**Java-8 – BiConsumer**

* **BiConsumer:**

interface BiConsumer<T, U>{

public void accept<T t, U u);

default method: andThen();

}

* **Example:**

import java.util.function.BiConsumer;

class BiConsumerDemo {

public static void main(String[] args){

BiConsumer<String, String> c = (s1, s2) -> System.out.println(s1 + s2);

c.accept(“Saravana”, “Pandiyan”);

}

}

* **Program to increment employee salary using BiFunction & BiConsumer:**

class Employee {

String name;

double salary;

Employee(String name, double salary){

this.name = name;

this.salary = salary;

}

}

import java.util.List;

import java.util.ArrayList;

import java.util.function.BiFunction;

import java.util.function.BiConsumer;

class Test{

public static void main(String[] args){

List<Employee> l = new ArrayList<>();

BiFunction<String, Double, Employee> f = (name, salary) -> new Employee(name, salary);

l.add(f.apply(“Saravana”, 1000));

l.add(f.apply(“Rithika”, 2000));

l.add(f.apply(“Gokul”, 3000));

l.add(f.apply(“Devi”, 4000));

l.add(f.apply(“Deva”, 5000));

BiConsumer<Employee, Double> c = (e, d) -> e.salary = e.salary + d;

for(Employee e : l){

c.accept(2, 500);

}

for(Employee e: l){

System.out.println(“Employee Name:”+e.name);

System.out.println(“Employee Salary:”+e.salary);

System.out.println();

}

}

}

* **Comparison between one and two argument functional interfaces:**

|  |  |
| --- | --- |
| One-argument functional interface | Two-argument functional interface |
| interface Predicate<T> {  public boolean test(T t);  default Predicate and(Predicate p)  default Predicate or(Predicate p)  default Predicate negate(Predicate p)  static Predicate isEqual(Object o)  } | interface BiPredicate<T, U> {  public boolean test(T t, U u);  default BiPredicate and(BiPredicate p);  default BiPredicate or(BiPredicate p);  default BiPredicate negate(BiPredicate p);  } |
| interface Function<T, R> {  public R apply(T t);  default Function andThen(Function f)  default Function compose(Function f);  static Function identity();  } | interface BiFunction<T, U, R>{  public R apply(T t, U u);  default BiFunction andThen(Function f);  } |
| interface Consumer<T> {  public void accept(T t);  default Consumer andThen(Consumer c);  } | interface BiConsumer<T, U> {  public void accept(T t, U u);  default BiConsumer andThen(BiConsumer c);  } |

* **This document contains contents of 63 to 65.**