JAVA-8 FEATURES

YUKTI SHARMA

1. **Write the following a functional interface and implement it using lambda:**
   a. **(1) First number is greater than second number or not Parameter (int ,int) Return boolean**
   b. **(2) Increment the number by 1 and return incremented value Parameter (int) Return int**
   c. **(3) Concatination of 2 string Parameter (String , String) Return (String)**
   d. **(4) Convert a string to uppercase and return . Parameter (String) Return (String)**

```
interface Greater{
    boolean greater(int a,int b);
}

interface Increment{
    int increment(int a);
}
interface Concat{
    String concat(String one, String two);
}
interface UpperCase{
    String upper(String input);
}

public class Ques1 {
    public static void main(String[] args) {

        Greater object1= (a,b)->(a>b)?true: false;
        System.out.println("is 5 greater than 2? "+object1.greater( a: 5, b: 2));

        Increment object2 = a->a+1;
        System.out.println("increment 8 by 1= "+object2.increment( a: 8));

        Concat object3= (a,b)->a+b;
        System.out.println("concat two strings- "+object3.concat( one: "heyyy ", two: "there!"));

        UpperCase object4= a->a.toUpperCase();
        System.out.println("upper case- "+object4.upper( input: "yukti sharma"));
    }
}
```

```
/home/yukti/.sdkman/candidates/java/8.0.202-amzn/bin/java ...
is 5 greater than 2? true
increment 8 by 1= 9
concat two strings- heyyy there!
upper case- YUKTI SHARMA

Process finished with exit code 0
```

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

```
public class Ques2 {

    static int oneReturn(int one, int two) { return one; }

    public static void main(String[] args) {

        Demo demo1 = Ques2::oneReturn;
        System.out.println("returning one integer- " + demo1.demo( a: 89,  b: 33));
    }

}

interface Demo{
    int demo(int a, int b);
}
```

```
/home/yukti/.sdkman/candidates/java/8.0.202-amzn/bin/java
returning one integer- 89

Process finished with exit code 0
```

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.**

```java
interface Sum{
    int sum(int a, int b);
}

interface Subt{
    int sub(int a,int b);
}

interface Mul{
    int multiply(int x, int y);
}
```

```java
public class Ques3 {

    int add(int a, int b) { return a+b; }

    int subt(int a,int b) { return a-b; }

    static int multiply(int a, int b) { return a*b; }
    public static void main(String[] args) {

        Sum obj= new Ques3()::add;
        System.out.println("sum of 6,7== "+obj.sum( a: 6, b: 7));

        Subt obj2= new Ques3()::subt;
        System.out.println("subtracting 91, and 1= "+obj2.sub( a: 91, b: 1));

        Mul obj3= Ques3::multiply;
        System.out.println("multiplying 40, and 4=  s"+obj3.multiply( x: 40, y: 4));
    }
}
```

```
sum of 6,7== 13
subtracting 91, and 1= 90
multiplying 40, and 4=  s160

Process finished with exit code 0
```

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

```java
public class Ques4 {

    public static void main(String[] args) {
        EmployeeInterface object= Employee::new;
        Employee emp= object.getEmployee( name: "Yukti", age: 20, city: "Noida");
        System.out.println("name is "+emp.name+" ,Age is- "+emp.age+" ,city is- "+emp.city);
    }
}

class Employee{
    String name;
    Integer age;
    String city;

    public Employee(String name, Integer age, String city) {
        this.name = name;
        this.age = age;
        this.city = city;
    }
}

interface EmployeeInterface{
    Employee getEmployee(String name,Integer age,String city);
}
```

```
sum of 6,7== 13
subtracting 91, and 1= 90
multiplying 40, and 4=  s160

Process finished with exit code 0
```

