**JAVA 8 FEATURES ASSIGNMENT**

Ques 1:Write the following a functional interface and implement it using lambda:

* (1) First number is greater than second number or not Parameter (int ,int ) Return boolean
* (2) Increment the number by 1 and return incremented value Parameter (int) Return int
* (3) Concatination of 2 string Parameter (String , String ) Return (String)
* (4) Convert a string to uppercase and return . Parameter (String) Return (String)

Ans: **JAVA-CODE**

@FunctionalInterface

interface GreaterNumber{

boolean checkgreater(int a, int b);

}

@FunctionalInterface

interface IncreasedValue{

int increase(int a);

}

@FunctionalInterface

interface ConcatenateString{

String concatenate(String str1, String str2);

}

@FunctionalInterface

interface StringToUpperCase{

String uppercase(String str);

}

public class FunctionalInterfaceAndLambda{

public static void main(String[] args) {

GreaterNumber greaterNumber = (a,b) ->{

if (a >b){

return true;

}

else{

return false;

}

};

System.out.println(greaterNumber.checkgreater(6,5));

IncreasedValue increasedValue = (a) -> {

a= a+1;

return a;

};

System.out.println(increasedValue.increase(78));

ConcatenateString concatenateString = (str1, str2) -> {

String str =str1+str2;

return str;

};

System.out.println(concatenateString.concatenate("Harsh ","Yadav"));

StringToUpperCase stringToUpperCase = (str) ->{

return (str.toLowerCase());

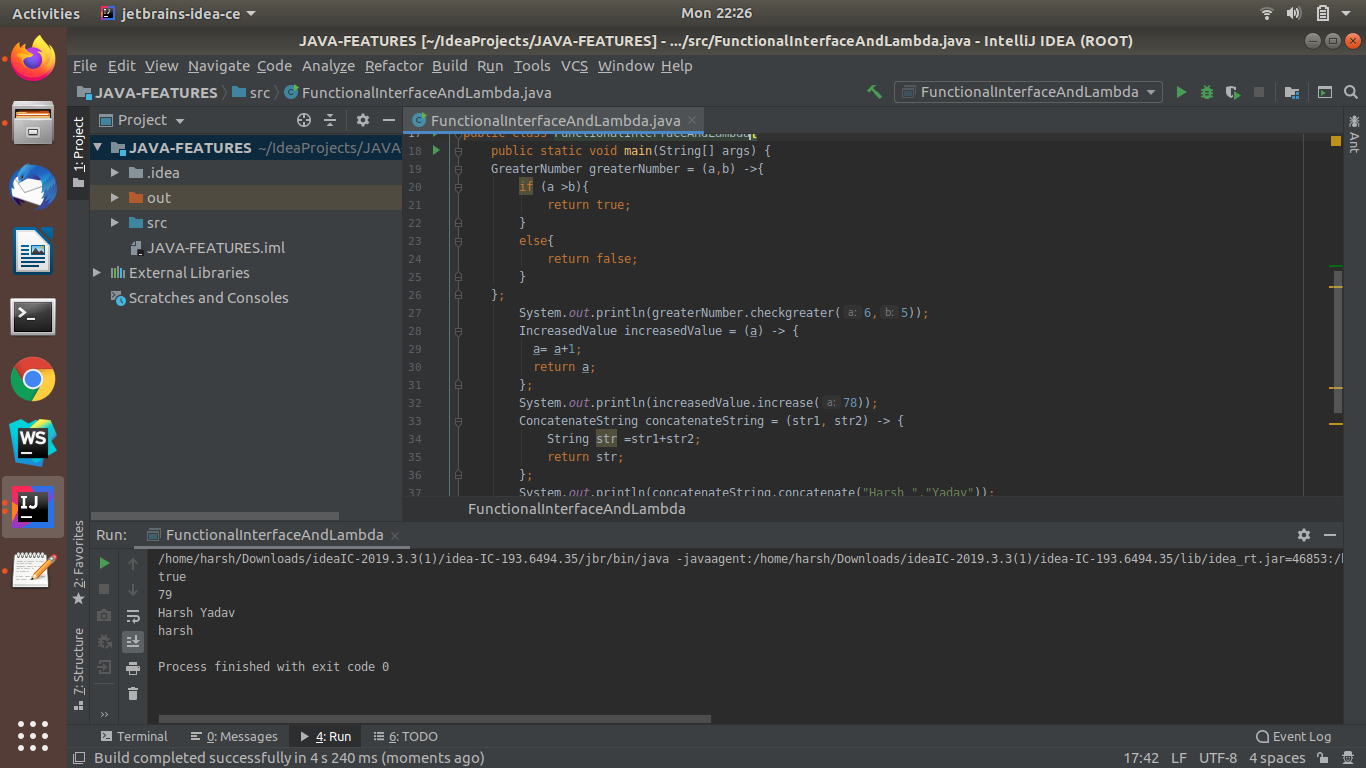
};

System.out.println(stringToUpperCase.uppercase("harsh"));

}

}

Output:



Ques 2:Create a functional interface whose method takes 2 integers and return one integer.

Ans: **JAVA-CODE**

@FunctionalInterface

interface TwoToOne{

int ReturnOne(int a, int b);

}

public class TakesTwoIntegerAndReturnOne {

public static void main(String[] args) {

TwoToOne twoToOne = (a,b) ->{

return a;

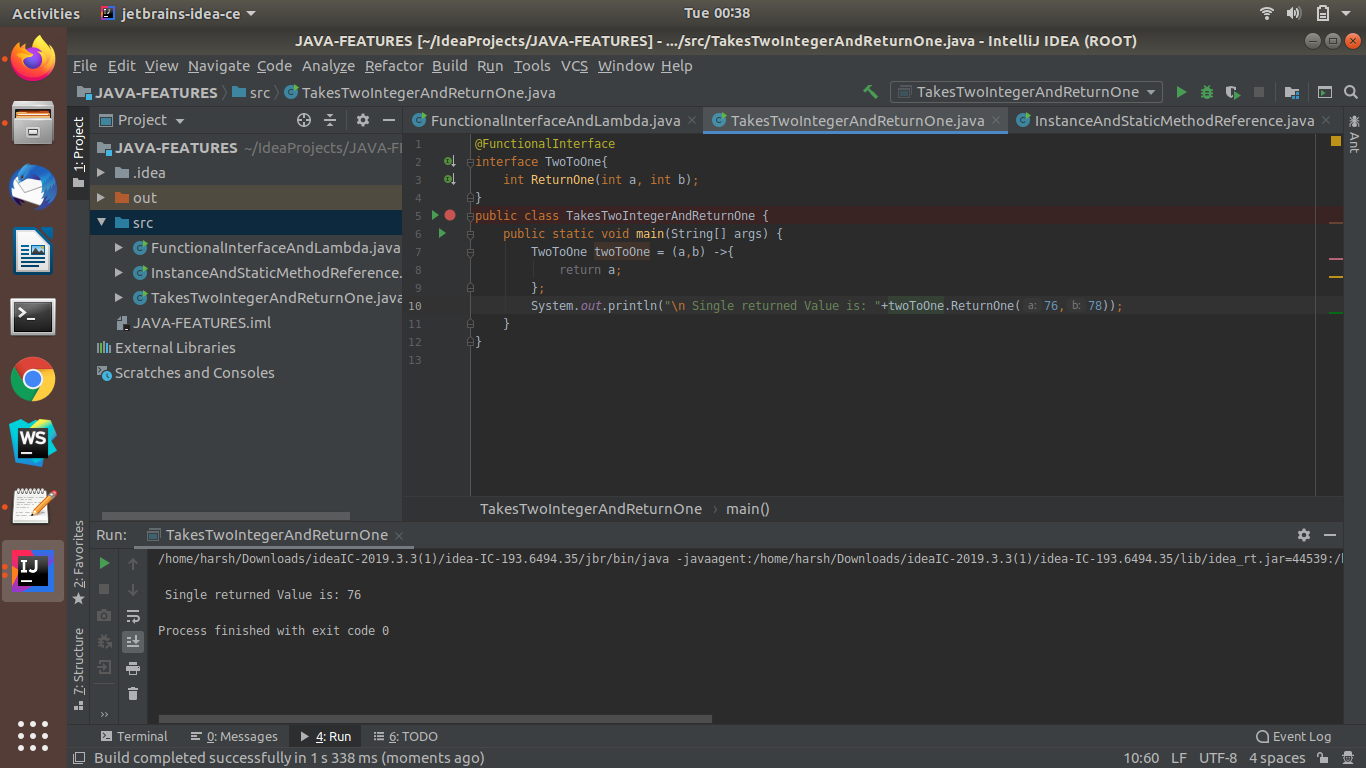
};

