**JAVA8 ASSIGNMENT**

**RAGHAV GUPTA**

**---------------------------------------------------------------**

* **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) Concatenation of 2 string Parameter (String , String ) Return (String)**
  + **(4) Convert a string to uppercase and return . Parameter (String) Return (String)**

============= COMMON PROGRAM ===============

**package** java8Assignment;

*// part A*

@FunctionalInterface

**interface** MyInterface1{

**boolean** compare(**int** a, **int** b);

}

*// part B*

@FunctionalInterface

**interface** MyInterface2{

**int** increment(**int** a);

}

*// part C*

@FunctionalInterface

**interface** MyInterface3{

String concat(String a, String b);

}

*// part D*

@FunctionalInterface

**interface** MyInterface4{

String toUpper(String s);

}

**public class** Question1{

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

System.***out***.println(**"part A ---- "**);

MyInterface1 mi1 = (a,b)->a>b;

System.***out***.println( mi1.compare(5,6) );

System.***out***.println( mi1.compare(6,5) );

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

System.***out***.println(**"part B ---- "**);

MyInterface2 mi2 = (a)->a+1;

System.***out***.println( mi2.increment(10));

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

System.***out***.println(**"part C ---- "**);

MyInterface3 mi3 = (a,b)->a+b;

System.***out***.println( mi3.concat(**"raghav "**, **"gupta"**) );

System.***out***.println( mi3.concat(**"mohit "**, **"gupta"**) );

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

System.***out***.println(**"part D ---- "**);

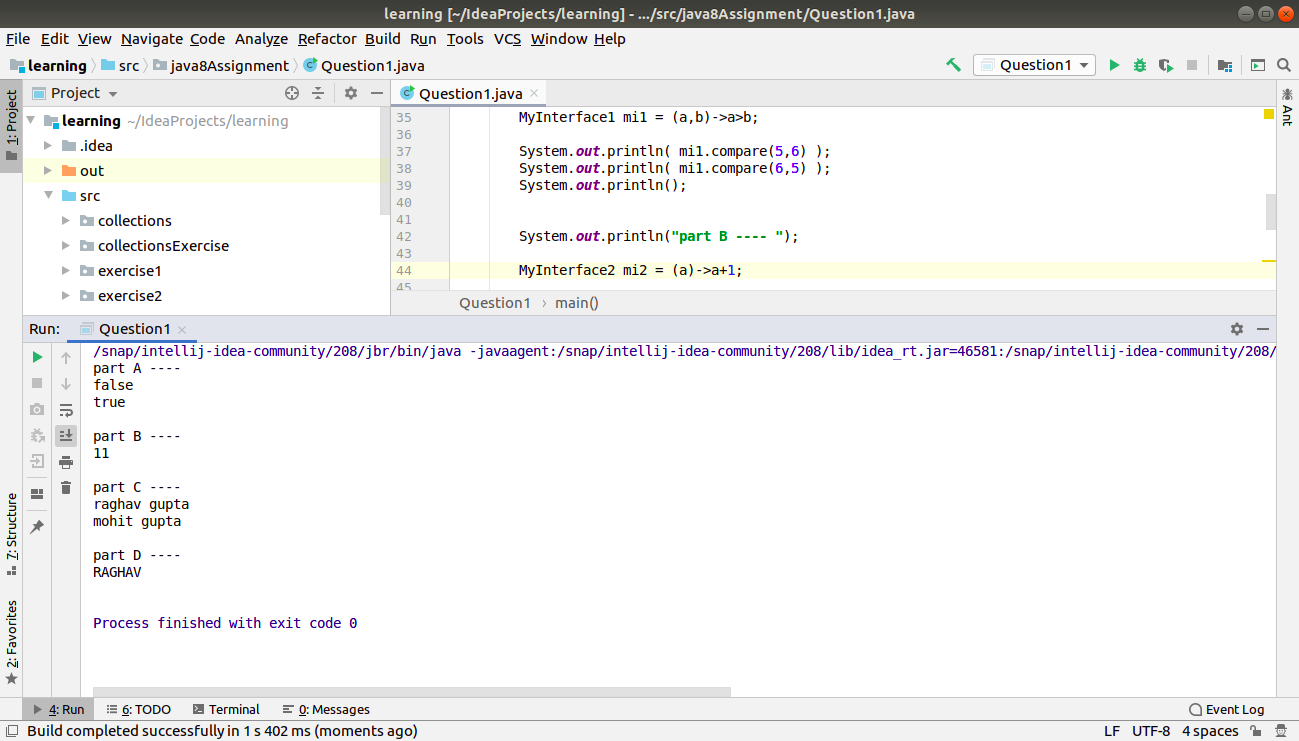
MyInterface4 mi4 = (a)->a.toUpperCase();

