**1.Write an application to perform basic arithmetic operations like add, subtract, multiply & divide. You need to define a functional interface first.**

@FunctionalInterface

**interface** Arithmetic{

**int** operations(**int** a,**int** b);

}

**public** **class** lambda1 {

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

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

System.***out***.println(" "+ addition.operations(13,5));

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

System.***out***.println(" "+ subtraction.operations(13,5));

Arithmetic division = (a, b) -> (a + b);

System.***out***.println(" "+ division.operations(13,5));

Arithmetic mul = **new** Arithmetic() {

@Override

**public** **int** operations(**int** a, **int** b) {

**return** (a \* b);

}

};

System.***out***.println(" "+ mul.operations(13,5));

}

}

**2.Write an application using lambda expressions to print Orders having 2 criteria implemented: 1) order price more than 10000 2) order status is ACCEPTED or COMPLETED.**

**import** java.util.ArrayList;

**import** java.util.List;

**class** orderDetails{

**int** price;

String status;

**public** orderDetails(**int** price, String status) {

**super**();

**this**.price = price;

**this**.status = status;

}

**public** **int** getPrice() {

**return** price;

}

**public** String getStatus() {

**return** status;

}

**public** String toString() {

**return** "Price:"+price+" Status:"+status;

}

}

**public** **class** lambda2 {

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

orderDetails obj0 = **new** orderDetails(15000, "Accepted");

orderDetails obj1 = **new** orderDetails(5000, "Accepted");

orderDetails obj2 = **new** orderDetails(25000, "notAccepted");

orderDetails obj3 = **new** orderDetails(2000, "notAccepted");

orderDetails obj4 = **new** orderDetails(55000, "Accepted");

orderDetails obj5 = **new** orderDetails(105000, "Accepted");

List<orderDetails> ordrList = **new** ArrayList<>();

ordrList.add(obj0);

ordrList.add(obj1);

ordrList.add(obj2);

ordrList.add(obj3);

ordrList.add(obj4);

ordrList.add(obj5);

*printAll*(ordrList,condition->condition.getStatus().startsWith("A") && condition.getPrice()>10000);

}

**private** **static** **void** printAll(List<orderDetails> ordrList,Condition condition) {

// **TODO** Auto-generated method stub

**for**(orderDetails iDetails : ordrList)

**if**(condition.test(iDetails))

System.***out***.println(iDetails);

}

}

**interface** Condition{

**boolean** test(orderDetails iDetails);

}

**3.Use the functional interfaces Supplier, Consumer, Predicate & Function to invoke built-in methods from Java API.**

**Description:**

Write a program using the Java API’s mentioned in the question.

**import** java.util.ArrayList;

**import** java.util.HashMap;

**import** java.util.List;

**import** java.util.Map;

**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) {

orderDetails obj0 = **new** orderDetails(15000, "Accepted");

orderDetails obj1 = **new** orderDetails(5000, "Accepted");

orderDetails obj2 = **new** orderDetails(25000, "notAccepted");

orderDetails obj3 = **new** orderDetails(2000, "notAccepted");

orderDetails obj4 = **new** orderDetails(55000, "Accepted");

orderDetails obj5 = **new** orderDetails(105000, "Accepted");

List<orderDetails> ordrList = **new** ArrayList<>();

ordrList.add(obj0);

ordrList.add(obj1);

ordrList.add(obj2);

ordrList.add(obj3);

ordrList.add(obj4);

ordrList.add(obj5);

*printAll*(ordrList,condition->condition.getStatus().startsWith("A") && condition.getPrice()>10000,()->ordrList.get(3),x->x.getPrice(),p->System.***out***.println(p));

}

**private** **static** **void** printAll(List<orderDetails> ordrList,Predicate<orderDetails> predicate,Supplier<orderDetails> supplier,Function<orderDetails, Integer> function,Consumer<orderDetails> consumer) {

Map<Integer,String> list2 = **new** HashMap<>();

**for**(orderDetails iDetails : ordrList) {

list2.put(function.apply(iDetails),iDetails.getStatus());

**if**(predicate.test(iDetails)) {

consumer.accept(iDetails);

}

}

System.***out***.println(supplier.get());

System.***out***.println(list2);

}

}

**4.Remove the words that have odd lengths from the list. HINT: Use one of the new methods from JDK 8. Use removeIf() method from Collection interface.**

**Description:-**

Write a program using java 8 features which can remove the odd length words from the list.

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.function.Predicate;

**class** Words{

String word;

**public** Words(String word) {

**super**();

**this**.word = word;

}

**public** **void** setWord(String word) {

**this**.word = word;

}

**public** String getWord() {

**return** word;

}

**public** String toString() {

**return** word;

}

}

**public** **class** lambda4 {

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

Words word1 = **new** Words("Aparna");

Words word2 = **new** Words("Deeph");

Words word3 = **new** Words("Ram");

Words word4 = **new** Words("Raj");

List<Words> wordsList = **new** ArrayList<>();

wordsList.add(word1);

wordsList.add(word2);

wordsList.add(word3);

wordsList.add(word4);

*PrintWords*(wordsList,p->p.getWord().length()%2!=0);

}

**private** **static** **void** PrintWords(List<Words> wordsList,Predicate<Words> predicate) {

wordsList.removeIf(predicate);

**for**(Words words:wordsList) {

System.***out***.println(words);

}

}

}

**5.Create a string that consists of the first letter of each word in the list of Strings provided. HINT: Use Consumer interface & a String Builder to construct the result.**

**Description:**

Write a java program using StringBuilder and Consumer interface which will return a string. The returned string should consistes of the first let of each word in the list of words.