System.out.println("\n Single returned Value is: "+twoToOne.ReturnOne(76,78));

}

}

OUTPUT:



Ques 3:Using (instance) Method reference create and apply add and subtract method and using (Static) Method reference create and apply multiplication method for the functional interface created.

Ans: **JAVA-CODE**

@FunctionalInterface

interface Add{

int add (int a, int b);

}

@FunctionalInterface

interface Subtract{

int sub (int a, int b);

}

@FunctionalInterface

interface Multiplication{

int mul (int a, int b);

}

public class InstanceAndStaticMethodReference {

public int add1 (int a, int b){

return a+b;

}

public int sub1 (int a, int b){

return a - b;

}

public static int multiplication (int a, int b){

return a \* b;

}

public static void main(String[] args) {

Add add = new InstanceAndStaticMethodReference()::add1;

Subtract subtract = new InstanceAndStaticMethodReference()::sub1;

Multiplication multiplication = InstanceAndStaticMethodReference::multiplication;

System.out.println("Addition of Two Numbers is: "+add.add(24,23));

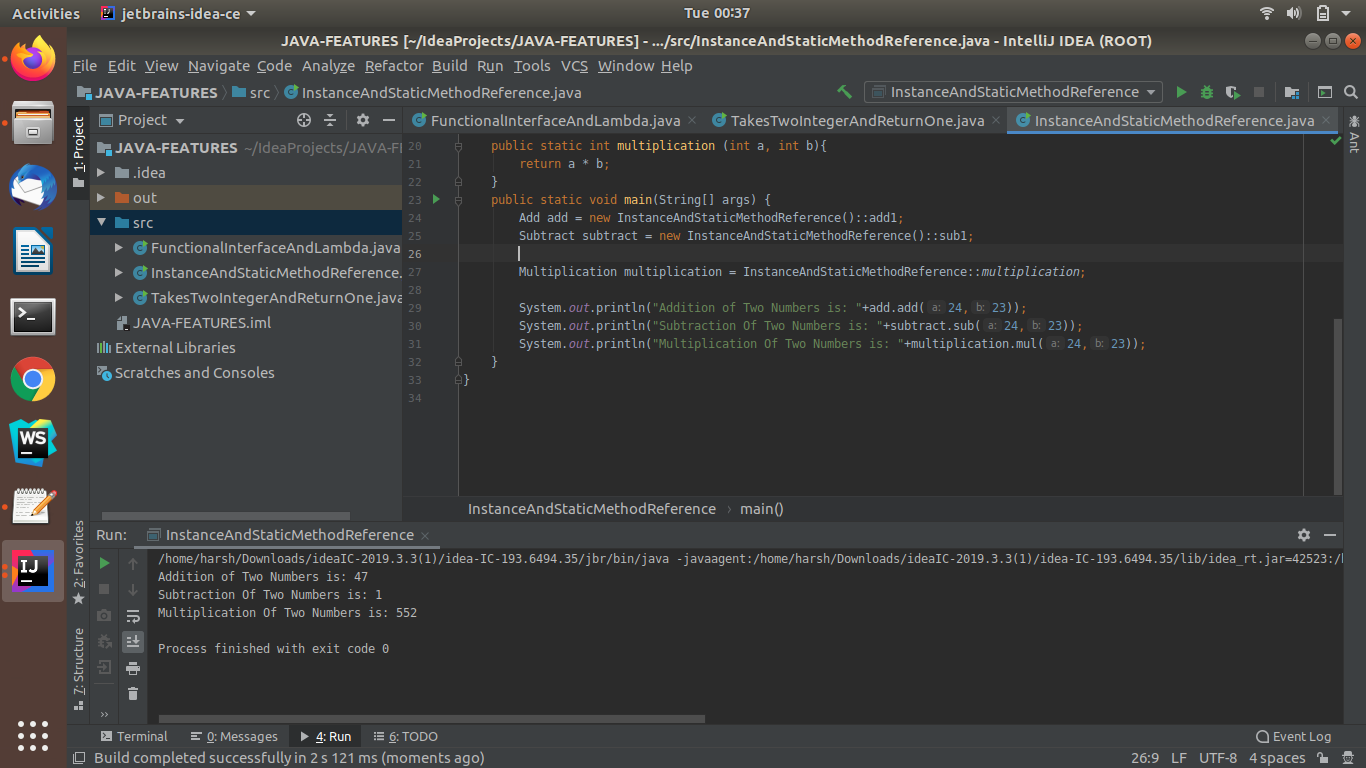
System.out.println("Subtraction Of Two Numbers is: "+subtract.sub(24,23));

System.out.println("Multiplication Of Two Numbers is: "+multiplication.mul(24,23));

}

}

OUTPUT:



Ques 4: Create an Employee Class with instance variables (String) name, (Integer)age, (String)city and get the instance of the Class using constructor reference

Ans: **JAVA-CODE**

@FunctionalInterface

interface Employee1{

Employee setdetails(String name, int age, String city);

}

class Employee{

String name;

int age;

String city;

Employee(String name1, int age1, String city1){

this.name = name1;

this.age = age1;

this.city = city1;

System.out.println("\nName of Employee: "+name);

System.out.println("Age of Employee: "+age);

System.out.println("City of Employee: "+city);

}

}

public class ConstructorReference {

public static void main(String[] args) {

Employee1 employee1 = Employee::new;

