**\*\*\*\*\*\*\*\*\*\*\*\*Core Java 8 Classroom Demos\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Path Interface\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

import java.io.IOException;

import java.nio.file.DirectoryStream;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.util.List;

public class ListingDir {

public static void main(String[] args) throws IOException {

Path javaHome= Paths.get("C:/Program Files/Java/Jdk8");

System.out.println(javaHome.getNameCount()); //3 doesn't count root

System.out.println(javaHome.getRoot()); // C:\

System.out.println(javaHome.getName(0));// Program Files

System.out.println(javaHome.getName(1)); // Java

System.out.println(javaHome.getFileName()); //jdk1.8.0\_25

System.out.println(javaHome.getParent());//C:\Program Files\Java

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

DirectoryStream<Path> contents = Files.newDirectoryStream(javaHome);

for(Path content: contents)

{

System.out.println(content.getFileName());

}

contents.close();

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

Path file = Paths.get("D:/Output.txt");

List<String> lines = Files.readAllLines(file);

for(String line:lines) {

System.out.println(line);

}

System.out.println("End of File....");

}

}

**\*\*\*\*\*\*\*\*\*\*\*OutPut\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

3

C:\

Program Files

Java

Jdk8

C:\Program Files\Java

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

bin

COPYRIGHT

db

include

javafx-src.zip

jre

lib

LICENSE

README.html

release

src.zip

THIRDPARTYLICENSEREADME-JAVAFX.txt

THIRDPARTYLICENSEREADME.txt

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Hello

How r u?

I am fine.

End of File....

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Generics\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

import java.util.\*;

public class GenericsDemo1

{

public static void add(List<? extends Number> lss)

{

for(Iterator<? extends Number> i=lss.iterator();i.hasNext();)

{

Number n= i.next();

System.*out*.println(" n="+n);

}

//lss.add(new Integer(55));//This is invalid

}

public static <T,S extends T >void copy(List<T> newlist,List<S> srclist)

{

for(Iterator<S> i=srclist.iterator();i.hasNext();)

{

S n= i.next();

newlist.add(n);

}

System.*out*.println(" copied list="+newlist);

}

public static <E> void rev(List<E> lss)

{

for(Iterator<E> i=lss.iterator();i.hasNext();)

{

E n= i.next();

System.*out*.println(" rev method n="+n);

}

//lss.add(new Integer(55));//This is invalid

}

public static void main(String[] args)

{

List<Integer> ls=new LinkedList<Integer>();

List<Number> ns=new LinkedList<Number>();

ls.add(new Integer(10));

ls.add(new Integer(5));

*add*(ls);

*rev*(ls);

*copy*(ns,ls);

List<String> ss=new LinkedList<String>();

ss.add(new String("aaa"));

ss.add(new String("bbb"));

//add(ss);//invalid compiler error

*rev*(ss);

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*Output\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

n=10

n=5

rev method n=10

rev method n=5

copied list=[10, 5]

rev method n=aaa

rev method n=bbb

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Test6.java\*\*\*\*\*\*\*\*\*\***

import java.util.\*;

public class Test6

{

public static void main(String[] args)

{

List<Integer> intList=new LinkedList<Integer>();

intList.add(78);

intList.add(90);

intList.add(56);

Test6 tt=new Test6();

System.*out*.println(tt.sum(intList));

}

public int sum(List<? extends Number> ls)

{

int sum=0;

Iterator<? extends Number> it=ls.iterator();

while(it.hasNext())

{

Number i=it.next();

sum=sum+i.intValue();

}

return sum;

}}

**\*\*\*\*\*\*\*\*\*\*\*output is 224\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*Test7.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

import java.util.\*;

public class Test7

{

public static void main(String[] args)

{

List<Integer> intL=new LinkedList<Integer>();

intL.add(90);

intL.add(34);

intL.add(21);

intL.add(55);

Test7 tt=new Test7();

tt.reverse(intL);

}

public <E> void reverse(List<E>tlist)

{

for(int i=0;i<(tlist.size()/2);i++)

{

int j=tlist.size()-1-i;

E e=tlist.get(i);

tlist.set(i,tlist.get(j));

tlist.set(j, e);

}

System.*out*.println(tlist);

}

}

\*\*\*\*\*output is [55, 21, 34, 90]\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Get.java\*\*\*\*\*\*\*\*\*\*\*\*\*\***

public class Get<T>

{

T o;

int age;

Get(T o,int age)

{

this.o=o;

this.age=age;

}

T getOb()

{

return o;

}

public String toString()

{

return (o +" "+age);

}}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*TestGet.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

public class TestGet

{

public static void main(String[] args)

{

Get<String> g1=new Get<String>("Vaishali",43);

System.*out*.println(g1.getOb());

System.*out*.println(g1);

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*output is\*\*\*\*\*\*\*\*\*\*\*\*\***

Vaishali

Vaishali 43

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Streams\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*BasicStream.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.igate.demos;

import java.util.Arrays;

import java.util.List;

import java.util.stream.Stream;

public class BasicStream {

public static void main(String[] args) {

//stream can be created with static data

Stream<String> stream = Stream.of("I","G","A","T","E","\n");

stream.forEach((location)->System.out.print(location));

//stream can be acquired from array or collections

List<String> locations = Arrays.asList(new String[]{"Pune","Mumbai","Chennai","Banglore","Noida"});

stream = locations.stream();

stream.forEach(System.out::println);