**import** java.util.Arrays;

**import** java.util.List;

**import** java.util.function.Consumer;

**public** **class** lambda5{

**static** List<String> *list* = Arrays.*asList*("alpha", "ben", "charry", "danny", "ester", "feroz");

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

*PrintString*(*list*,p->System.***out***.println(p));

}

**private** **static** **void** PrintString(List<String> list2,Consumer<StringBuilder> consumer) {

StringBuilder stringBuilder = **new** StringBuilder();

**for**(String s:*list*)

stringBuilder.append(s.charAt(0));

consumer.accept(stringBuilder);

}

}

**6.Replace every word in the list with its upper case equivalent. Use replaceAll() method & Unary Operator interface.**

Using replaceAll() method and Unary Operator interface write a java program which replaces evry word in the list with its upper case equivalent.

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.function.UnaryOperator;

**public** **class** lambda6 {

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

List<String> list = **new** ArrayList<>();

list.add("Java");

list.add("JavaScript");

list.add("Python");

list.add("Sql");

list.add("OpenNLP");

*printList*(list,p->p.toUpperCase());

}

**private** **static** **void** printList(List<String> list,UnaryOperator<String> unaryOperator) {

list.replaceAll(unaryOperator);

**for** (String i: list) {

unaryOperator.apply(i);

System.***out***.println(i);

}

}

}

**7.Convert every key-value pair of the map into a string and append them all into a single string, in iteration order. HINT: Use Map.entrySet() method & a StringBuilder to construct the result String.**

**Description:-**

Write a java program using Map.entrySet() method & a StringBuilder which will return a string by appending all the key value pairs of a map into a single string ,in insertion order.

**import** java.util.Map;

**import** java.util.TreeMap;

**import** java.util.function.Consumer;

**import** java.util.Map.Entry;

**class** Details **implements** Comparable<Details>{

**private** String name;

**private** String email;

**private** String gender;

**public** Details(String name, String email, String gender) {

**super**();

**this**.name = name;

**this**.email = email;

**this**.gender = gender;

}

**public** String getName() {

**return** name;

}

**public** String getEmail() {

**return** email;

}

**public** String getGender() {

**return** gender;

}

@Override

**public** String toString() {

**return** "Name:"+name+" Email:"+email+" Gender:"+gender;

}

@Override

**public** **int** compareTo(Details o) {

// **TODO** Auto-generated method stub

**return** **this**.getName().compareTo(o.getName());

}

}

**class** phoneNumber **implements** Comparable<phoneNumber>{

**private** **long** number;

**public** phoneNumber(**long** number) {

**super**();

**this**.number = number;

}

**public** **long** getNumber() {

**return** number;

}

**public** String toString() {

**return** String.*valueOf*(number);

}

@Override

**public** **int** compareTo(phoneNumber o) {

// **TODO** Auto-generated method stub

**return** **this**.toString().compareTo(o.toString());

}

}

**public** **class** lambda7 {

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

Details D1 = **new** Details("aparna", "A99@gmail.com", "Female");

Details D2 = **new** Details("deeph", "d00@gmail.com", "Female");

Details D3 = **new** Details("gourav", "gp@gmail.com", "Male");

Details D4 = **new** Details("cherry", "ch@gmail.com", "Female");

Details D5 = **new** Details("havi", "h00@gmail.com", "Female");

phoneNumber P1 = **new** phoneNumber(9505962633l);

phoneNumber P2 = **new** phoneNumber(9515962633l);

phoneNumber P3 = **new** phoneNumber(9525962633l);

phoneNumber P4 = **new** phoneNumber(9535962633l);

phoneNumber P5 = **new** phoneNumber(9545962633l);

Map<phoneNumber, Details> map = **new** TreeMap<>();

map.put(P1,D1);

map.put(P2,D2);

map.put(P3,D3);

map.put(P4,D4);

map.put(P5,D5);

*newStringBuilder*(map,p->System.***out***.println(p));

}

**private** **static** **void** newStringBuilder(Map<phoneNumber, Details> map,Consumer<StringBuilder> consumer) {

StringBuilder stringBuilder =**new** StringBuilder();

**for**(Entry<phoneNumber, Details> entry:map.entrySet()) {

stringBuilder.append(entry.getKey());

stringBuilder.append(entry.getValue());

}

consumer.accept(stringBuilder);

}

}

**8.Create a new thread that prints the numbers from the list. Use class Thread & interface Consumer.**

**Description:-**

Write a java program which will print the list of number using Thread and interface Consumer.

**import** java.util.Arrays;

**import** java.util.List;

**import** java.util.function.Consumer;

**public** **class** lambda8 {

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

List<Integer> list= Arrays.*asList*(1,2,3,4,5);

*Print*(list,p->System.***out***.println(p));

}

**private** **static** **void** Print(List<Integer> list,Consumer<Integer> consumer) {

Thread thread =**new** Thread(()-> {

**for** (Integer i:list)

consumer.accept(i);

}

);

thread.start();

}

}