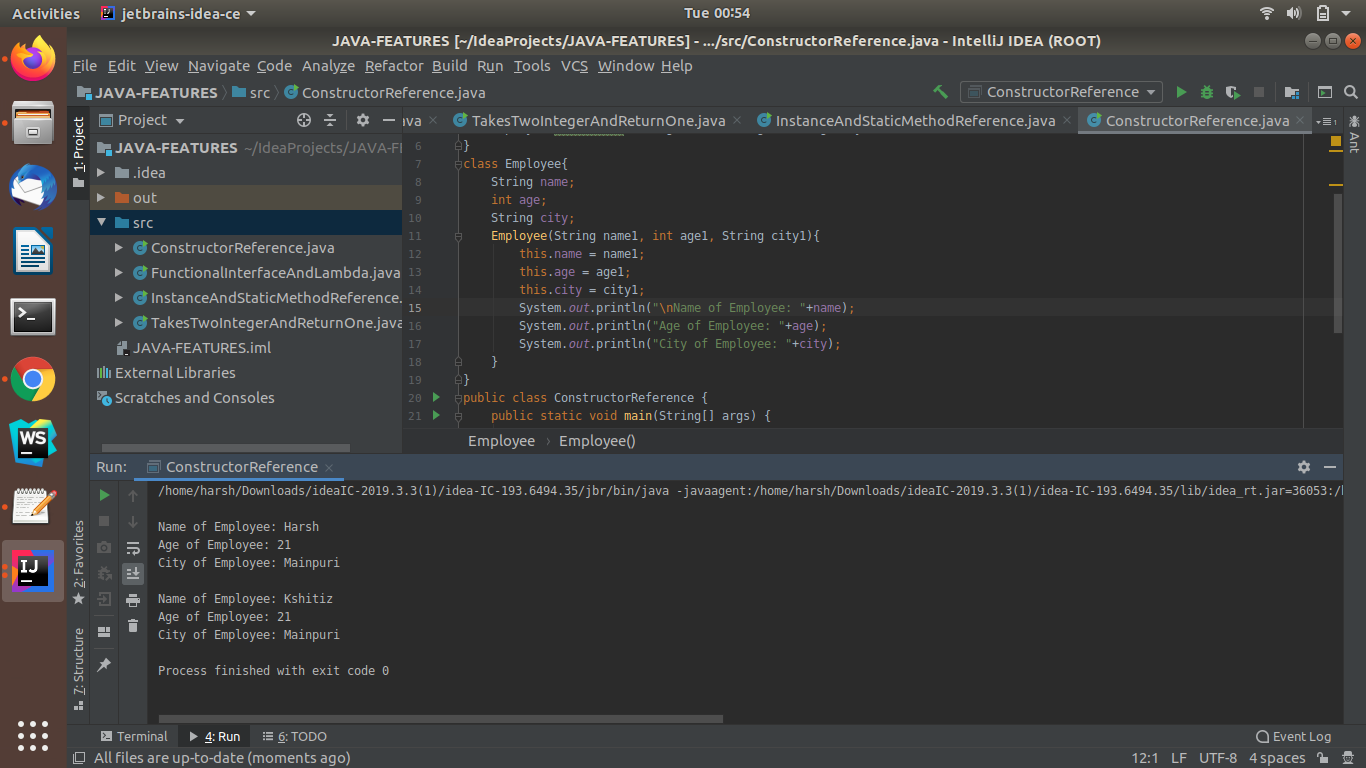
employee1.setdetails("Harsh",21,"Mainpuri");

employee1.setdetails("Kshitiz",21,"Mainpuri");

}

}

OUTPUT:



Ques 5:Implement following functional interfaces from java.util.function using lambdas:

(1)Consumer

Ans: **JAVA-CODE**

import java.util.Arrays;

import java.util.List;

import java.util.function.Consumer;

import java.util.function.Supplier;

public class FunctionalInterfaces {

public static void main(String[] args) {

List<String> list = Arrays.asList("Harsh","Ashish","Aman","Neha","Apoorva");

// Consumer<String> con = new Consumer<String>() {

// @Override

// public void accept(String s) {

// System.out.println(s);

// }

// };

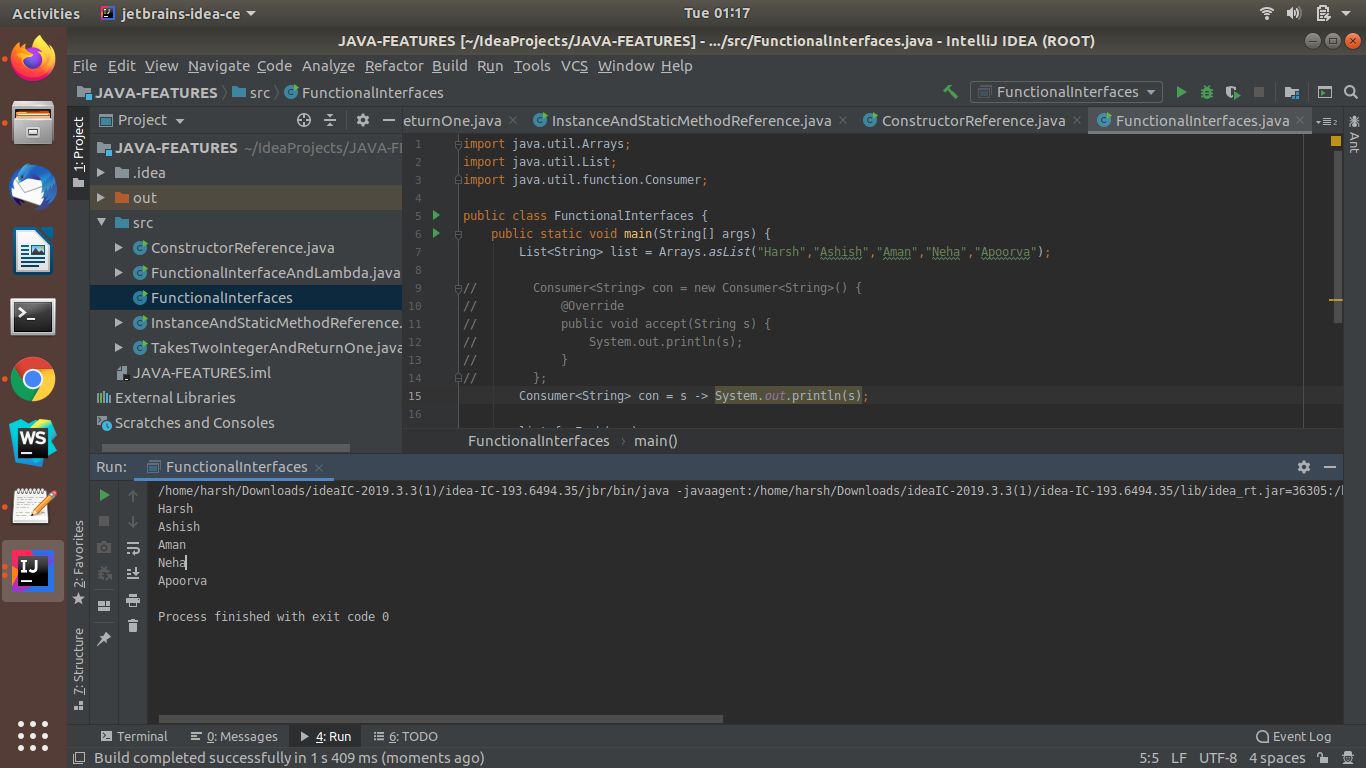
Consumer<String> con = s -> System.out.println(s);

list.forEach(con);