}

}

\*\*\*\*\*\*\*\*\*\*output\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

IGATE Pune Mumbai Chennai Banglore Noida

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Collection Lambda.java\*\*\*\*\*\*\*\*\*\*\***

package com.igate.demos;

import java.util.ArrayList;

import java.util.List;

public class CollectionLambda {

public static void main(String[] args) {

List<String> cities = new ArrayList<>();

cities.add("Pune");

cities.add("Banglore");

cities.add("Mumbai");

cities.add("Chennai");

cities.forEach(city->System.out.println(city));

cities.forEach(System.out::println);//same

}}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*ComparableLambda.java\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.igate.demos;

import java.util.Arrays;

import java.util.Collections;

import java.util.Comparator;

public class ComparableLambda {

public static void main(String[] args) {

Comparator<String> comp = (s1, s2) -> Integer.compare(

s1.length(), s2.length());

String[] values = {"\*\*\*\*","\*\*\*","\*","\*\*"};

Collections.sort(Arrays.asList(values),comp);

for(String value:values) {

System.out.println(value);

}

}

}

**\*\*\*\*\*\*\*output\*\*\*\*\*\*\*\*\*\*\***

\*

\*\*

\*\*\*

\*\*\*\*

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Constructor Reference.java\*\*\*\*\*\***

**package** com.igate.demos;

**import** java.util.function.Supplier;

**public** **class** ContructorReference {

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

Supplier<Item> s1 = Item::**new**;

System.***out***.println(s1.get().getName());

System.***out***.println(s1.get().getPrice());

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*output\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Unknown

0.0

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*File\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.igate.demos;

import java.io.File;

import java.io.FileFilter;

public class FileFilterLambda {

public static void main(String[] args) {

FileFilter filter = (File pathname) -> pathname.getName().endsWith(".txt");

File dir = new File("D:/");

File[] contents = dir.listFiles(filter);

for(File file:contents) {

System.out.println(file);

}

}

}

\*\*\*\*\*\*output\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

D:/output.txt\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*LocationFilter.java\*\*\*\*\*\*\*\*\*\*\***

package com.igate.demos;

import java.util.Arrays;

import java.util.List;

import java.util.stream.Stream;

public class Filtering {

public static void main(String[] args) {

List<String> locations = Arrays.asList(new String[]{"Pune","Mumbai","Chennai","Banglore","Noida"});

Stream<String> stream = locations.stream();

stream.filter((location)->location.length()>5).forEach(System.out::println);

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Mapping .java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.igate.demos;

import java.util.Arrays;

import java.util.List;

import java.util.stream.Stream;

public class Mapping {

public static void main(String[] args) {

List<String> locations = Arrays.asList(new String[]{"Pune","Mumbai","Chennai","Banglore","Noida"});

Stream<String> stream = locations.stream();

System.out.println("Word legnth for locations:");

stream.map(String::length).forEach(System.out::print);

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*Lambda Expression\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.igate.demos;

import java.util.Arrays;

import java.util.Collections;

import java.util.List;

public class MethodReference {

public static void main(String[] args) {

List<String> values = Arrays.asList("one","Two","three","Four","five");

Collections.sort(values,String::compareToIgnoreCase);

System.out.println(values);

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*output\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

[five, Four, one, three, Two]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**\*\*\*\*\*\*\*RunnableLambda.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.igate.demos;

public class RunnableLambda {

public static void main(String[] args) {

Runnable run = ()-> System.*out*.println("Hello from thread");

new Thread(run).start();

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*output\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Hello from thread

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

package com.igate.demos;

public class SimpleLamda {

public static void main(String[] args) {

Runnable r1 = new Runnable() {

@Override

public void run() {

// TODO Auto-generated method stub

System.*out*.println("Hello from thread!");

}

};

Runnable r2 = () -> {

System.*out*.println("Hello from lambda!");

};

new Thread(r1).start();

new Thread(r2).start();

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Hello from thread!

Hello from lambda!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**\*\*\*\*\*\*\*\*\*\*\*\*\*Functionalinterface.java\*\*\*\*\*\*\*\*\*\*\*\***

package com.igate.demos;

import java.util.function.BiFunction;

import java.util.function.BinaryOperator;

import java.util.function.Consumer;

import java.util.function.Predicate;

import java.util.function.Supplier;

public class FunctionalInterfaces {

public static void main(String[] args) {

Consumer<String> consumer = (String str)-> System.out.println(str);

consumer.accept("Hello LE!");

Supplier<String> supplier = () -> "Hello from Supplier!";

consumer.accept(supplier.get());

//even number test

Predicate<Integer> predicate = num -> num%2==0;

System.out.println(predicate.test(24));

System.out.println(predicate.test(15));

//max or min test

BiFunction<Integer, Integer, Integer> maxFunction = (x,y)->x>y?x:y;

System.out.println(maxFunction.apply(25, 14));

}

}

\*\*\*\*\*\*\*\*\*\*\*\*output\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Hello LE!

Hello from Supplier!

true

false

25

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DateAPIUtilities.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

/\* \*\*\*\*\*\*\*\*\*\*\*\*output\*\*\*\*\*\*\*\*\*\*\*\*

Year 2015 is Leap Year? false

Today is before 01/01/2015? false

Current Time=2015-05-25T11:01:28.590

10 days after today will be 2015-06-04

3 weeks after today will be 2015-06-15

20 months after today will be 2017-01-25

10 days before today will be 2015-05-15

3 weeks before today will be 2015-05-04

20 months before today will be 2013-09-25

First date of this month= 2015-05-01

Last date of this year= 2015-12-31

Period Format= P7M6D

Months remaining in the year= 7

\*/

import java.time.LocalDate;

import java.time.LocalTime;

import java.time.Period;

import java.time.temporal.TemporalAdjusters;

public class DateAPIUtilities {

public static void main(String[] args) {

LocalDate today = LocalDate.now();

//Get the Year, check if it's leap year

System.out.println("Year "+today.getYear()+" is Leap Year? "+today.isLeapYear());

//Compare two LocalDate for before and after

System.out.println("Today is before 01/01/2015? "+today.isBefore(LocalDate.of(2015,1,1)));