System.***out***.println( mi4.toUpper(**"raghav"**) );

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

}

}



============================================================================

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

**package** java8Assignment;

@FunctionalInterface

**interface** MyInterface5{

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

}

**public class** Question2 {

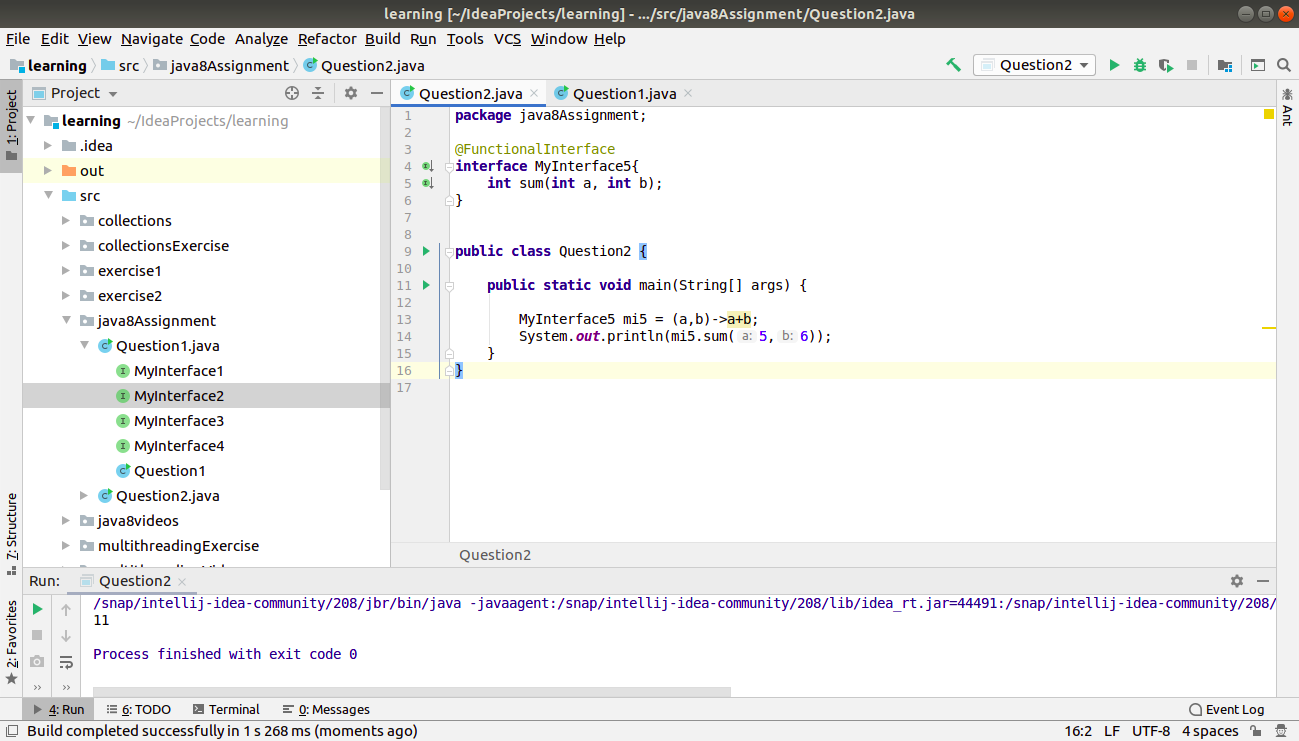
**public static void** main(String[] args) {

MyInterface5 mi5 = (a,b)->a+b;

System.***out***.println(mi5.sum(5,6));

}

}



============================================================================

* **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.**

**package** java8Assignment;

**import** org.w3c.dom.ls.LSOutput;

@FunctionalInterface

**interface** MyInterface6{

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

}

**class** Temp{

**public static int** multiply(**int** a, **int** b){

**return** a\*b;

}

**public int** add(**int** a, **int** b){

**return** a+b;

}

**public int** subtract(**int** a, **int** b){

**return** a-b;

}

}

**public class** Question3 {

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

MyInterface6 mi6 = Temp::*multiply*;

System.***out***.println(mi6.operation(5,6));

*// see, we can change in the same object also.*

*// MyInterface6 mi7 = new Temp()::add;*

mi6 = **new** Temp()::add;

System.***out***.println(mi6.operation(5,6));