}

}

OUTPUT:



(2)Supplier

Ans: **JAVA-CODE**

import java.util.Arrays;

import java.util.List;

import java.util.function.Supplier;

public class FunctionalInterfaces{

public static void main(String[] args) {

List<String> list = Arrays.asList("Harsh", "Ashish", "Aman", "Neha","Apoorva");

list.forEach((item)->{printnames(()->item);});

}

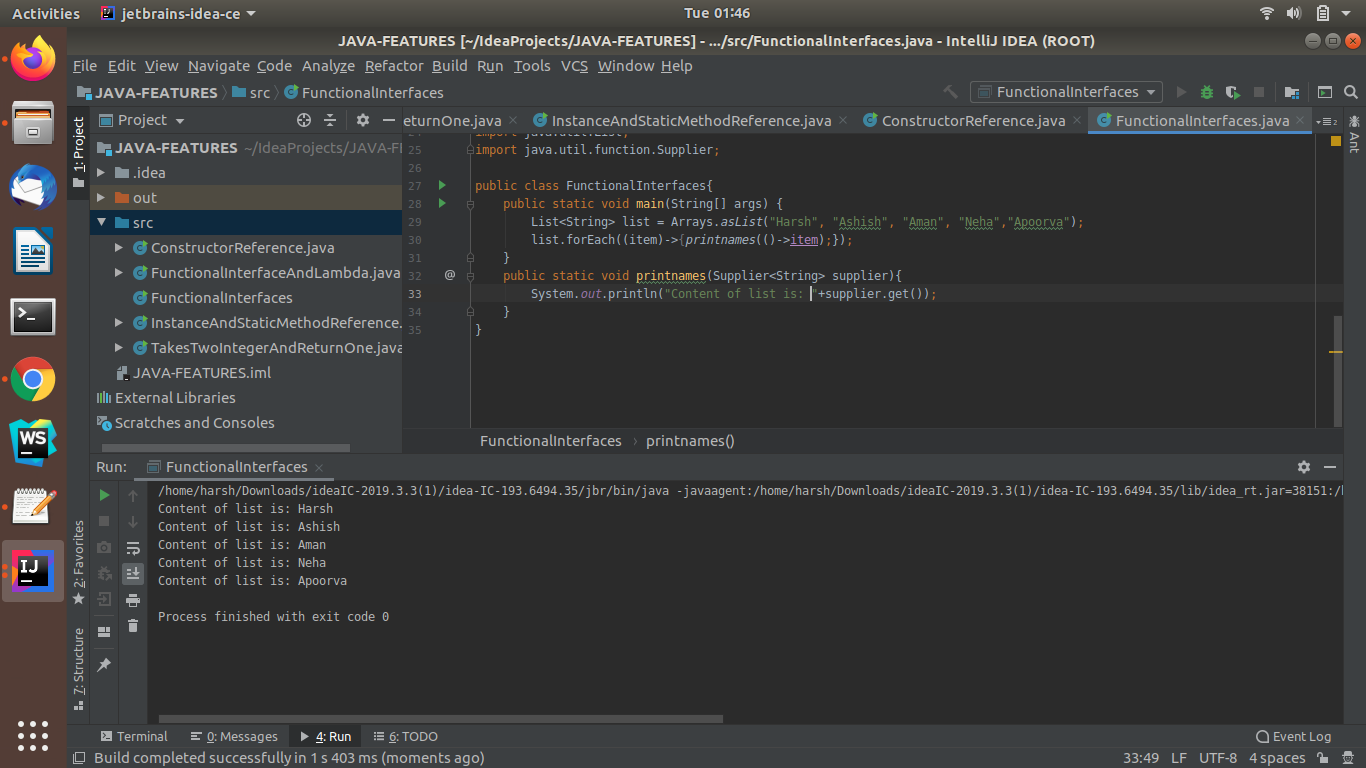
public static void printnames(Supplier<String> supplier){

System.out.println(supplier.get());

}

}

OUTPUT:



(3)Predicate

Ans:

public class FunctionalInterfaces{

public static void main(String[] args) {

Predicate<Integer> predicate = FunctionalInterfaces::isEven;

System.out.println("\n"+predicate.test(93));

System.out.println("\n"+predicate.test(98));

}

static boolean isEven(int num){

if (num % 2 == 0){

return true;

}

else{

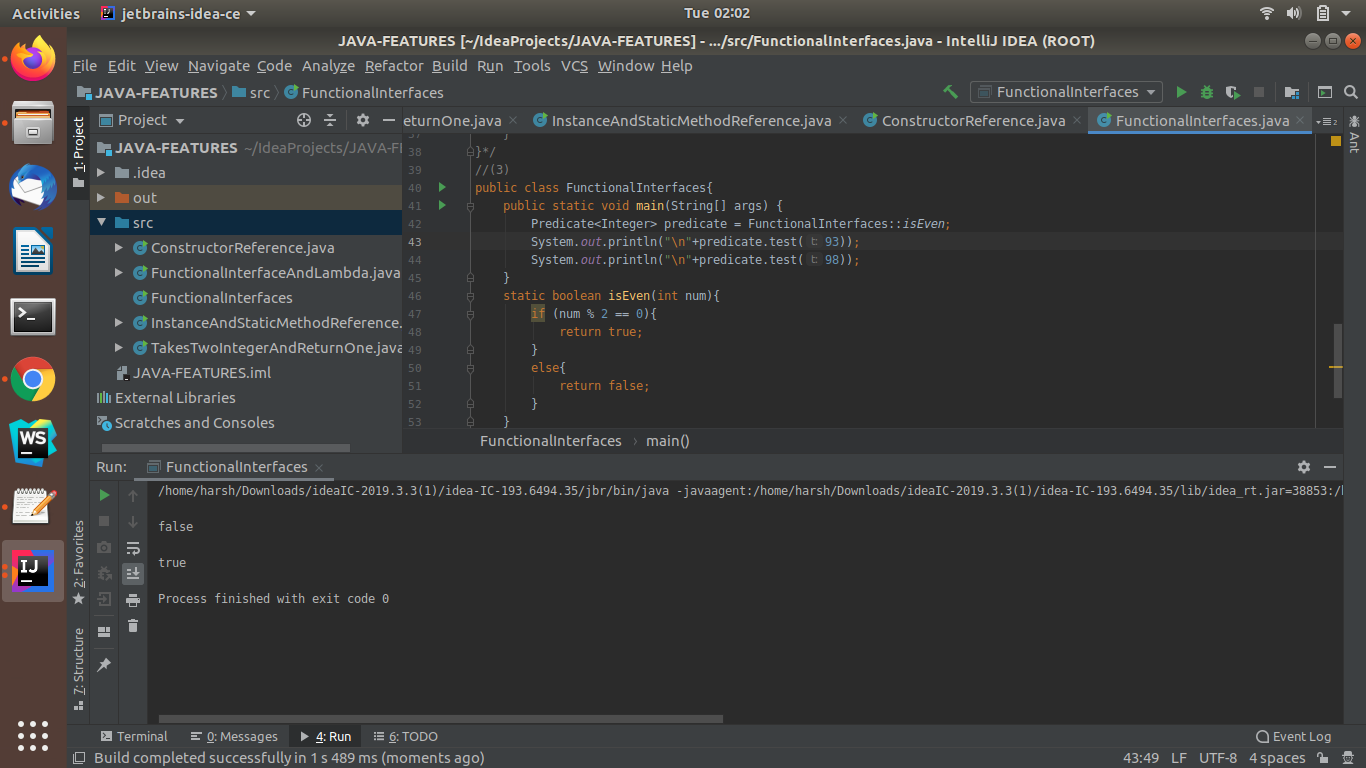
return false;

