single abstract method that we called it as Functional interface

Ex:Runnable:run()

Comparable:compareTo()

Comparator:compare()

Callable :call();

* What is use of that functional interface:

-if you want to invoke lambda expression we use functional interface.

-Lambda Expression is only applicable for functional interfaces.

-if we don’t have FI then we can’t write LMBD Expression.

-in collection lambda expression is only used in sorting by Fi(comparator ,and comparable).

-Lambda expression is not a replacement for anonymous inner class.

-VERY Imp

-EX:

Runnable r=new Runnable()

{

………………

};

Here we are not creating the object of interface we can not that actually. We are trying to create the implementation class for the interface.

-But anonymous concept is more powerfull than lambda expression becz lambda expression are only applicable for single Sam but not for anonymous.

3) Default||virtual extension method||Defender:

-Until 1.7 version

-Every method present inside interface in always public and abstract whether you declared or not by default it

-From 1.8V(default + static method allowed)

-1.9v private methods also allowed.

-Without affecting implementation classes if you want to add new method in interface then we need to go for default method;

-default you can only use within the interface not within the classes becz it lead to error what to take

Switch case default or what.

-Default method in interface provide dummy implementation the implemented class is responsible to provide the proper implementation.

-Object class method we can’t implement using default method they are already available to class.

INTERFACE STATIC METHOD.

-To define general utility method which are never going to talk with object go for static method in interface

-Interface static method are not by default available to implementation class.

-Interface method should be called by interface name .method name

-main method is possible to declare inside interface from 1.8V on word

-why static method included in interface?

Answer: as if your method is static which is not going to related

With object then it is not recommended to go for class as class

Are heavy weight and costly use interface which can directly call by interface name.method

**4) Predefined Interface Function**

1)Predicate:

-This is also Fi which contain only one SAM

EX:

Interface Predicate

{

Public abstract Boolean test(T t);

}

-when ever some Boolean condition you want to check

Then we can go for predicate function interface having test method.

EX:

**public** **class** Predicateexp

{

**public** **static** **void** main(String[] args)

{

Predicate<Integer>p1=i->i%2==0;

System.***out***.println(p1.test(10));

System.***out***.println(p1.test(15));

}

}

2)Function:

-If return type is not Boolean it like Interger, string then we should go for function predefined functional interface

EX:

Interface Function<Input Return type>

{

Public R apply(T t)

}

EX:BASIC

**public** **class** Test {

**public** **static** **void** main(String[] args)

{

Function<Integer,Integer>f=i->i\*i;

System.out.println(f.apply(4));

}

}

3)Consumer:

-Consumer consume item but never going to return anything .Take the i/p perform certain operation and won’t return anything.

Consumer<T> void

Ex:

Interface Consume<T>

{

Public abstract void accept(T t);

}