**Lambda Expression Assignments**

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

**interface** Arithmetic

{

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

}

**public** **class** arithmeticClass

{

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

{

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

System.***out***.println("Addition = " + addition.operation(10, 20));

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

System.***out***.println("Subtraction = " + subtraction.operation(20, 10));

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

System.***out***.println("Multiplication = " + multiplication.operation(10, 20));

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

System.***out***.println("Division = " + division.operation(120, 10));

}

}

**OUTPUT:**

Addition = 30

Subtraction = 10

Multiplication = 200

Division = 12

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.Scanner;

**interface** lamb

{

**public** **void** show();

}

**public** **class** example

{

**public** **static** **void** main(String a[])

{

System.***out***.print("Enter the order price:");

Scanner sc=**new** Scanner(System.***in***);

**int** order=sc.nextInt();

System.***out***.println("Entered order is: "+order);

lamb lb = ()-> {

**if**(order>10000)

{

System.***out***.println("Order is more than 10000");

}

**else**

{

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

}

};

lb.show();

}

}

**OUTPUT:**

Enter the order price:1000

Entered order is: 1000

Order is Accepted

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

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

**import** java.util.function.Function;

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

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

**class** Product

{

**private** **double** price = 0.0;

**public** **void** setPrice(**double** price)

{

**this**.price = price;

}

**public** **void** printPrice() {

System.***out***.println("This is Consumer functional interface: "+price);

}

}

**public** **class** third

{

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

{

//consumer functional interface

Consumer<Product> updatePrice = p -> p.setPrice(5.9);

Product p = **new** Product();

updatePrice.accept(p);

p.printPrice();

//Predicate functional interface

Predicate<String> isALongWord = t -> t.length() > 10;

String s = "successfully";

**boolean** result = isALongWord.test(s);

System.***out***.println("This is predicate functional interface: "+s);

//Function functional interface

Function<Integer, Double> half = a -> a / 2.0;

System.***out***.println("This is Function functional interface: "+half.apply(10));

//Supplier functional interface

Supplier<Double> randomValue = () -> Math.*random*();

System.***out***.println("This is supplier functional interface: "+randomValue.get());

}

}

**OUTPUT:**

This is Consumer functional interface: 5.9

This is predicate functional interface: successfully

This is Function functional interface: 5.0

This is supplier functional interface: 0.4246407250483417

4. Remove the words that have odd lengths from the list. HINT: Use one of the

new methods from JDK 8. Use removelf() method from Collection interface.

**import** java.util.\*;

**public** **class** removeOdd

{

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

{

ArrayList<Integer> num = **new** ArrayList<Integer>();

num.add(1);

num.add(2);

num.add(3);

num.add(4);

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

**for** (**int** i : num)

{

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

}

}

}

**OUTPUT:**

2

4

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

**import** java.util.Arrays;

**import** java.util.List;

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

**public** **class** fifth

{

**public** **static** **void** main(String a[])

{

List<String> strings = Arrays.*asList*("Humble", "Elegant", "Lovely", "Lily", "Output");

Consumer<String> consumerString=s->System.***out***.println(s);

strings.stream()

.map(s -> s.substring(0, 1))

.forEach(s -> System.***out***.println(s));

}

}

**OUTPUT:**

H

E

L

L

O

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

**import** java.util.ArrayList;

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

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

{

**public** String apply(String str)

{

**return** str.toUpperCase();

}

}

**public** **class** replaceMethod

{

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

{

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

list.add("java");

list.add("is ");

list.add("my");

list.add("favourite");

list.add("subject");

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, is , my, favourite, subject]

Contents of the list after replace operation:

[JAVA, IS , MY, FAVOURITE, SUBJECT]

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.

**import** java.util.HashMap;

**import** java.util.Map;

**import** java.util.TreeMap;

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

**public** **class** mapKey {

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

Map<String,String> m1=**new** TreeMap<String,String>();

m1.put("AJAY", "PRADHAN");

m1.put("VIJAY","SHEKHAWAT");

m1.put("SHIVAKAR","CHATURVEDI");

m1.put("NILESH","TIWARI");

m1.put("NILESH","SHARMA");

String s = m1.entrySet().stream().map((entry) -> entry.getKey() + " " + entry.getValue()+ " ")

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

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

}

}

**OUTPUT:**

AJAY PRADHAN NILESH SHARMA SHIVAKAR CHATURVEDI VIJAY SHEKHAWAT

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

**import** java.util.Arrays;

**import** java.util.List;

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

**public** **class** eight

{

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

{

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

//Thread Class

**new** Thread(() ->

{

**for**(**int** i=1; i <= 5; i++)

{

System.***out***.println("Thread: "+ i);

}

}).start();

//Consumer interface

Consumer<Integer> consumer = (Integer x) -> System.***out***.println(x);

*forEach*(list, consumer);

}

**static** <T> **void** forEach(List<T> list, Consumer<T> consumer)

{

**for** (T t : list)

{

consumer.accept(t);

}

}

}

**OUTPUT:**

Thread: 1

Thread: 2

Thread: 3

Thread: 4

Thread: 5

1

2

3

4

5