}

}

}

OUTPUT:



(4) Function

Ans : **JAVA-CODE**

public class FunctionalInterfaces{

public static void main(String[] args) {

Function<Integer,Integer> function = a -> a/2;

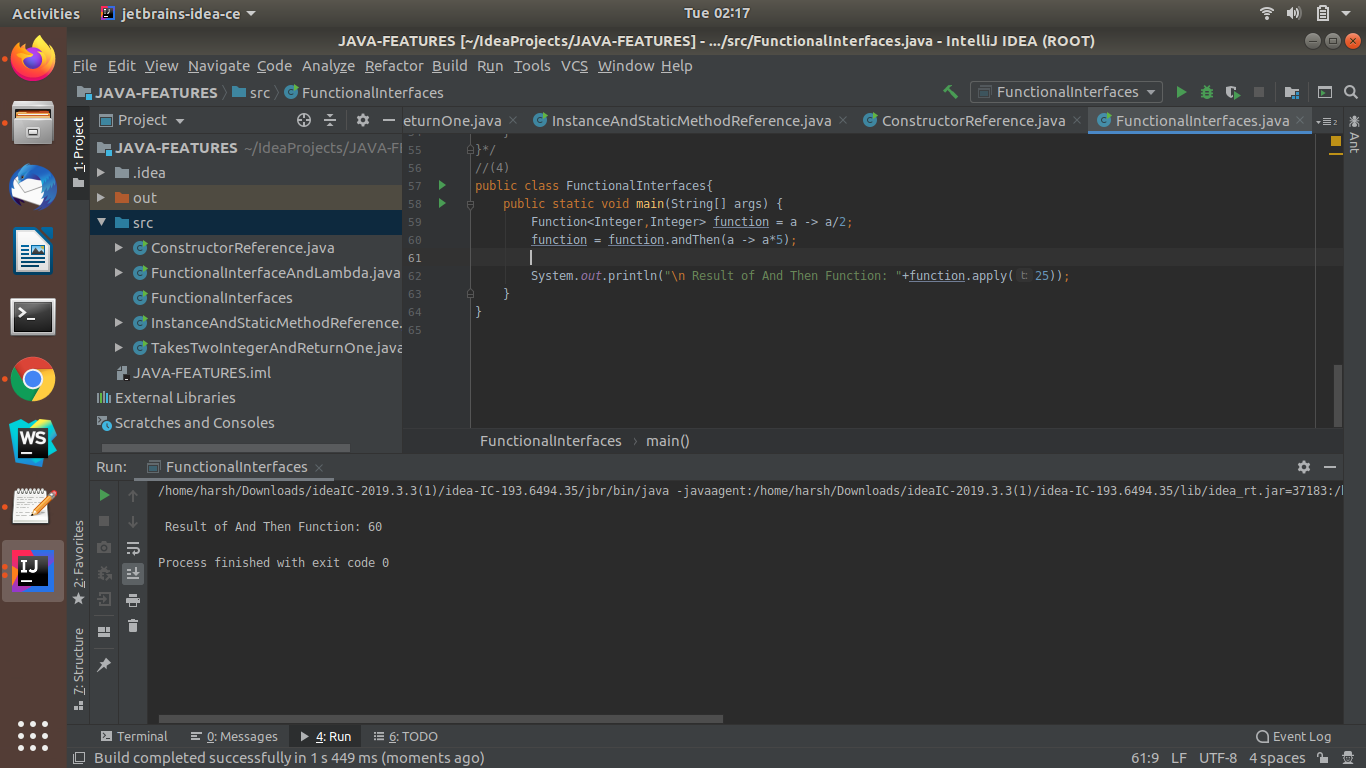
function = function.andThen(a -> a\*5);

System.out.println("\n Result of And Then Function: "+function.apply(25));

}

}

OUTPUT:



Ques 6:Create and access default and static method of an interface.

Ans: **JAVA-CODE**

interface DefaultAndStaticDemo{

static boolean isEven(int a){

if (a%2 ==0){

return true;

}

else{

return false;

}

}

default boolean isOdd(int a){

if(a%2 != 0){

return true;

}

else {

return false;

}

}

}

public class DefaultAndStaticMethodsOfInterface implements DefaultAndStaticDemo {

public static void main(String[] args) {

DefaultAndStaticMethodsOfInterface obj = new DefaultAndStaticMethodsOfInterface();

System.out.println(DefaultAndStaticDemo.isEven(24));

System.out.println(DefaultAndStaticDemo.isEven(23));

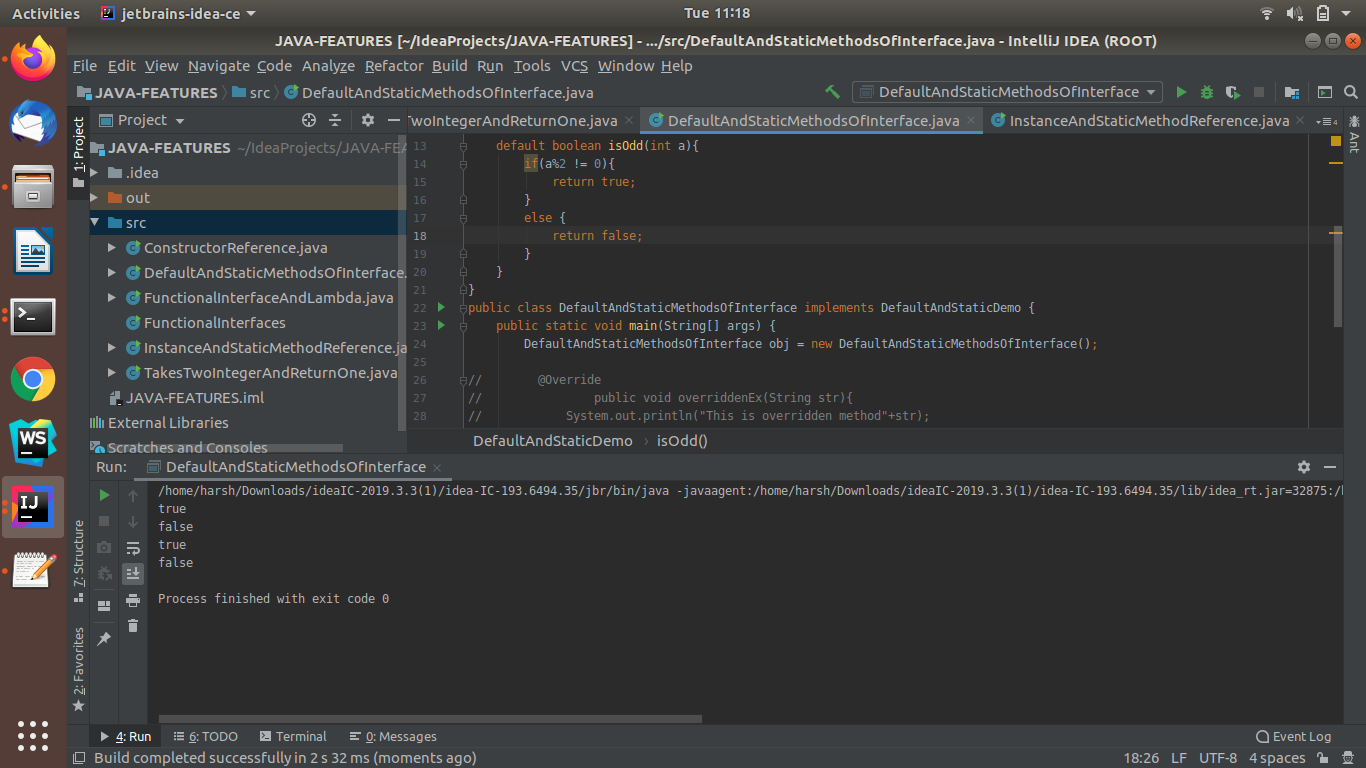
System.out.println(obj.isOdd(23));