//Create LocalDateTime from LocalDate

System.out.println("Current Time="+today.atTime(LocalTime.now()));

//plus and minus operations

System.out.println("10 days after today will be "+today.plusDays(10));

System.out.println("3 weeks after today will be "+today.plusWeeks(3));

System.out.println("20 months after today will be "+today.plusMonths(20));

System.out.println("10 days before today will be "+today.minusDays(10));

System.out.println("3 weeks before today will be "+today.minusWeeks(3));

System.out.println("20 months before today will be "+today.minusMonths(20));

//Temporal adjusters for adjusting the dates

System.out.println("First date of this month= "+today.with(TemporalAdjusters.firstDayOfMonth()));

LocalDate lastDayOfYear = today.with(TemporalAdjusters.lastDayOfYear());

System.out.println("Last date of this year= "+lastDayOfYear);

Period period = today.until(lastDayOfYear);

System.out.println("Period Format= "+period);

System.out.println("Months remaining in the year= "+period.getMonths());

Period per= Period.between(LocalDate.of(2015,1,1),today );

System.out.println(per.getMonths());

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DateParseFormatExample.java\*\*\*\*\*\*\*\*\*\*\*\*\***

**/\*\*\*\*\*\*\*\*\*\*\*\*Output\*\*\*\*\*\*\*\*\*\*\*\*\*\***

Default format of LocalDate=2015-05-25

25::May::2015

20150525

Default format of LocalDateTime=2015-05-25T11:08:46.097

25::May::2015 11::08::46

20150525

Default format of Instant=2015-05-25T05:38:46.097Z

Default format after parsing = 2014-04-27T21:39:48

\*/

import java.time.Instant;

import java.time.LocalDate;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

public class DateParseFormatExample {

public static void main(String[] args) {

//Format examples

LocalDate date = LocalDate.now();

//default format

System.out.println("Default format of LocalDate="+date);

//specific format

System.out.println(date.format(DateTimeFormatter.ofPattern("d::MMM::uuuu")));

System.out.println(date.format(DateTimeFormatter.BASIC\_ISO\_DATE));

LocalDateTime dateTime = LocalDateTime.now();

//default format

System.out.println("Default format of LocalDateTime="+dateTime);

//specific format

System.out.println(dateTime.format(DateTimeFormatter.ofPattern("d::MMM::uuuu HH::mm::ss")));

System.out.println(dateTime.format(DateTimeFormatter.BASIC\_ISO\_DATE));

Instant timestamp = Instant.now();

//default format

System.out.println("Default format of Instant="+timestamp);

//Parse examples

LocalDateTime dt = LocalDateTime.parse("27::Apr::2014 21::39::48",

DateTimeFormatter.ofPattern("d::MMM::uuuu HH::mm::ss"));

System.out.println("Default format after parsing = "+dt);

System.out.println("d-m-u format"+date.format(DateTimeFormatter.ofPattern("d::MM::u")));

System.out.println("d-m-uu format"+date.format(DateTimeFormatter.ofPattern("d::MM::uu")));

System.out.println("d-m-uuuuu format"+date.format(DateTimeFormatter.ofPattern("d::MM::uuuu")));

System.out.println("d-m-y format"+date.format(DateTimeFormatter.ofPattern("d::MM::y")));

System.out.println("d-m-yy format"+date.format(DateTimeFormatter.ofPattern("d::MM::yy")));

System.out.println("d-m-yyyy format"+date.format(DateTimeFormatter.ofPattern("d::MM::yyyy")));

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*LocalDateExample.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**/\*\*\*\*\*\*\*OutPut\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

\* Current Date=2015-05-25

Specific Date=2014-01-01

Current Date in IST=2015-05-25

365th day from base date= 1971-01-01

100th day of 2014=2014-04-10

\*/

import java.time.LocalDate;

import java.time.Month;

import java.time.ZoneId;

public class LocalDateExample

{

public static void main(String[] args)

{

//Current Date

LocalDate today = LocalDate.now();

System.out.println("Current Date="+today);

//Creating LocalDate by providing input arguments

LocalDate firstDay\_2014 = LocalDate.of(2014, Month.JANUARY, 1);

System.out.println("Specific Date="+firstDay\_2014);

//Try creating date by providing invalid inputs

//LocalDate feb29\_2014 = LocalDate.of(2014, Month.FEBRUARY, 29);

//Exception in thread "main" java.time.DateTimeException:

//Invalid date 'February 29' as '2014' is not a leap year

//Current date in "Asia/Kolkata", you can get it from ZoneId javadoc

LocalDate todayKolkata = LocalDate.now(ZoneId.of("Asia/Kolkata"));

System.out.println("Current Date in IST="+todayKolkata);

//java.time.zone.ZoneRulesException: Unknown time-zone ID: IST

//LocalDate todayIST = LocalDate.now(ZoneId.of("IST"));

//Getting date from the base date i.e 01/01/1970

LocalDate dateFromBase = LocalDate.ofEpochDay(365);

System.out.println("365th day from base date= "+dateFromBase);

LocalDate hundredDay2014 = LocalDate.ofYearDay(2014, 100);

System.out.println("100th day of 2014="+hundredDay2014);

} }

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*LocalDateTimeExample.java\*\*\*\*\*\*\*\*\*\***

/\*\*\*\*\*\*\*\*\*Output\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Current DateTime=2015-05-25T10:41:22.091

Current DateTime=2015-05-25T10:41:22.091

Specific Date=2014-01-01T10:10:30

Current Date in IST=2015-05-25T10:41:22.091

10000th second time from 01/01/1970= 1970-01-01T02:46:40

\*/

import java.time.LocalDate;

import java.time.LocalDateTime;

import java.time.LocalTime;

import java.time.Month;

import java.time.ZoneId;

import java.time.ZoneOffset;

public class LocalDateTimeExample {

public static void main(String[] args) {

//Current Date

LocalDateTime today = LocalDateTime.now();