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

   (1) Consumer (2) Supplier (3) Predicate (4) Function

```java
import java.util.function.Consumer;
import java.util.function.Function;
import java.util.function.Predicate;
import java.util.function.Supplier;

public class Ques5 {
    public static void main(String[] args) {

        Predicate<Integer> isGreaterThan5 = e -> e > 5 ? true : false;
        System.out.println("is 2 greater than 5- " + isGreaterThan5.test( 2));

        Consumer<String> display= e-> System.out.println("consumer running on string= "+e);
        display.accept( "consumerrr");

        Supplier<Integer> supplier = ()->1;
        System.out.println("value returned from supplier is- "+supplier.get());

        Function<Integer,Integer> square = e->e*e;
        System.out.println("square of 13 is- "+square.apply( 13));

    }
}
```

```
/home/yukti/.sdkman/candidates/java/8.0.202-
is 2 greater than 5- false
consumer running on string= consumerrr
value returned from supplier is- 1
square of 13 is- 169

Process finished with exit code 0
```

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

```java
interface DefaultExample{
    default void sum()
    {  int sum=0;
        for(int i=1;i<=10;i++)
        {
            sum+=i;
        }
        System.out.println("sum of first 10 natural numbers is- "+sum);
    }

    static void display()
    {
        System.out.println("Displaying output from static method of interface");
    }
}

public class Ques6 implements DefaultExample
{
    public static void main(String[] args) {
        Ques6 obj= new Ques6();
        obj.sum();
        DefaultExample.display();
    }
}
```

```java
import java.util.function.Consumer;
import java.util.function.Function;
import java.util.function.Predicate;
import java.util.function.Supplier;

public class Ques5 {
    public static void main(String[] args) {

        Predicate<Integer> isGreaterThan5 = e -> e > 5 ? true : false;
        System.out.println("is 2 greater than 5- " + isGreaterThan5.test( t: 2));

        Consumer<String> display= e-> System.out.println("consumer running on string= "+e);
        display.accept( t: "consumerrr");

        Supplier<Integer> supplier = ()->1;
        System.out.println("value returned from supplier is- "+supplier.get());

        Function<Integer,Integer> square = e->e*e;
        System.out.println("square of 13 is- "+square.apply( t: 13));

    }
}
```

7. Override the default method of the interface.

```java
interface DefaultDemo {
    default void display() { System.out.println("Display from Demo interface"); }
}

public class Ques7 implements DefaultDemo {

    public void display() { System.out.println("Display from Ques7"); }

    public static void main(String[] args) {
        Ques7 defaultMethods= new Ques7();
        defaultMethods.display();
    }
}
```

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

```java
interface inter1 {
    default void display(){
        System.out.println("inter1");
    }
}

interface child1 extends inter1{
    default void display(){
        System.out.println("child1");
    }
}

interface child2 extends inter1{
    default void display(){
        System.out.println("child2");
    }
}

public class Ques8 implements child1,child2 {
    public void display() { System.out.println("Ques8"); }

    public static void main(String[] args) {
        Ques8 defaultMethods=new Ques8();
        defaultMethods.display();
    }
}
```

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

```java
import java.util.Arrays;
```

```
import java.util.List;

import java.util.stream.Collectors;


public class Ques9 {

  public static void main(String[] args) {

     List<Integer> list = Arrays.asList(32,5,9,67,90,24,65);

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

  }

}
```

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

```
import java.util.Arrays;

import java.util.List;

import java.util.stream.Collectors;

public class Ques10 {

  public static void main(String[] args) {

     List<Integer> list = Arrays.asList(3,8,65,9,4,0,1,2,6);

     System.out.println("sum of numbers more than 5  are- "+

          list.stream()

               .filter(e->e>5)

               .collect(Collectors.summingInt(e->e)));

  }

}
```

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

```
                    import java.util.Arrays;

import java.util.List;

import java.util.stream.Collectors;



public class Ques11 {

  public static void main(String[] args) {



    List<Integer> list = Arrays.asList(6, 3, 0, 7, 40, 10, 2);

    System.out.println(list.stream().collect(Collectors.averagingInt(e->e*2)));

  }

}
```

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

```
import java.util.Arrays;
import java.util.List;
import java.util.Optional;

public class Ques12 {
  public static void main(String[] args) {
    List<Integer> list = Arrays.asList(5, 13, 2, 9, 7, 56, 43, 12);
    Optional<Integer> optional = list.stream().filter(e -> e % 2 == 0).filter(e -> e > 3).findFirst();
    if(optional.isPresent())
      System.out.println(optional.get());
    else
      System.out.println("no element present");
  }
}
```