System.out.println(obj.isOdd(24));

}

}

OUTPUT:



Ques 7:Override the default method of the interface.

Ans: **JAVA-CODE**

interface overrideDemo{

default void showDetails(){

System.out.println("\nThis is interface method");

}

}

public class OverrideDefaultMethod implements overrideDemo{

@Override

public void showDetails(){

System.out.println("\nThis is class Method:");

}

public static void main(String[] args) {

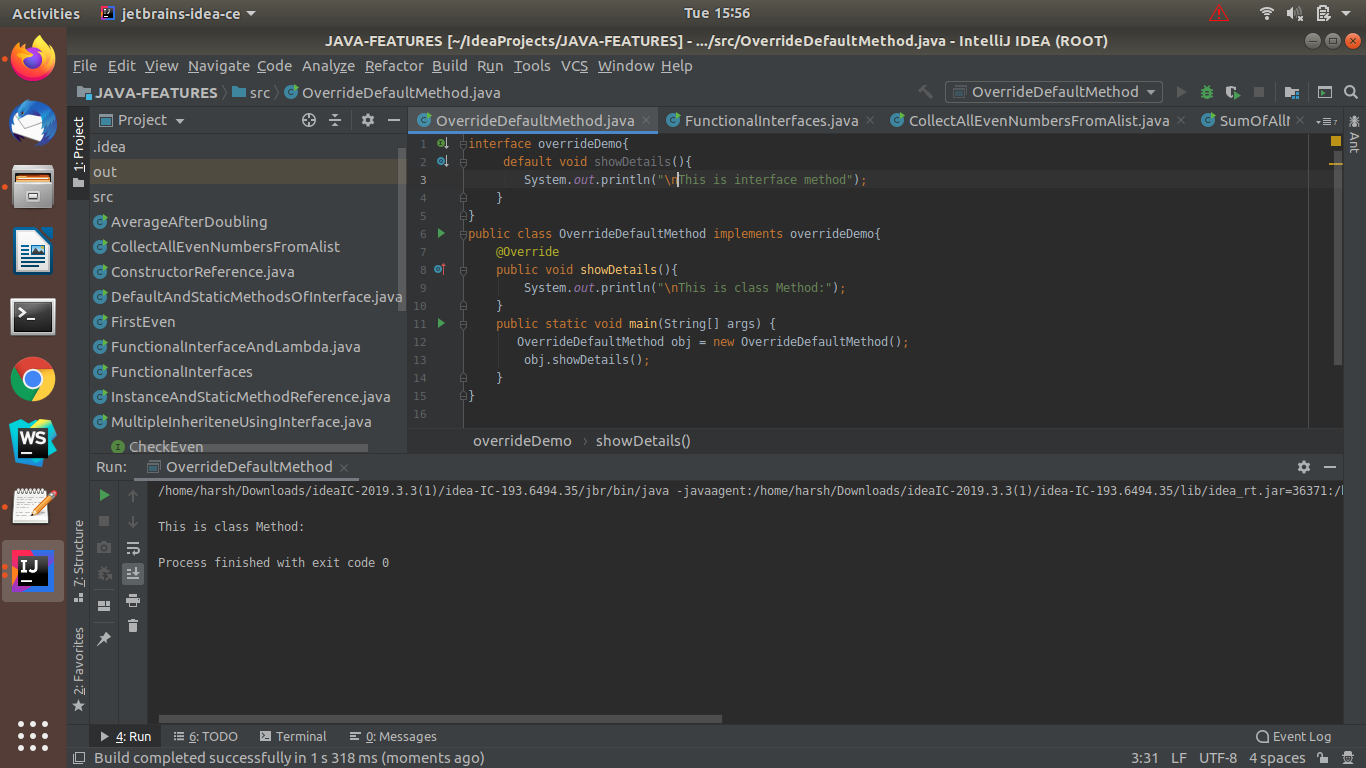
OverrideDefaultMethod obj = new OverrideDefaultMethod();

obj.showDetails();

}

}

OUTPUT:



Ques 8:Implement multiple inheritance with default method inside interface.

Ans: **JAVA-CODE**

interface CheckEven{

default boolean isEven(int a){

if (a%2 == 0){

return true;

}

else{

return false;

}

}

}

interface CheckOdd{

default boolean isOdd(int a){

if (a%2 != 0){

return true;

}

else{

return false;

}

}

}

public class MultipleInheriteneUsingInterface implements CheckEven,CheckOdd{

public static void main(String[] args) {

MultipleInheriteneUsingInterface obj = new MultipleInheriteneUsingInterface();