System.out.println("Current DateTime="+today);

//Current Date using LocalDate and LocalTime

today = LocalDateTime.of(LocalDate.now(), LocalTime.now());

System.out.println("Current DateTime="+today);

//Creating LocalDateTime by providing input arguments

LocalDateTime specificDate = LocalDateTime.of(2014, Month.JANUARY, 1, 10, 10, 30);

System.out.println("Specific Date="+specificDate);

//Try creating date by providing invalid inputs

// LocalDateTime feb29\_2014 = LocalDateTime.of(2014, Month.FEBRUARY, 28, 25,1,1);

//Exception in thread "main" java.time.DateTimeException:

//Invalid value for HourOfDay (valid values 0 - 23): 25

//Current date in "Asia/Kolkata", you can get it from ZoneId javadoc

LocalDateTime todayKolkata = LocalDateTime.now(ZoneId.of("Asia/Kolkata"));

System.out.println("Current Date in IST="+todayKolkata);

//java.time.zone.ZoneRulesException: Unknown time-zone ID: IST

// LocalDateTime todayIST = LocalDateTime.now(ZoneId.of("IST"));

//Getting date from the base date i.e 01/01/1970

LocalDateTime dateFromBase = LocalDateTime.ofEpochSecond(10000, 0, ZoneOffset.UTC);

System.out.println("10000th second time from 01/01/1970= "+dateFromBase);

}}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Lambda Expression\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MaxFinder.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

@FunctionalInterface

public interface MaxFinder

{

//single abstract method to find max between two numbers

public int maximum(int num1,int num2);

}

**\*\*\*\*\*\*MaxFinderImpl.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

public class MaxFinderImpl implements MaxFinder

{

@Override

public int maximum(int num1, int num2)

{

return num1>num2?num1:num2;

}

//Is it worthy to create separate class for single method implementation?

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*TestMaxFinderLambdaDemo.java\*\*\*\*\*\*\*\***

**public** **class** TestMaxFinderLambdaDemo {

public static void main(String[] args)

{

MaxFinder finder = new MaxFinderImpl();

int result = finder.maximum(190, 20);

System.*out*.println("Greatest Is :"+result);

//same thing without functional interface

//implementation is done using lamda expression.

MaxFinder finder2 = (num1,num2) -> num1>num2?num1:num2;

int result2 = finder.maximum(10, 20);

System.*out*.println("Greatest Is :"+result2);

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*BuiltInFunctionalInterface.java\*\*\*\*\*\*\*\*\*\*\*\*\***

import java.util.function.BiFunction;

import java.util.function.BinaryOperator;

import java.util.function.Consumer;

import java.util.function.Predicate;

import java.util.function.Supplier;

public class BuiltInFunctionalInterface

{

public static void main(String args[])

{

Consumer<String> consumer = (String str)-> System.out.println(str);

consumer.accept("Hello LE!");

Supplier<String> supplier = () -> "Hello from Supplier!";

consumer.accept(supplier.get());

//even number test

Predicate<Integer> predicate = num -> num%2==0;

System.out.println(" Is Even ?"+predicate.test(24));

System.out.println(" Is Even ?"+predicate.test(15));

//max test

BiFunction<Integer, Integer, Integer> maxFunction = (x,y)->x>y?x:y;

System.out.println(maxFunction.apply(25, 14));

BinaryOperator<Integer> maxFunction2 = (x,y) -> x>y?x:y;

System.out.println(maxFunction2.apply(45, 5));

//Method References

Consumer<String> consumer2 = System.out :: println;

consumer2.accept("Hello IGattors using Method Reference Lmda Expression!");

}}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Person.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

public class Person

{

int age;

String name;

public int getAge() { return age;}

public void setAge(int age){ this.age = age;}

public String getName() { return name;}

public void setName(String name){ this.name = name;}

public Person(String name,int age)

{

super();

this.age = age;

this.name = name;

}

@Override

public String toString() { return "Person [age=" + age + ", name=" + name + "]";}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*TestStreamAPIDemo\_7.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

import java.util.ArrayList;

import java.util.Arrays;

import java.util.IntSummaryStatistics;

import java.util.List;

import java.util.Optional;

import java.util.Random;

import java.util.function.Predicate;

import java.util.stream.Collectors;

import java.util.stream.Stream;

public class TestStreamAPIDemo\_7 {

public static void main(String[] args)

{

System.out.println("\*\*\*\*\*\*\*\*\*\*Empty String\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Using Java 7: ");

// Count empty strings

List<String> strings = Arrays.asList("abc", "", "bc", "efg", "abcd","", "jkl");

System.out.println("List: " +strings);

long count = getCountEmptyStringUsingJava7(strings);

System.out.println("Empty Strings: " + count);

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*String with length 3\*\*\*\*");

count = getCountLength3UsingJava7(strings);

System.out.println("Strings of length 3: " + count);

System.out.println("\*\*\*\*\*\*\*\*Remove Empty String\*\*\*\*\*\*\*\*\*\*\*");

//Eliminate empty string

List<String> filtered = deleteEmptyStringsUsingJava7(strings);

System.out.println("Filtered List: " + filtered);

System.out.println("\*\*\*\*\*\*Remove empty and join with comma\*\*\*\*\*\*\*\*\*\*\*");

//Eliminate empty string and join using comma.

String mergedString = getMergedStringUsingJava7(strings,", ");

System.out.println("Merged String: " + mergedString);

System.out.println("\*\*\*\*\*\*Find Squares of Number\*\*\*\*\*\*\*\*\*\*\*");

List<Integer> numbers = Arrays.asList(3, 2, 2, 3, 7, 3, 5);

//get list of square of distinct numbers

List<Integer> squaresList = getSquares(numbers);

System.out.println("Squares List: " + squaresList);

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*Other Utility Functions\*\*\*\*\*\*\*\*");

List<Integer> integers = Arrays.asList(1,2,13,4,15,6,17,8,19);

System.out.println("List: " +integers);

System.out.println("Highest number in List : " + getMax(integers));

System.out.println("Lowest number in List : " + getMin(integers));

System.out.println("Sum of all numbers : " + getSum(integers));

System.out.println("Average of all numbers : " + getAverage(integers));

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*Random Numbers: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

//print ten random numbers

Random random = new Random();

for(int i=0; i < 10; i++)

{

System.out.println(random.nextInt());

}

}

private static int getCountEmptyStringUsingJava7(List<String> strings)

{

int count = 0;

for(String string: strings)

{

if(string.isEmpty())

{

count++;

}

}

return count;

}

private static int getCountLength3UsingJava7(List<String> strings)

{

int count = 0;

for(String string: strings)

{

if(string.length() == 3)

{

count++;

}

}

return count;

}

private static List<String> deleteEmptyStringsUsingJava7(List<String> strings)

{

List<String> filteredList = new ArrayList<String>();

for(String string: strings)

{

if(!string.isEmpty())

{

filteredList.add(string);

}

}

return filteredList;

}

private static String getMergedStringUsingJava7(List<String> strings, String seperator)

{

StringBuilder stringBuilder = new StringBuilder();

for(String string: strings)

{

if(!string.isEmpty())

{

stringBuilder.append(string);

stringBuilder.append(seperator);

}

}

String mergedString = stringBuilder.toString();

return mergedString.substring(0, mergedString.length()-2);

}

private static List<Integer> getSquares(List<Integer> numbers)

{

List<Integer> squaresList = new ArrayList<Integer>();

for(Integer number: numbers)

{

Integer square = new Integer(number.intValue() \* number.intValue());

if(!squaresList.contains(square))

{

squaresList.add(square);

}

}

return squaresList;

}

private static int getMax(List<Integer> numbers)

{

int max = numbers.get(0);

for(int i=1;i< numbers.size();i++)

{

Integer number = numbers.get(i);

if(number.intValue() > max){

max = number.intValue();

}

}

return max;

}

private static int getMin(List<Integer> numbers)

{

int min = numbers.get(0);

for(int i=1;i< numbers.size();i++)

{

Integer number = numbers.get(i);

if(number.intValue() < min){

min = number.intValue();

}

}

return min;

}

private static int getSum(List<Integer> numbers)

{

int sum = numbers.get(0);

for(int i=1;i< numbers.size();i++){

sum += numbers.get(i).intValue();

}

return sum;

}

private static int getAverage(List<Integer> numbers){

return getSum(numbers) / numbers.size();

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*TestStreamAPIDemo\_8.java\*\*\*\*\*\*\*\*\*\*\*\*\***

/\*

\* \*\*\*\*\*\*\*\*\*\*\*\*Stream API Using Java 8: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*filter\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*Stream API Using Java 8: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*filter\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Original Int List: [11, 3, 44, 5, 66, 33, 44]

1144663344\*\*\*\*\*\*\*\*\*\*\*\*\*\*distict\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Original List : [AA, AA, BB]

Number of distinct element:2

\*\*\*\*\*Distinct Using foreach\*\*\*\*\*\*\*\*

Original : [11, 3, 44, 5, 66, 33, 44]

11

3

44

5

66

33

\*\*\*\*\*\*\*\*\*\*\*\*Limit\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Original : [11, 3, 44, 5, 66, 33, 44]

First 4 elements :

11

3

44

5

\*\*\*\*\*\*\*\*\*\*\*Length Of Each Word using map\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Original wordsList: [IGATE, GLOBAL, SOLUTIONS]

5

6

9

\*\*\*\*\*\*\*\*Empty String Count \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Original Strings : [abc, , bc, efg, abcd, , jkl]

Empty Strings Count Is : 2

\*\*\*\*\*\*\*\*String With Length 3\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Original Strings : [abc, , bc, efg, abcd, , jkl]

Strings of length 3: 3

\*\*\*\*\*\*\*\*Non Empty String List\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Original Strings : [abc, , bc, efg, abcd, , jkl]

Filtered Non Empty List: [abc, bc, efg, abcd, jkl]

\*\*\*\*\*Merged Non Empty String Using Comma \*\*\*\*\*\*\*\*\*\*\*\*\*

Original Strings : [abc, , bc, efg, abcd, , jkl]

Merged Non Empty String Using Comma : abc,bc,efg,abcd,jkl

\*\*\*\*\*\*\*\*Number List With square\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Original numberes : [3, 2, 2, 3, 7, 3, 5]

Squares List: [9, 4, 49, 25]

\*\*\*\*\*\*\*\*Use Of IntSummerStatistic\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Original Integer List: [1, 2, 13, 4, 15, 6, 17, 8, 19]

Highest number in List : 19

Lowest number in List : 1

Sum of all numbers : 85

Average of all numbers : 9.444444444444445

\*\*\*\*\*\*\*\*\*\*Random Numbers:\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

-2084717139

-1670905023

-1581655105

-1537215599

-1378156139

-914507298

-730747183

1041478155

1081412895

1679707721

\*\*\*\*\*\*\*\*\*\*parallel processing\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Original Strings : [abc, , bc, efg, abcd, , jkl]

Empty Strings: 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Concat\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[a, b, c, d, e, f, g]

