**Core Java Assignment 8**

public class Lamba1 {

public static void main(String[] args) {

int a=11;

int b=9;

Arithmatic add = (c,d) -> System.out.println("Addition : " + (c+d));

Arithmatic substract = (c,d) -> System.out.println("Subtract : " + (c-d));

Arithmatic multiply = (c,d) -> System.out.println("Multiply : " + (c\*d));

Arithmatic division = (c,d) -> System.out.println("Division : " + (c/d));

add.calculation(a, b);

substract.calculation(a, b);

multiply.calculation(a, b);

division.calculation(a, b);

}

}

interface Arithmatic{

public void calculation(int a,int b);

}

**2**

|  |
| --- |
| import java.util.Arrays; |
|  | import java.util.List; |
|  |  |
|  | public class OrderDemo { |
|  |  |
|  | public static void main(String[] args) { |
|  | List<Order> ordersList = Arrays.asList( |
|  | new Order("A", 11000, "ACCEPTED"), |
|  | new Order("B", 12000, "COMPLETED"), |
|  | new Order("C", 1000, "ACCEPTED"), |
|  | new Order("D", 2000, "COMPLETED") |
|  | ); |
|  |  |
|  | SortOrder orders = list ->{ for (Order order : list) { |
|  | if(order.oPrice > 10000) { |
|  | System.out.println(order); |
|  | } |
|  | }}; |
|  |  |
|  | orders.sortedList(ordersList); |
|  |  |
|  | } |
|  |  |
|  | } |
|  |  |
|  | interface SortOrder{ |
|  | public void sortedList(List<Order> l); |
|  | } |
|  |  |
|  | class Order { |
|  |  |
|  | String oName; |
|  | int oPrice; |
|  | String status; |
|  | public Order(String oName, int oPrice, String status) { |
|  | super(); |
|  | this.oName = oName; |
|  | this.oPrice = oPrice; |
|  | this.status = status; |
|  | } |
|  |  |
|  | @Override |
|  | public String toString() { |
|  | // TODO Auto-generated method stub |
|  | return "Order Name: " + oName + " Prics: " + oPrice + " Status: " + status ; |
|  | } |
|  |  |
|  | } |

**3**

import java.util.Arrays;

import java.util.function.Consumer;

import java.util.function.Function;

import java.util.function.Predicate;

import java.util.function.Supplier;

public class Lambda3 {

public static void main(String[] args) {

String[] str = {"Vinod", "Teju","Shiva"};

Supplier<String> supplier = ()-> Arrays.toString(str) ; //A Supplier interface has only one single method called get().

System.out.println(supplier.get()); //It does not accept any arguments and returns an object of any data type.

Consumer<String[]> consumer = (string) -> System.out.println(Arrays.toString(string)); //The Consumer interface has only one single method called accept().

consumer.accept(str); //It accepts a single argument of any data type and does not return any result.

Predicate<String[]> predicate = (string) -> Arrays.toString(string).contains("Ksheera"); //The Predicate interface has only one single method test().

System.out.println(predicate.test(str)); //It may be true or false depending on the values of its variables.

Function<String[], String> function = (string) -> Arrays.toString(string); //The Function interface has only one single method apply().

System.out.println(function.apply(str)); //It can accept an object of any data type and returns a result of any data.

}

}

**4**

import java.util.ArrayList;

import java.util.List;

public class Lambda4 {

public static void main(String[] args) {

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

str.add("cake");

str.add("sai");

str.add("Vinod");

str.add("Teju");

str.removeIf(l -> l.length()%2 !=0);

System.out.println(str);

}

}

**5**

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

import java.util.function.Function;

public class Lambda5 {

public static void main(String[] args) {

List<String> str = Arrays.asList("Vinod", "Teju","Shiva");

Function<List<String>,List<String>> function = (string) -> {

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

for (String s : string) {

stringList.add(""+s.charAt(0));

} return stringList;};

System.out.println(function.apply(str));

}

}

**6**

import java.util.Arrays;

import java.util.List;

import java.util.function.UnaryOperator;

public class Lambda6 {

public static void main(String[] args) {

List<String> str = Arrays.asList("Vinod", "Teju","Meghana");

UnaryOperator<String> unaryOperator = (list) -> list.toUpperCase();

str.replaceAll(l -> unaryOperator.apply(l));

System.out.println(str);

}

}

**7**

import java.util.HashMap;

import java.util.Map;

import java.util.Map.Entry;

import java.util.function.Function;

public class Lambda7 {

public static void main(String[] args) {

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

map.put(1, "Vinod");

map.put(2, "Teju");

Function<Map<Integer, String>, StringBuilder> function = mapValues -> {

StringBuilder sb = new StringBuilder();

for (Entry<Integer, String> string : mapValues.entrySet()) {

sb.append(string.getKey());

sb.append(string.getValue());

}

return sb;

};

System.out.println(function.apply(map));

}

}

**8**

import java.lang.reflect.Array;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Iterator;

import java.util.List;

import java.util.function.Consumer;

public class LambdaThread {

public static void main(String[] args) {

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

Consumer<List<Integer>>dispList = (list1) -> {

for(Integer integer : list1) {

System.out.print(integer + " ");

}

};

Thread newthread = new Thread( ()-> dispList.accept(list) );

newthread.start();

}

}