Q1 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)

1 Code -

```java
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;

interface tushar {
    boolean anmol(int a, int b);
}

public class Q1 {
    public static void main(String[] args) {

        tushar naresh = (int x, int y) -> {
            if (x > y)
                return true;
            else
                return false;
        };
        boolean veena = naresh.anmol( a: 15,  b: 25);
        System.out.println(veena);
    }
}
```

Ouput -

```
/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
false

Process finished with exit code 0
```

2 Code -

```java
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;

interface veena{
    int tush( int a);
        }

public class Q12 {
    public static void main(String[] args) {

        veena myobj = (int x) ->{x=++x;

        return x;

        };
        int answer=myobj.tush( a: 52);
        System.out.println(answer);

    }
}
```

Output-

```
/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
53

Process finished with exit code 0
```

3 code-

Concatination of 2 string

```java
import com.sun.org.apache.xpath.internal.objects.XString;

import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;

interface var {
    String tush(String s ,String a);
}
public class Q13 {
    public static void main(String[] args) {
        var myobj=(String s, String a) ->{
        String b =s+a;
        return b;

        };
        String naresh=myobj.tush( s: "i am tushar", a: "i am naresh");
        System.out.println(naresh);

    }


}
```

Output-

```
/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
i am tushari am naresh

Process finished with exit code 0
```

4 code-

```java
import java.util.List;
import java.util.stream.Collectors;

interface obj {
    String tush(String a);
}

public class Q14 {
    public static void main(String[] args) {

        obj myobj = (String b) -> {
            b= b.toUpperCase();
             return b;

        };

        String tush = myobj.tush( a: "i am tushar");
        System.out.println(tush);

    }


}
```

Output-

```
/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
I AM TUSHAR

Process finished with exit code 0
```

Q2 Create a functional interface whose method takes 2 integers and return one integer.

Code -

```java
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;

interface mohit{
    int tushar(int a, int b);
}
public class Q2 {
    public static void main(String[] args) {

        mohit myobj=(int a,int b) ->{
            int c = a+b;
            return c;

        };

        int tushar=myobj.tushar( a: 12, b: 68);
        System.out.println(tushar);

    }
}
```

Output -

```
/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
80

Process finished with exit code 0
```

Q3 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.

Code-

```
interface myclass{
    int calculate(int a, int b);
}

class newclass{
    int add (int a,int b){
        return a+b;
    }
    int subtract(int a,int b){
        return a-b;
    }
     int multiply(int a,int b){
        return a*b;
    }
}
class classrefernce{
    public static void main(String[] args) {
        myclass numbers =new newclass()::add;
        System.out.println("Adding 10 and 8 is" +numbers.calculate( a: 10, b: 8));

         numbers =new newclass()::subtract;
        System.out.println("Subtracting 8 from 10 is" +numbers.calculate( a: 10, b: 8));

        numbers = new newclass()::multiply;
        System.out.println("Multiply 10 and 8 is" +numbers.calculate( a: 10, b: 8));
    }
}
```

Output-

```
/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Adding 10 and 8 is18
Subtracting 8 from 10 is2
Multiply 10 and 8 is80

Process finished with exit code 0
```

Q4 Create an Employee Class with instance variables (String) name, (Integer)age, (String)city and get the instance of the Class using constructor reference

Code-

```java
interface EmployeeHolder {
  Employee1 constructorReferenceMethod(String name, Integer age, String city);
}

class Employee1 {
  String name;
  Integer age;
  String city;


  public Employee1(String name, Integer age, String city) {
    this.name = name;
    this.age = age;
    this.city = city;
  }

  @Override
  public String toString() {
    return "Employee{" +
        "name='" + name + '\'' +
        ", age=" + age +
        ", city='" + city + '\'' +
        '}';
  }
}

public class q41 {
  public static void main(String[] args) {
    EmployeeHolder emp = Employee1::new;
    System.out.println(emp.constructorReferenceMethod("Emp", 22, "Gzb"));
    System.out.println(emp.constructorReferenceMethod("Emp2", 22, "Gzb"));
  }

}
```

Output-

```
/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Employee{name='Emp', age=22, city='Gzb'}
Employee{name='Emp2', age=22, city='Gzb'}

Process finished with exit code 0
```

Q5 Implement following functional interfaces from java.util.function using lambdas:

(1) Consumer

(2) Supplier

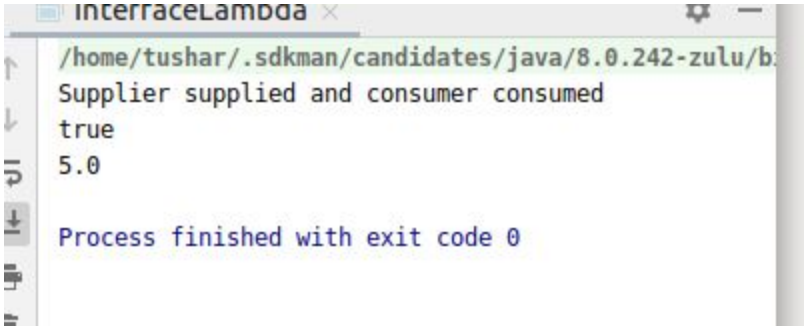
(3) Predicate

(4) Function

Code-

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

public class InterfaceLambda {
    public static void main(String[] args) {
        //Supplier example
        Supplier supply = ()->"Supplier has Supplied";
        String k = (String) supply.get();