\*\*\*\*\*\*\*\*\*\*\*collect\*\*\*\*\*\*\*\*\*\*\*\*

12:Samir, 16:Vaishali

\*\*\*\*\*\*Collect method other way round\*\*\*\*\*\*\*\*

21:Ola Hansen, 30:Nikhil, 45:Vishal, 89:Karthik

\*\*\*\*\*\*\*\*\*\*\*stream reduce function\*\*\*\*\*\*\*\*\*\*\*\*

Result:24

\*/

import java.util.ArrayList;

import java.util.Arrays;

import java.util.IntSummaryStatistics;

import java.util.List;

import java.util.Optional;

import java.util.Random;

import java.util.function.Predicate;

import java.util.stream.Collector;

import java.util.stream.Collectors;

import java.util.stream.IntStream;

import java.util.stream.Stream;

public class TestStreamAPIDemo\_8 {

public static void main(String[] args)

{

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*Stream API Using Java 8: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("\*\*\*\*\*\*\*\*filter\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

List<Integer> listInt1 = Arrays.asList(11,3,44,5,66,33,44);

System.out.println("Original Int List: "+listInt1);

listInt1.stream().filter(num -> num > 10).forEach(num->System.out.print(num));

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*distict\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

List<String> list = Arrays.asList("AA","AA","BB");

System.out.println("Original List : "+list);

Stream listDistinct=list.stream().distinct();

long distictCount = listDistinct.count();

System.out.println("Number of distinct element:"+distictCount);

System.out.println("\*\*\*\*\*Distinct Using foreach\*\*\*\*\*\*\*\*");

List<Integer> listInt2 = Arrays.asList(11,3,44,5,66,33,44);

System.out.println("Original : "+listInt2);

listInt2.stream().distinct().forEach(System.out :: println);

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*Limit\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

List<Integer> listInt3 = Arrays.asList(11,3,44,5,66,33,44);

System.out.println("Original : "+listInt3);

System.out.println("First 4 elements : ");

listInt3.stream().limit(4).forEach(System.out :: println);

System.out.println("\*\*\*\*\*\*\*\*\*\*\*Length Of Each Word using map\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

List<String> wordsList = Arrays.asList("IGATE","GLOBAL","SOLUTIONS");

System.out.println("Original wordsList: "+wordsList);

Stream<Object> wordsListStream=wordsList.stream().map(str->str.length());

wordsListStream.forEach(System.out :: println);

//wordsList.stream().map(str->str.length()).forEach(System.out :: println);

System.out.println("\*\*\*\*\*\*\*\*Empty String Count \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

List<String> strings = Arrays.asList("abc", "", "bc", "efg", "abcd","", "jkl");

System.out.println("Original Strings : " +strings);

Stream<String> strmS=strings.stream();

Predicate<String> predicate=(String string)->string.isEmpty();

Stream strmD=strmS.filter(predicate);

long count = strmD.count();

System.out.println("Empty Strings Count Is : " + count);

System.out.println("\*\*\*\*\*\*\*\*String With Length 3\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

count = strings.stream().filter(string -> string.length() == 3).count();

System.out.println("Original Strings : " +strings);

System.out.println("Strings of length 3: " + count);

System.out.println("\*\*\*\*\*\*\*\*Non Empty String List\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Original Strings : " +strings);

Stream<String> nonEmptyFilteredList=strings.stream().filter(string -> !string.isEmpty());

/\*

Returns a Collector that accumulates the input elements into a new List.

There are no guarantees on the type, mutability, serializability, or

thread-safety of the List returned; if more control over the returned List is required,

use toCollection(Supplier).

Parameters:

<T> the type of the input elements

Returns:

Collector which collects all the input elements into a List,

in encounter order\*/

List<String> newNonEmptyList=nonEmptyFilteredList.collect(Collectors.toList());

//List<String> filtered = strings.stream().filter(string -> !string.isEmpty()).

//collect(Collectors.toList());

System.out.println("Filtered Non Empty List: " + newNonEmptyList);

System.out.println("\*\*\*\*\*Merged Non Empty String Using Comma \*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Original Strings : " +strings);

/\*

\* Returns a Collector that concatenates the input elements,

\* separated by the specified delimiter, in encounter order.

Parameters:

delimiter the delimiter to be used between each element

Returns:

A Collector which concatenates CharSequence elements,

separated by the specified delimiter, in encounter order

\*/

String mergedString = strings.stream().filter(string -> !string.isEmpty()).

collect(Collectors.joining(","));

System.out.println("Merged Non Empty String Using Comma : " + mergedString);

System.out.println("\*\*\*\*\*\*\*\*Number List With square\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

List<Integer> numbers = Arrays.asList(3, 2, 2, 3, 7, 3, 5);

System.out.println("Original numberes : " +numbers);

Stream<Integer> squareStream=numbers.stream().map( i -> i\*i);

Stream<Integer> squareDistStream=squareStream.distinct();

List<Integer> squaresList=squareDistStream.collect(Collectors.toList());

System.out.println("Squares List: " + squaresList);

System.out.println("\*\*\*\*\*\*\*\*Use Of IntSummerStatistic\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

List<Integer> integers = Arrays.asList(1,2,13,4,15,6,17,8,19);

System.out.println("Original Integer List: " +integers);

Stream<Integer> integerStream=integers.stream();

/\* Returns an IntStream consisting of the results of applying the

\* given function to the elements of this stream.

This is an intermediate operation.

Parameters:

mapper a non-interfering, stateless function to apply to each element

Returns:

the new stream

\*/

IntStream integerMapStream=integerStream.mapToInt((x) -> x);

//IntSummaryStatistics stats = integers.stream().mapToInt((x) -> x).summaryStatistics();

/\* Returns an IntSummaryStatistics describing various summary data

\* about the elements of this stream. This is a special case of a reduction.

This is a terminal operation.

Returns:

an IntSummaryStatistics describing various summary data about the elements of this stream

\*/

IntSummaryStatistics stats=integerMapStream.summaryStatistics();

System.out.println("Highest number in List : " + stats.getMax());

System.out.println("Lowest number in List : " + stats.getMin());

System.out.println("Sum of all numbers : " + stats.getSum());

System.out.println("Average of all numbers : " + stats.getAverage());

System.out.println("\*\*\*\*\*\*\*\*\*\*Random Numbers:\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ");

Random random = new Random();

random.ints().limit(10).sorted().forEach(System.out::println);

//parallel processing

System.out.println("\*\*\*\*\*\*\*\*\*\*parallel processing\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ");

System.out.println("Original Strings : " +strings);

count = strings.parallelStream().filter(string -> string.isEmpty()).count();

System.out.println("Empty Strings: " + count);

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Concat\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

String[] arr1 = { "a", "b", "c", "d" };

String[] arr2 = { "e", "f", "g" };

Stream<String> stream1 = Stream.of(arr1);

Stream<String> stream2 = Stream.of(arr2);

Stream<String> stream3 = Stream.concat(stream1, stream2);

String[] arr = stream3.toArray(String[]::new);

System.out.println(Arrays.toString(arr));

System.out.println("\*\*\*\*\*\*\*\*\*\*\*collect\*\*\*\*\*\*\*\*\*\*\*\*");

//Data

List<Person> persons = new ArrayList<>();

persons.add(new Person("Ola Hansen", 21));

persons.add(new Person("Karthik", 89));

persons.add(new Person("Samir", 12));

persons.add(new Person("Vaishali", 16));

persons.add(new Person("Vishal", 45));

persons.add(new Person("Nikhil", 30));

//Solution

String names = persons.stream()

.filter(p -> p.getAge() < 18)

.sorted((p1, p2) -> p1.getAge() - p2.getAge())

.map(p -> p.getAge() + ":" + p.getName())

.collect(Collectors.joining(", "));

System.out.println(names);

System.out.println("\*\*\*\*\*\*Collect method other way round\*\*\*\*\*\*\*\*");

List<Person> persons2 = new ArrayList<>();

persons2.add(new Person("Ola Hansen", 21));

persons2.add(new Person("Karthik", 89));

persons2.add(new Person("Samir", 12));

persons2.add(new Person("Vaishali", 16));

persons2.add(new Person("Vishal", 45));

persons2.add(new Person("Nikhil", 30));

Stream<Person> strmOriginalPerson=persons2.stream();

Predicate<Person> predicateAdult=(Person per)->per.getAge() > 18;

Stream<Person> streamAdult=strmOriginalPerson.filter(predicateAdult);

Stream <Person>streamSorted=streamAdult.sorted((p1, p2) -> p1.getAge() - p2.getAge());//para pass in sorted is comparator

Stream streamMapped =streamSorted.map(p -> p.getAge() + ":" + p.getName());

String namesAdults=(String)streamMapped.collect(Collectors.joining(", "));

System.out.println(namesAdults);

System.out.println("\*\*\*\*\*\*\*\*\*\*\*stream reduce function\*\*\*\*\*\*\*\*\*\*\*\*");

List<Integer> intList = Arrays.asList(5,7,3,9);

Optional<Integer> result = intList.stream().reduce((a,b)->a+b);

if(result.isPresent())

{

System.out.println("Result:"+result.get());

}}}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Java8Tester.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**public** **class** Java8Tester

{

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

Java8Tester tester = **new** Java8Tester();

//with type declaration

MathOperation addition = (**int** a, **int** b) -> a + b;

//with out type declaration

MathOperation subtraction = (a, b) -> a - b;

//with return statement along with curly braces

MathOperation multiplication = (**int** a, **int** b) -> { **return** a \* b; };

//without return statement and without curly braces

MathOperation division = (**int** a, **int** b) -> a / b;

System.***out***.println("10 + 5 = " + tester.operate(10, 5, ((**int** a, **int** b) -> a + b)));

System.***out***.println("10 - 5 = " + tester.operate(10, 5, subtraction));

System.***out***.println("10 x 5 = " + tester.operate(10, 5, multiplication));

System.***out***.println("10 / 5 = " + tester.operate(10, 5, division));

//with parenthesis

GreetingService greetService1 = message -> System.***out***.println("Hello " + message);

//without parenthesis

GreetingService greetService2 = (message) -> System.***out***.println("Hello " + message);

greetService1.sayMessage("Mahesh");

greetService2.sayMessage("Suresh");

}

interface MathOperation { int operation(int a, int b); }

interface GreetingService { void sayMessage(String message); }

private int operate(int a, int b, MathOperation mathOperation){

return mathOperation.operation(a, b);

} }

**\*\*\*\*\*\*\*\*\*State.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

public class State {

String city;

public String getCity() { return city; }

public void setCity(String city) { this.city = city; }

@Override

public String toString() { return "State [city=" + city + "]"; }

public State(String city) { super(); this.city = city; }

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*StateChangeListener.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

public interface StateChangeListener

{

public void onStateChange(State oldState, State newState);

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*StateOwner.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

public class StateOwner{

public void addStateListener(StateChangeListener listener) {

State oldState=new State("Pune");

State newState=new State("Chennai");

listener.onStateChange(oldState, newState);

System.*out*.println("In owner.....");}

}

**\*\*\*\*\*\*\*\*\*\*\*\*TestStateDemo.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*8**

import java.util.function.Consumer;

public class TestStateDemo {

public static void main(String[] args)

{

StateOwner sow=new StateOwner();

sow.addStateListener(

(oState,nState)->System.*out*.println("Hello...")

