1. **public** **interface** Arithmetic {

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

}

**Public** **class** lambda {

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

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

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

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

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

System.***out***.println("Addition of the number:"

+add.operations(3, 6)); System.***out***.println("Substraction of the number:"+

sub.operations(7, 3));

System.***out***.println("Multiplication of the number:"+

multiply.operations(6, 9));

System.***out***.println("Division of the number:"+

division.operations(2, 8));

}

}

Output

Addition of the number:9

Substraction of the number:4

Multiplication of the number:54

Division of the number:0

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

**import** java.util.stream.Stream;

**class** orders

{

**float** price;

String ordername;

String orderaddress;

String orderstatus;

**public** orders(**float** price,String ordername, String

orderaddress,String orderstatus)

{

**this**.price=price;

**this**.ordername=ordername;

**this**.orderaddress=orderaddress;

**this**.orderstatus=orderstatus;

}

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

{

ArrayList<orders> l1=**new** ArrayList<orders>();

l1.add(**new**

orders(36000f,"furniture","banglore","ACCEPTED"));

l1.add(**new**

orders(45000f,"Utensils","Mumbai","COMPLETED"));

l1.add(**new**

orders(2000f,"grocery","TamilNadu","ACCEPTED"));

l1.add(**new** orders(2000f,"Tv","Kerala","COMPLETED"));

Stream<orders> filtereddata=l1.stream().filter(o-

>o.price>10000);

filtereddata.forEach

(

orders ->{

System.***out***.println(orders.ordername+":"+orders.price);

System.***out***.println(orders.orderstatus);

}

);

}}

3.

**public** **class** Person {

**private** String name;

**private** **int** id;

**private** **int** age;

**public** Person(String name, **int** id, **int** age) {

**this**.name = name;

**this**.id = id;

**this**.age = age;

}

/\*\*

\* **@return** the name

\*/

**public** String getName() {

**return** name;

}

/\*\*

\* **@param** name the name to set

\*/

**public** **void** setName(String name) {

**this**.name = name;

}

/\*\*

\* **@return** the id

\*/

**public** **int** getId() {

**return** id;

}

/\*\*

\* **@param** id the id to set

\*/

**public** **void** setId(**int** id) {

**this**.id = id;

}

/\*\*

\* **@return** the age

\*/

**public** **int** getAge() {

**return** age;

}

/\*\*

\* **@param** age the age to set

\*/

**public** **void** setAge(**int** age) {

**this**.age = age;

}

@Override

**public** String toString() {

**return** "Person [name=" + name + ", id=" + id + ", age=" + age + "]";

}

}

**import** java.util.Arrays;

**import** java.util.List;

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

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

**import** java.util.Collections;

**public** **class** java1 {

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

List<Person> list=Arrays.*asList*(

**new** Person("nam",23,21),

**new** Person("var",12,23),

**new** Person("so",18,22)

);

//predicate interface

System.***out***.println("print all person");

*printConditionally*(list,p->**true**);

//consumer interface

System.***out***.println("print all person");

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

}

//predicate interface

**private** **static** **void** printConditionally(List<Person> list, Predicate<Person> predicate ) {

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

**for**(Person p:list) {

**if**(predicate.test(p))// Predicate interface

{

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

}

}

}

//consumer interface

**private** **static** **void** performConditionally(List<Person> list,

Predicate<Person> predicate,Consumer<Person>consumer ) {

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

**for**(Person p:list) {

**if**(predicate.test(p))// Predicate interface

{

consumer.accept(p);

}}}}

4. **import** java.util.ArrayList;

**public** **class** java

{

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

{

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

string.add("hi");

string.add("hello");

string.add("good");

string.add("evening");

string.add("bad");

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

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

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

}

}

}

Output:

hi

good

5. **import** java.util.ArrayList;

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

**public** **class** order {

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

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

list.add("hi");

list.add("hello");

list.add("good");

list.add("evening");

Consumer<String> print=(str)->System.out.println("first letter of

strings:"+str.charAt(0));

list.forEach(print);

}

}

6. **import** java.util.ArrayList;

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

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

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

**return** str.toUpperCase();

}

}

**public** **class** nam {

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

CloneNotSupportedException {

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

list.add("hi");

list.add("hello");

list.add("good");

list.add("evening");

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: [hi, hello, good, evening]

Contents of the list after replace operation:

[HI, HELLO, GOOD, EVENING]

8. **import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.List;

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

**public** **class** Thread {

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

Consumer<Integer> consumer=(i)-> System.***out***.println(" "+i);

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

list.add(3);

list.add(5);

list.add(7);

list.add(9);

list.add(2);

*printList*(list,consumer);

}

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

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

**for**(Integer integer:list) {

consumer.accept(integer);

}}}

Output:

3

5

7

9

2