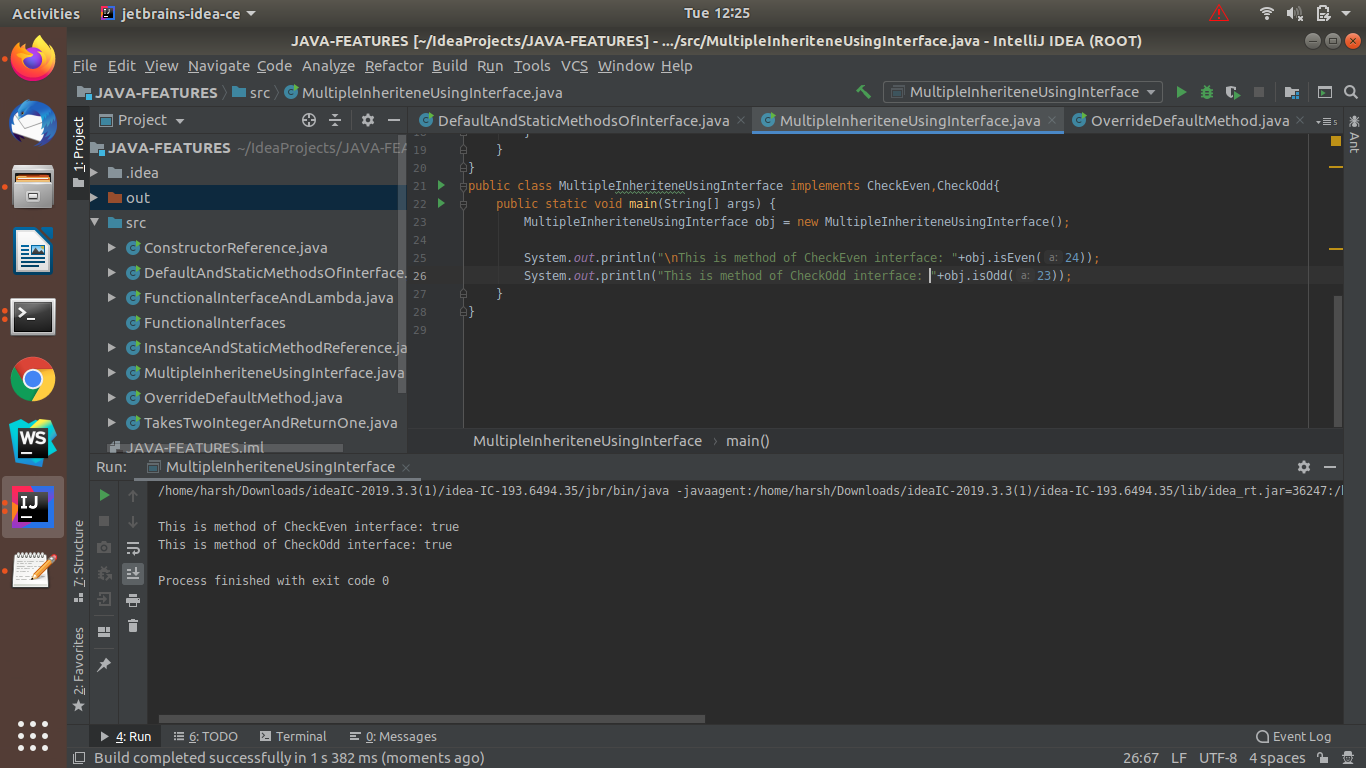
System.out.println("\nThis is method of CheckEven interface: "+obj.isEven(24));

System.out.println("This is method of CheckOdd interface: "+obj.isOdd(23));

}

}

OUTPUT:



Ques 9:Collect all the even numbers from an integer list.

Ans: **JAVA-CODE**

import java.util.Arrays;

import java.util.List;

import java.util.stream.Collectors;

public class CollectAllEvenNumbersFromAlist {

public static void main(String[] args) {

List<Integer> list = Arrays.asList(1,2,3,4,5,6,7,8,9,10,11,12);

System.out.println();

System.out.println(

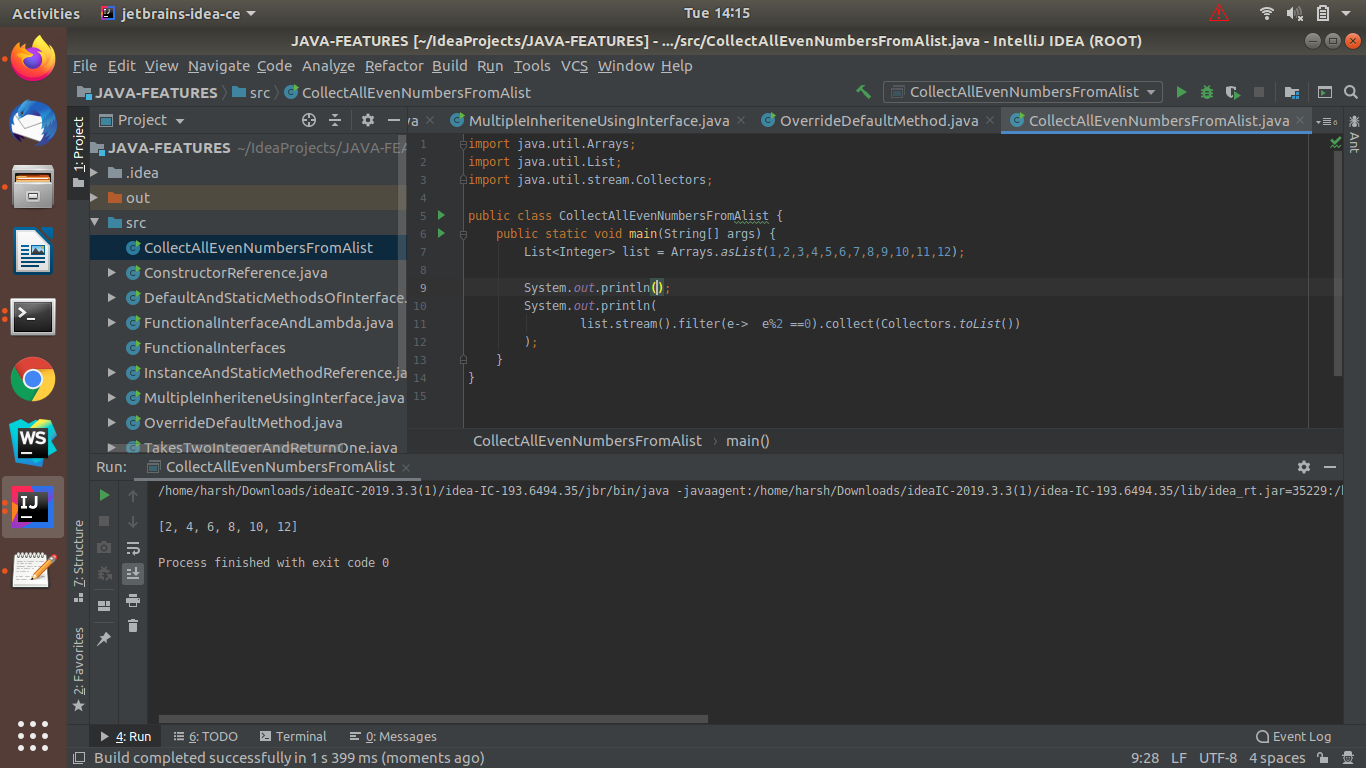
list.stream().filter(e-> e%2 ==0).collect(Collectors.toList())

);

}

}

OUTPUT:



Ques 10: Sum all the numbers greater than 5 in the integer list.

Ans: **JAVA-CODE**

import java.util.Arrays;

import java.util.List;

public class SumOfAllNumbersGreaterThanFive {

public static void main(String[] args) {

List<Integer> list = Arrays.asList(1,2,3,4,5,6,7,8,9,10,11,12);

System.out.println("sum of numbers greater than 5 is :");

