JAVA 8
Consumer

Consumer (I)

Sometimes our requirment is we have to provide some input value, perform certain operation, but not required to return anything,then we should go for Consumer.i.e Consumer can be used to consume object and perform certain operation.

Consumer Functional Interface contains one abstract method accept.

```
1) interface Consumer<T>
2) {
3)    public void accept(T t);
4)  }
```

Demo Program-1 for Consumer:

```
1) import java.util.function.Consumer;
2) class Test
3)  {
4)     public static void main(String[] args)
5)     {
6)        Consumer<String> c=s->System.out.println(s);
7)        c.accept("Hello");
8)        c.accept("DURGASOFT");
9)     }
10) }
```

Output:

```
Hello
DURGASOFT
```

Demo Program-2 to display Movie Information by using Consumer:

```
1) import java.util.function.*;
2) import java.util.*;
3) class Movie
4) {
5)     String name;
6)     String hero;
7)     String heroine;
8)     Movie(String name,String hero,String heroine)
```

```
9)      {
10)        this.name=name;
11)        this.hero=hero;
12)        this.heroine=heroine;
13)     }
14) }
15) class Test
16) {
17)     public static void main(String[] args)
18)     {
19)        ArrayList<Movie> l= new ArrayList<Movie>();
20)        populate(l);
21)        Consumer<Movie> c= m->{
22)           System.out.println("Movie Name:"+m.name);
23)           System.out.println("Movie Hero:"+m.hero);
24)           System.out.println("Movie Heroine:"+m.heroine);
25)           System.out.println();
26)        };
27)        for(Movie m : l)
28)        {
29)           c.accept(m);
30)        }
31)
32)     }
33)     public static void populate(ArrayList<Movie> l)
34)     {
35)        l.add(new Movie("Bahubali","Prabhas","Anushka"));
36)        l.add(new Movie("Rayees","Sharukh","Sunny"));
37)        l.add(new Movie("Dangal","Ameer","Ritu"));
38)        l.add(new Movie("Sultan","Salman","Anushka"));
39)     }
40)
41) }
```

```
D:\durgaclasses>java Tes
Movie Name:Bahubali
Movie Hero:Prabhas
Movie Heroine:Anushka

Movie   Name:Rayees
Movie   Hero:Sharukh
Movie Heroine:Sunny

Movie Name:Dangal
Movie Hero:Ameer
Movie Heroine:Ritu
```

Movie Name:Sultan
Movie Hero:Salman
Movie Heroine:Anushka

Demo Program-3 for Predicate, Function & Consumer:

```
1)  import java.util.function.*;
2)  import java.util.*;
3)  class Student
4)  {
5)      String name;
6)      int marks;
7)      Student(String name,int marks)
8)      {
9)          this.name=name;
10)         this.marks=marks;
11)     }
12) }
13) class Test
14) {
15)     public static void main(String[] args)
16)     {
17)         ArrayList<Student> l= new ArrayList<Student>();
18)         populate(l);
19)         Predicate<Student> p= s->s.marks>=60;
20)         Function<Student,String> f=s->{
21)             int marks=s.marks;
22)             if(marks>=80)
23)             {
24)                 return "A[Dictinction]";
25)             }
26)             else if(marks>=60)
27)             {
28)                 return "B[First Class]";
29)             }
30)             else if(marks>=50)
31)             {
32)                 return "C[Second Class]";
33)             }
34)             else if(marks>=35)
35)             {
36)                 return "D[Third Class]";
37)             }
38)             else
39)             {
40)                 return "E[Failed]";
41)             }
```

```
42)         };
43)         Consumer<Student> c=s->{
44)            System.out.println("Student Name:"+s.name);
45)            System.out.println("Student Marks:"+s.marks);
46)            System.out.println("Student Grade:"+f.apply(s));
47)            System.out.println();
48)         };
49)         for(Student s : l)
50)         {
51)            if(p.test(s))
52)            {
53)               c.accept(s);
54)            }
55)         }
56)      }
57)      public static void populate(ArrayList<Student> l)
58)      {
59)         l.add(new Student("Sunny",100));
60)         l.add(new Student("Bunny",65));
61)         l.add(new Student("Chinny",55));
62)         l.add(new Student("Vinny",45));
63)         l.add(new Student("Pinny",25));
64)      }
65) }
```

Output:

```
Student Name:Sunny
Student Marks:100
Student Grade:A[Dictinction]

Student Name:Bunny
Student Marks:65
Student Grade:B[First Class]
```

Consumer Chaining:

Just like Predicate Chaining and Function Chaining, Consumer Chaining is also possible. For this Consumer Functional Interface contains default method andThen().

c1.andThen(c2).andThen(c3).accept(s)

First Consumer c1 will be applied followed by c2 and c3.

Demo Program for Consumer Chaining:

```
1)  import java.util.function.*;
2)  class Movie
3) {
4)      String name;
5)      String result;
6)      Movie(String name,String result)
7)      {
8)          this.name=name;
9)          this.result=result;
10)     }
11) }
12) class Test
13) {
14)     public static void main(String[] args)
15)     {
16)         Consumer<Movie> c1=m-
    >System.out.println("Movie:"+m.name+" is ready to release");
17)
18)         Consumer<Movie> c2=m-
    >System.out.println("Movie:"+m.name+" is just Released and it is:"+m.result);
19)
20)         Consumer<Movie> c3=m-
    >System.out.println("Movie:"+m.name+" information storing in the database");
21)
22)         Consumer<Movie> chainedC = c1.andThen(c2).andThen(c3);
23)
24)         Movie m1= new Movie("Bahubali","Hit");
25)         chainedC.accept(m1);
26)
27)         Movie m2= new Movie("Spider","Flop");
28)         chainedC.accept(m2);
29)     }
30) }
```

Output:

Movie:Bahubali is ready to release
Movie:Bahubali is just Released and it is:Hit
Movie:Bahubali information storing in the database
Movie:Spider is ready to release
Movie:Spider is just Released and it is:Flop
Movie:Spider information storing in the database