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

Ans🡪

**package** org.lambda.java;

**interface** Arithmetic{

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

}

**public** **class** AllOperations {

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

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

System.***out***.println("Addition : "+ addition.operation(5,6));

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

System.***out***.println("Subtraction : "+ subtraction.operation(8, 1));

Arithmetic multiplication = (**int** a, **int** b) -> a\*b;

System.***out***.println("Multiplication : "+ multiplication.operation(8, 9));

Arithmetic divide = (**int** a, **int** b) -> a/b;

System.***out***.println("Divide : "+ divide.operation(16, 8));

}

}

Output:-

Addition : 11

Subtraction : 7

Multiplication : 72

Divide : 2

1. 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.

Ans🡪

**package** org.lambda.java;

**interface** Order{

**int** foo(**int** a);

}

**public** **class** Print {

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

Order mylambda = (**int** a) ->{

**if**(a>10000){

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

}

**else**

{

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

}

**return** a;

};

System.***out***.println("Order amount:" +(mylambda.foo(5000000)));

}

}

Output:

Accepted

Order amount:5000000

1. Use the functional interfaces Supplier, Consumer, Predicate, & Function to invoke built in methods from java API

Ans🡪

**package** org.lambda.java;

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

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

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

**public** **class** Interface {

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

Predicate<Integer> pt= a-> (a>10);

System.***out***.println("Predicate :" +pt.test(20));

String str="Hello";

Supplier<Integer> su=()-> str.length();

System.***out***.println("Supplier: "+ su.get());

Consumer <String > print = a-> System.***out***.println("Consumer: "+a);

print.accept("How are you ");

}

}

Output:

Predicate :**true**

Supplier: 5

Consumer: How are you

1. Remove the words that have odd lengths from the list. Hint: Use one of the method from jdk 8. Use removef() method from collection interface.

Ans🡪

**package** org.lambda.java;

**import** java.util.ArrayList;

**public** **class** OddLength {

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

ArrayList<String> words = **new** ArrayList<String>();

words.add("Hii");

words.add("Chaitali");

words.add("Chaudhari");

words.add("Whatss");

words.add("up");

words.removeIf(n->(n.length()%2!=0));

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

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

}

}

}

Output:

Chaitali

Whatss

up

1. Create a string that consist of the first letter of each word in the list of Stings provided, hint: Use Consumer interface & s StringBuilder to construct the result.

Ans🡪

**package** org.lambda.java;

**import** java.util.ArrayList;

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

**public** **class** FirstLetter {

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

ArrayList<String> words= **new** ArrayList<String>();

words.add("Hii");

words.add("Hello");

words.add("Chaitali");

words.add("Chaudhari");

words.add("Whatss");

words.add("Up");

Consumer <String> print= (str) -> System.***out***.println("The first letter of strings: " +str.charAt(0));

words.forEach(print);

}

}

Output:

The first letter of strings: H

The first letter of strings: H

The first letter of strings: C

The first letter of strings: C

The first letter of strings: W

The first letter of strings: U

1. Replace every word in the list with its upper case equivalent. Use replaceAll() method & UnaryOperator interface.

Ans🡪

**package** org.lambda.java;

**import** java.util.ArrayList;

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

**class** Op **implements** UnaryOperator<String> {

**public** String apply(String str) {

**return** str.toUpperCase();

}

}

**public** **class** Test {

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

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

list.add("Java");

list.add("JavaScript");

list.add("Array");

list.add("JUnit");

list.add("Java8");

System.***out***.println("Contents of the list: "+list);

list.replaceAll(**new** Op());

System.***out***.println("Contents of the list after replace operation: \n"+list);

}

}

Output:

Contents of the list: [Java, JavaScript, Array, JUnit, Java8]

Contents of the list after replace operation:

[JAVA, JAVASCRIPT, ARRAY, JUNIT, JAVA8]

1. Convert every key value pair of the map info a string and append them all into a single string, in iterartion order, Hint: Use Map.entrySet() method & a StringBuilder to construct the result String.

Ans🡪

**package** org.lambda.java;

**import** java.util.HashMap;

**import** java.util.Map;

**import** java.util.stream.Collectors;

**public** **class** Example7 {

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

Map<String, String> map= **new** HashMap<>(5);

map.put("Janu", "9");

map.put("Aadesh", "3");

map.put("Chaitu", "7");

map.put("Priya", "4");

String s= map.entrySet().stream().map((entry) -> " "+ entry.getKey()+

"\""+ entry.getValue().replaceAll("\"", "\\\\\"") + "\"").collect(Collectors.*joining*(" "));

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

}

}

Output:

Aadesh "3" Janu "9" Priya "4" Chaitu "7"

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

Ans🡪

**package** org.lambda.java;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** Exxmple8 {

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

List<Integer> n =**new** ArrayList<Integer>(){

{

add(12);

add(59);

add(45);

add(85);

}

};

Thread mylambda = **new** Thread(() -> System.***out***.println(n));

mylambda.run();

}

}

Output:

[12, 59, 45, 85]