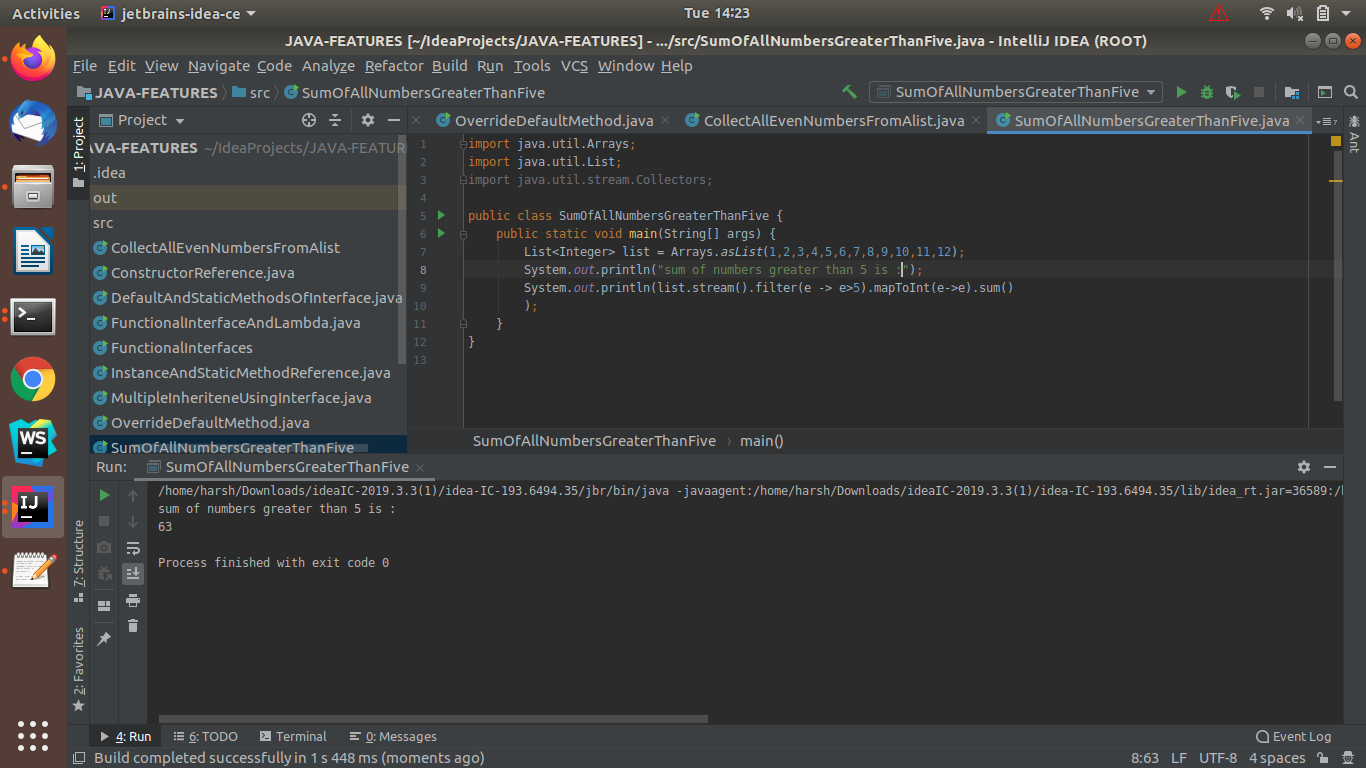
System.out.println(list.stream().filter(e -> e>5).mapToInt(e->e).sum()

);

}

}

OUTPUT:



Ques 11:Find average of the number inside integer list after doubling it.

Ans: **JAVA-CODE**

import java.util.Arrays;

import java.util.List;

public class AverageAfterDoubling {

public static void main(String[] args) {

List<Integer> list = Arrays.asList(1,2,3,4,5,6,7,8,9,10,11,12);

System.out.println("Initial Average Of list :");

System.out.println(list.stream().mapToInt(e->e).average()

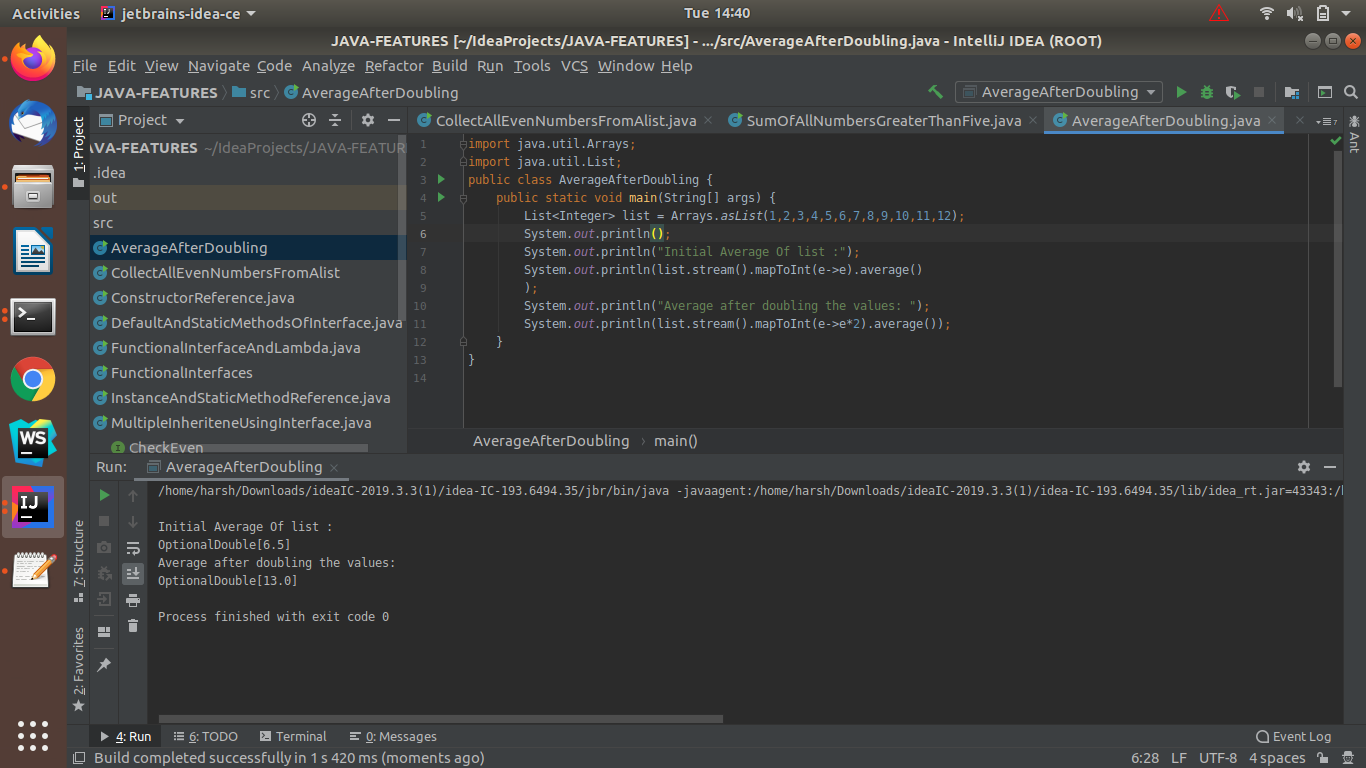
);

System.out.println("Average after doubling the values: ");

System.out.println(list.stream().mapToInt(e->e\*2).average());

}

}

OUTPUT:

Ques 12:Find the first even number in the integer list which is greater than 3.

Ans: **JAVA-CODE**

import java.util.Arrays;

import java.util.List;

public class FirstEven {

public static void main(String[] args) {

List<Integer> list = Arrays.asList(1,2,3,4,5,6,7,8,9,10,11,12);

System.out.println();

System.out.println("First Even Number Greater Than 3 is:");

System.out.println(list.stream().filter(integer -> integer >3).filter(integer -> integer%2 == 0).findFirst()

);

}

}

OUTPUT:

