@FunctionalInterface

public interface I1 {

public void add(int x,int y);

public default void default1() {System.out.println("default1"); }

public static void static1() {System.out.println("static method");}

}

//Lambda Expressions

package com.training.java8;

//Java 8 lambda expressions

public class Demo1 {

public static void main(String[] args) {

I1.static1();

I1 i1=new I1() {

@Override

public void add(int x, int y) {

System.out.println("adding:"+(x+y));

}

};

i1.add(12, 34);

I1 i2 = (int a,int b)->System.out.println("Implementing add:"+(a+b));

i2.add(56, 89);

Thread t1=new Thread(()->{for(int i=0;i<=5;i++) System.out.println(i);});

t1.start();

}

}

package com.training.java8;

import java.util.Arrays;

import java.util.Collections;

import java.util.Comparator;

import java.util.List;

import java.util.stream.Collectors;

import java.util.stream.Stream;

import com.training.bean.Person;

public class Java8\_Examples {

public static void main(String[] args) {

//List<Integer> numbers = Arrays.asList(6,5,4,2,1,9,7,3,8);

// print all even numbers wo stream()

/\*

\* for(Integer n:numbers) { if(n%2==0) System.out.println(n); }

\*/

//print all numbers

/\*

\* numbers.forEach(i->System.out.println(i));

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

\*/

//Java 8

//Stream<Integer> stream1 = numbers.stream();

//stream1.filter(i->i%2==0 && i>4).forEach(i->System.out.println(i));

//stream1.filter(i->i%2==0 && i>4).forEach(System.out::println);

// get all even numbers into another list

/\*

\* List<Integer> evenNumbers

\* =numbers.stream().filter(i->i%2==0).collect(Collectors.toList());

\* System.out.println(evenNumbers);

\*/

// Collect squares of all even numbers

/\*

\* List<Integer> sqNumbers =

\* numbers.stream().filter(i->i%2==0).map(i->i\*i).collect(Collectors.toList());

\* System.out.println(sqNumbers);

\*/

// maximum element of numbers.

// wo java 8

/\*

\* System.out.println(Collections.max(numbers));

\*

\* System.out.println(numbers.stream().max(Integer::compareTo).get());

\*/

List<Person> persons = Arrays.asList(new Person("sam", 23),new Person("john",21),new Person("bob",24));

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

Comparator<Person> c=(Person p1,Person p2)->{return p1.getAge() - p2.getAge();};

System.out.println(persons.stream().max(c).get());

System.out.println(persons.stream().max(Comparator.comparing(Person::getAge)));

persons.stream().sorted(c).forEach(System.out::println);

persons.stream().sorted(Comparator.comparing(Person::getAge)).forEach(System.out::println);

}

}