**LAMBDA Expression Assignment**

**1.Write an application to perform basic arithmetic operations like add, sub, multiply and divide.**

**package** Lambda;

**interface** Arithmetic

{

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

}

**public** **class** Arithoperation

{

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

{

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

System.***out***.println("Addition ->"+addition.operations(10,20));

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

System.***out***.println("Subtraction ->"+sub.operations(100,65));

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

System.***out***.println("Multiplication ->"+multiply.operations(30,2));

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

System.***out***.println("Division ->"+division.operations(10,5));

}

}

Output:

Addition ->30

Subtraction ->35

Multiplication ->60

Division ->2

**2. Write an Application using lambda expression to print orders having 2 criteria: 1) order price more than 10000 2) order status is accepted or completed.**

**package** Lambda;

**import** java.util.Scanner;

**public** **class** Order

{

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

{

order Lambda=(**int** x)->

{

**if**(x>10000)

{

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

}

**else**

{

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

}

**return** x;

};

System.***out***.println("Order amount:"+(Lambda.print(10)));

}

**interface** order

{

**int** print(**int** x);

}

}

Output:

Rejected

Order amount:10

**3. Create a string that consists of the first letter of each word in the list provided.**

**package** Lambda;

**import** java.util.ArrayList;

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

**public** **class** string

{

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

{

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

words.add("Hi");

words.add("Pretty");

words.add("Girl");

words.add("How");

words.add("You");

words.add("Doin?");

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: P

The first letter of Strings: G

The first letter of Strings: H

The first letter of Strings: Y

The first letter of Strings: D

**4. Remove the words that have odd length from the List.**

**package** Lambda;

**import** java.util.ArrayList;

**public** **class** Odd

{

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

{

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

length.add("Market");

length.add("Mall");

length.add("Fun");

length.add("Suprise");

length.add("Come");

length.add("Cube");

length.add("Super");

length.add("Warmth");

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

**for**(String i:length)

{

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

}

}

}

Output:

Market

Mall

Come

Cube

Warmth

**5. Use the functional interfaces supplier,consumer and predicate**

**package** Lambda;

**import** java.util.ArrayList;

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

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

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

**public** **class** Scp

{

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

{

//predicate

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

System.***out***.println("predicate:" + gt.test(20));

//supplier

String str="HI";

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

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

//Consumer

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

print.accept("HELLO!");

}

}

Output:

predicate:true

Supplier:2

Consumer:HELLO!

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

package lam;  
import java.util.ArrayList;  
public class exp6 {  
public static void main(String[] args)  
{  
ArrayList<String>words=new ArrayList<String>();  
words.add("hi");  
words.add("Hey");  
words.add("Hello");  
words.add("good");  
words.add("bad");  
words.add("three");  
words.replaceAll(s->s.toUpperCase());  
for(String i:words)  
{  
System.out.println("print:" +i);  
}  
}  
}

Output:

print:HI  
print:HEY  
print:HELLO  
print:GOOD  
print:BAD  
print:THREE

**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 String Builder to construct the result String.**

**package** Lambda;

**import** java.util.HashMap;

**import** java.util.Map;

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

**public** **class** maptostring {

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

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

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

map.put("J.K.", "Rowling");

map.put("Mark", "Twain");

map.put("Charles", "Dickens");

map.put("Lewis", "Carroll");

map.put("William", "Shakespeare");

String s = map.entrySet().stream().map((entry) ->" " + entry.getKey() + " " + entry.getValue().replaceAll("\\s ", "\\s ") + " ").collect(Collectors.*joining*(" "));

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

}

}

Output:

Mark Twain Charles Dickens William Shakespeare J.K. Rowling Lewis Carroll

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

**package** Lambda;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** ThreadList {

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

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

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

add(11);

add(55);

add(37);

add(95);

add(99);

}

};

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

mylambda.run();

}

}

Output:

[11, 55, 37, 95, 99]