**There is list of below features of java 1.8 as**

Functional interface

Lambda Expression

Default method

Static method

forEach Method

Optional Class

String Joiners

**Functional Interface**

An Interface that contains exactly one abstract method is known as functional interface.

Following are the functional interfaces(self-study)

predicate consumer Function

supplier

It can have any number of default, static methods but can contain only one abstract method.

Example

Package com.test;

@FunctionalInterface

public interface Test{

Void getStudentName(String name);

}

Package com.test;

Public class Main implements Test{

@Override

public void getStudentName(String name){

System.out.println(name);

}

public static void main(String[] args){

Main main=new Main();

main.getStudentName("ashok");

}

}

**Lambda Expression**

Why?

Less coding

Syntax-(argument-list)->{body}

 Argument-list:It can be empty or non-empty as well.

 Arrow-token:It is used to link arguments-list and body of expression.

 Body:It contains expressions and statements for lambda expression.

No Parameter Syntax

()->{

//Body of no parameter lambda

}

One Parameter Syntax

(p1)->{

//Body of single parameter lambda

}

Two Parameter Syntax

(p1,p2)->{

//Body of multiple parameter lambda

}

Example

package com.test;

public interface Addition{

int add(int a,int b);

}

package com.test;

public class Main{

public static void main(String[] args){

//Multiplep arguments in lambda expression

Addition addition=(a,b)->(a+b);

System.out.println(addition.add(10,20));

//Multiple parameters with data type in lambda expression Addition addition2=(inta,intb)->(a+b);

System.out.println(addition2.add(100,200));

}

}

Output

30

300

**Default method**

Java provides a facility to create default methods inside the interface. Methods which are defined inside the interface and tagged with default are known as default methods. The methods are non-abstract methods.

Example

package com.test;

public interface Example{

default void m1(){

System.out.println("this is default m1 method");

}

}

package com.test;

public class TestMain implements Example{

public static void main(String[] args){

TestMain testMain=new TestMain();

testMain.m1();

}

}

Output

this is default m1 method

**Static method**

Java provides a facility to create static methods inside the interface. Package com.demo;

public interface Example{

static void x1(){

System.out.println("this is static method");

}

}

package com.demo;

public class MainTest implements Example{

public static void main(String[] args){

Example.x1();

}

}

**forEach() method**

The Java forEach() method is a utility function to iterate over a collection such as(list, set or map) and stream. It is used to perform a given action on each the element of the collection.

package com.test;

import java.util.HashMap;

import java.util.Map;

public class MapDemo{

public static void main(String[]args){

Map<String,String>map=new HashMap<String,String>();

map.put("10","ram");

map.put("11","shyam");

map.put("12","ganesh");

//Old approach

Set<String> s=map.keySet();

for(String str : s) {

System.out.println("key>>" +str + " Value " + map.get(str)); }

//New Approach

map.forEach((k,v)->System.out.println("Key="+k+",

Value="+v));

}

}

Output

Key=11, Value=shyam

Key=12,Value=ganesh

Key=10,Value=ram

**Optional class**

Java introduced a new class Optional in jdk8. It is a public final class and used to deal with NullPointerException in Java application.

You must import java.util package to use this class. It provides methods which are used to check the presence of value for particular variable.

Why?

package com.test;

public class MapDemo{

public static void main(String[] args){

String[]str=new String[10];

String lowercaseString=str[5].toLowerCase();

System.out.print(lowercaseString);

}

}

Exception in thread "main" java.lang.NullPointerException

at com.test.MapDemo.main(MapDemo.java:8)

Here we are getting exception, to avoid this type of exception, we should go for optional class

Package com.test;

import java.util.Optional;

public class MapDemo{

public static void main(String[]args){

String[]str=new String[10]; Optional<String>checkNull=Optional.ofNullable(str[5]);

if(checkNull.isPresent()){//check for value is present or

not

String lowercaseString=str[5].toLowerCase();

System.out.print(lowercaseString);

}else

System.out.println("string value is not present");

}

}

Output

String value is not present.

**Java String Joiner**

Java added a new final class String Joiner in java.utilpackage. It is used to construct a sequence of characters separated by a delimiter. Now, you can create string by passing delimiters like comma(,), hyphen(-)etc

Example

Import java.util.StringJoiner;

public class Example{

public static void main(String[] args){

StringJoiner stringJoiner=new StringJoiner(",");//passing

comma(,) as delimiter

//Adding values to String Joiner

stringJoiner.add("Ram");

stringJoiner.add("Shyam");

stringJoiner.add("ashok");

stringJoiner.add("ajay");

System.out.println(stringJoiner);

}

}

Output

Ram, Shyam, ashok, ajay