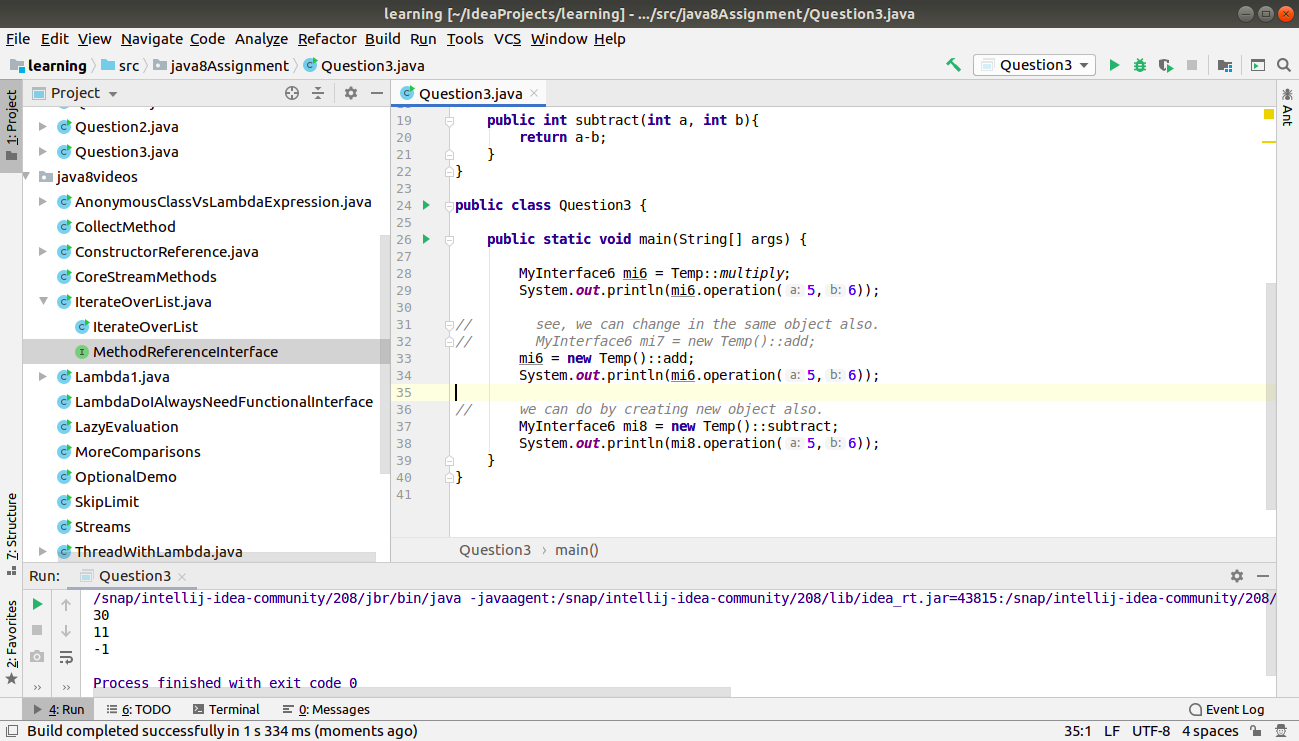
*// we can do by creating new object also.*

MyInterface6 mi8 = **new** Temp()::subtract;

System.***out***.println(mi8.operation(5,6));

}

}



============================================================================

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

**package java8Assignment;**

**@FunctionalInterface**

**interface getEmployee{**

**Employee getInstance(String name, int age, String city);**

**}**

**class Employee{**

**String name;**

**int age;**

**String city;**

**Employee(){**

**}**

**Employee(String name, int age, String city){**

**this.name = name;**

**this.age = age;**

**this.city = city;**

**}**

**public String toString(){**

**return "i am " + name+" of age "+age + " from " + city;**

**}**

**}**

**public class Question4 {**

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

**getEmployee ge = Employee::new;**

**Employee e1 = ge.getInstance("raghav", 22, "Vegas");**

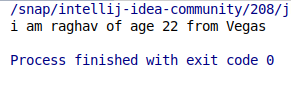
**System.*out*.println(**

**e1.toString()**

**);**

**}**

**}**

****

============================================================================

* **Implement following functional interfaces from java.util.function using lambdas:**
  + **(1) Consumer**
  + **(2) Supplier**
  + **(3) Predicate**
  + **(4) Function**

**package** java8Assignment;

**import** java.util.Random;

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

**import** java.util.function.Function;

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

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

**public class** Question5 {

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

System.***out***.println(**" ---- Consumer ----"**);

Consumer<Integer> con = a->System.***out***.println(a);

con.accept(5);

System.***out***.println(**"---- Supplier ----"**);

Supplier supp = ()->{

Random random = **new** Random();

**return** random.nextInt(1000);

};