);

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*JAXB\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Employee.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.igate.jaxb;

import javax.xml.bind.annotation.XmlAccessType;

import javax.xml.bind.annotation.\*;

@XmlAccessorType(XmlAccessType.FIELD)

@XmlType(name="empType",propOrder={"id","empNameType","sal","gen"})

public class Employee

{

@Override

public String toString() {

return "\n\nEmployee [count=" + count + ", id=" + id + ", empNameType="

+ empNameType + ", sal=" + sal + ", gen=" + gen + "]";

}

@XmlTransient int count;

@XmlAttribute( name="empId",required=true)

private int id;

@XmlElement(name="empName", required=true)

private EmpNameType empNameType;

@XmlElement(name="empSal", required=true)

private float sal;

@XmlElement(name="empGender", required=true)

private char gen;

public EmpNameType getEmpNameType()

{

return empNameType;

}

public void setEmpNameType(EmpNameType empNameType)

{

this.empNameType = empNameType;

}

public int getId(){ return id; }

public void setId(int id) { this.id = id; }

public float getSal() { return sal; }

public void setSal(float sal) { this.sal = sal; }

public char getGen() { return gen; }

public void setGen(char gen) { this.gen = gen; }

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Employees.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.igate.jaxb;

import java.util.List;

import javax.xml.bind.annotation.XmlAccessType;

import javax.xml.bind.annotation.XmlAccessorType;

import javax.xml.bind.annotation.XmlElement;

import javax.xml.bind.annotation.XmlRootElement;

import javax.xml.bind.annotation.XmlType;

@XmlAccessorType(XmlAccessType.FIELD)

@XmlRootElement(name="emps",namespace="com.emp.vinsys")

@XmlType(name="empsType",propOrder={"empList"})

public class Employees

{

@Override

public String toString() {

return "Employess [empList=" + empList + "]";

}

@XmlElement( name="emp",required=true)

private List<Employee> empList;

public List<Employee> getEmpList() { return empList; }

public void setEmpList(List<Employee> empList) { this.empList = empList; }

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* EmpNameType.java\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.igate.jaxb;

import javax.xml.bind.annotation.\*;

@XmlAccessorType(XmlAccessType.*FIELD*)

@XmlType(name="empNameType",propOrder={"fname","mname","lname"})

public class EmpNameType

{

@Override

public String toString() {

return "EmpNameType [fname=" + fname + ", lname=" + lname + ", mname="

+ mname + "]";

}

@XmlElement(name="fn",required=true)

String fname;

@XmlElement(name="ln",required=true)

String lname;

@XmlElement(name="mn",required=true)

String mname;

public String getFname() { return fname; }

public void setFname(String fname) { this.fname = fname; }

public String getLname() { return lname; }

public void setLname(String lname) { this.lname = lname; }

public String getMname() { return mname; }

public void setMname(String mname) { this.mname = mname; }}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\* TestEmpMarshaller.java\*\*\*\*\*\*\*\*\***

package com.igate.jaxb;

import java.io.File;

import java.util.\*;

import javax.xml.bind.JAXBContext;

import javax.xml.bind.Marshaller;

public class TestEmpMarshaller

{

private static final String EMP\_XML = "./emp.xml";

public static void main(String[] args) throws Exception

{

EmpNameType empName1=new EmpNameType();

empName1.setFname("Vaishali");

empName1.setMname("Abhi");

empName1.setLname("Srivastava");

Employee vaiE=new Employee();

vaiE.setGen('F');

vaiE.setEmpNameType(empName1);

vaiE.setId(386);

vaiE.setSal(200000);

EmpNameType empName2=new EmpNameType();

empName2.setFname("Umesh");

empName2.setMname("N");

empName2.setLname("Bahir");

Employee umeshE=new Employee();

umeshE.setGen('M');

umeshE.setEmpNameType(empName2);

umeshE.setId(222);

umeshE.setSal(100000);

Employees emps=new Employees();

ArrayList empList=new ArrayList();

empList.add(vaiE);

empList.add(umeshE);

emps.setEmpList(empList);

// create JAXB context and instantiate marshaller

JAXBContext context = JAXBContext.newInstance(Employees.class);

Marshaller m = context.createMarshaller();

m.setProperty(Marshaller.JAXB\_FORMATTED\_OUTPUT, Boolean.TRUE);

//m.setProperty(Marshaller.JAXB\_ENCODING,Boolean.TRUE);

// m.setProperty(Marshaller.JAXB\_SCHEMA\_LOCATION,Boolean.FALSE);

// Write to System.out

m.marshal(emps, System.out);

// Write to File

m.marshal(emps, new File(EMP\_XML));

// get variables from our xml file, created before

System.out.println();

System.out.println("Output from our XML File: "); }}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* TestEmpUnMarshaller.java\*\*\*\*\*\*\*\*\*\*\*\*\***

package com.igate.unmarshal;

import java.io.File;

import java.io.StringReader;

import java.util.\*;

import javax.xml.bind.JAXBContext;

import javax.xml.bind.Marshaller;

import javax.xml.bind.Unmarshaller;

import com.igate.jaxb.EmpNameType;

import com.igate.jaxb.Employee;

import com.igate.jaxb.Employees;

public class TestEmpUnMarshaller

{

private static final String EMP\_XML = "emp.xml";

public static void main(String[] args) throws Exception

{

// Creating an Unmarshaller

JAXBContext jaxbCtxt = JAXBContext.newInstance(Employees.class);

Unmarshaller jaxbUnmarshaller = jaxbCtxt.createUnmarshaller();

File ff=new File("emp.xml") ;

//Converting XML to java object using JAXB unmarshal API.

Employees emps = (Employees)jaxbUnmarshaller.unmarshal(ff) ;

System.out.println(emps.getEmpList());

}

**\*\*\*\*\*\*\*\*\*\*\*\*Thank You\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***