System.***out***.println(supp.get());

System.***out***.println(**"---- Predicate ----"**);

Predicate<Integer> pred = a->a>0;

System.***out***.println(pred.test(7));

System.***out***.println(pred.test(-4));

System.***out***.println(**"---- Function ----"**);

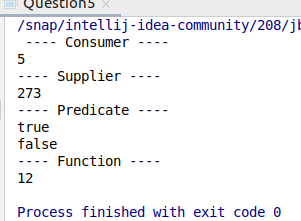
Function<String, Integer> func = (a)->a.length();

**int** value = func.apply(**"raghav gupta"**);

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

}

}



============================================================================

* **Create and access default and static method of an interface.**

**package** java8Assignment;

**interface** TestInterface

{

*// abstract method*

**default public int** square(**int** a){

**return** a\*a;

}

*// static method*

**static void** show()

{

System.***out***.println(**"Static Method Executed"**);

}

}

**class** Question6 **implements** TestInterface

{

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

{

Question6 q = **new** Question6();

**int** result = q.square(4);

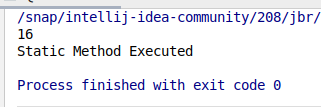
System.***out***.println(result);

*// Static method executed*

TestInterface.*show*();

}

}



============================================================================

* **Override the default method of the interface.**

**package** java8Assignment;

**interface** Test{

**default public void** introduce(){

System.***out***.println(**"i am the test interface..... "**);

}

**public int** square(**int** a);

}

**public class** Question7 **implements** Test{

*// implement abstract method of interface*

**public int** square(**int** a){

**return** a\*a;

}

*// override default method*

**public void** introduce(){

System.***out***.println(**"i have been implemented by Question6"**);

}

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

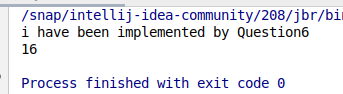
Question7 q = **new** Question7();

q.introduce();

System.***out***.println(q.square(4));

}

}



============================================================================

* **Implement multiple inheritance with default method inside interface.**

**package** java8Assignment;

**interface** Test1

{

*// default method*

**default void** show()

{

System.***out***.println(**"Default Test1"**);

}

}

**interface** Test2

{

*// Default method*

**default void** show()

{

System.***out***.println(**"Default Test2"**);

}

}

*// Implementation class code*

**class** Question8 **implements** Test1, Test2

{

*// Overriding default show method*

**public void** show()

{

*// use super keyword to call the show*

*// method of Test1 interface*

Test1.**super**.show();

*// use super keyword to call the show*

*// method of Test2 interface*

Test2.**super**.show();

System.***out***.println(**"i am show method of Question8 class"**);

}

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

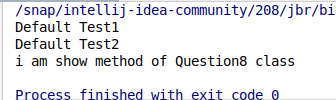
{

Question8 q = **new** Question8();

q.show();

}

}



============================================================================

* **Collect all the even numbers from an integer list.**

**package** java8Assignment;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.List;

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

**public class** Question9 {

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

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

System.***out***.println(

list

.stream()

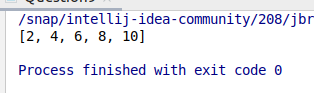
.filter(e->e%2==0)

.collect(Collectors.*toList*())

);

}

}



============================================================================

* **Sum all the numbers greater than 5 in the integer list.**

**package** java8Assignment;

**import** java.util.Arrays;

**import** java.util.List;

**public class** Question10 {

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

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

System.***out***.println(

list

.stream()

.filter(e->e>5)

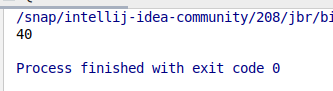
.mapToInt(e->e)

.sum()

);

}

}



============================================================================

* **Find average of the number inside integer list after doubling it.**

**package** java8Assignment;

**import** java.util.Arrays;

**import** java.util.List;

**public class** Question11 {

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

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

System.***out***.println(

list

.stream()

.mapToInt(e->e\*2)

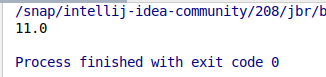
.average()

.getAsDouble()

);

}

}



============================================================================

* **Find the first even number in the integer list which is greater than 3.**

**package** java8Assignment;

**import** java.util.Arrays;

**import** java.util.List;

**public class** Question12 {

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

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

System.***out***.println(

list

.stream()

.filter(e->e>3)

.filter(e->e%2==0)

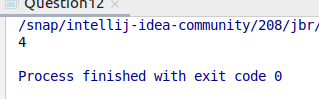
.findFirst()

.get()

);

}

}



============================================================================