        //Consumer example
        Consumer customer = (a)-> System.out.println("Supplier supplied and consumer consumed");
        customer.accept(k);

        //Predicate example
        Predicate<Integer> lesserthan = i -> (i < 18);
        System.out.println(lesserthan.test( t 10));

        //Function example
        Function<Integer, Double> half = a -> a / 2.0;
        System.out.println(half.apply( t 10));
    }
}
```

Output-



Q6 Create and access default and static method of an interface.

Code-

```java
interface newint{
    default void display(){
        System.out.println("Printed Default from Interface which prints only after creating an interface object");
    }
    static void displaystat(){
        System.out.println("Printed Using Static from Interface which prints after implementing the interface in class");
    }
}

public class StaticInterface implements newint{
    public static void main(String[] args) {
        newint.displaystat();
        newint interface1 = new StaticInterface();
        interface1.display();
    }
}
```

Output-

```
/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Printed Using Static from Interface which prints after implementing the interface in class
Printed Default from Interface which prints only after creating an interface object

Process finished with exit code 0
```

Q7 Override the default method of the interface.

Code-

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

public class Q7 {
    public static void main(String[] args) {
        myint interface1 = new myint() {
            @Override
            public void display() {
                System.out.println("Interface display overridden");
            }
        };
        interface1.display();
    }
}
```

Output-

```
/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Interface display overridden

Process finished with exit code 0
```

Q8 Implement multiple inheritance with default method inside interface.

Code-

```java
interface Parent1{
    default void display(){
        System.out.println("Printing for Parent 1");
    };
}
interface Parent2{
    default void print(){
        System.out.println("printing for Parent 2");
    };
}
public class MultipleInherit implements Parent1,Parent2 {
    public static void main(String[] args) {
        MultipleInherit inherit1 = new MultipleInherit();
        inherit1.display();
        inherit1.print();
    }

}
```

Output-

```
/home/tushar/.sdkman/candidates/java/8.0.242-zulu/b
Printing for Parent 1
printing for Parent 2

Process finished with exit code 0
```

Q9 Collect all the even numbers from an integer list.

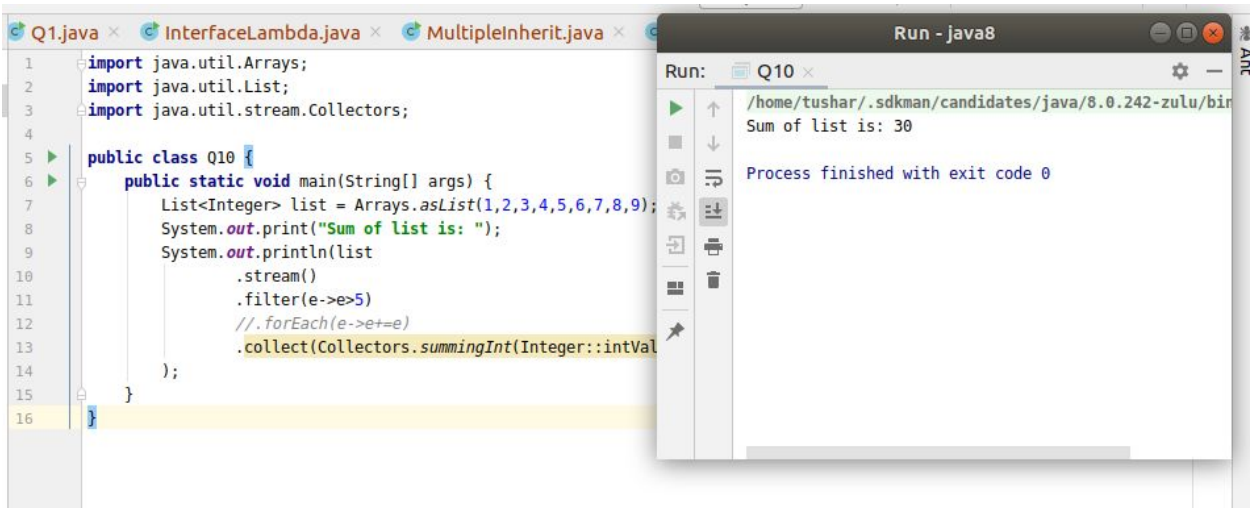
Code-

```java
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.stream.Collectors;

public class EvenNumbers {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        List<Integer> list = new ArrayList<>();
        System.out.println("Enter number of elements : ");
        int n = input.nextInt();
        int element;
        System.out.println("Enter numbers :");
        for(int i=1; i<=n; i++){
            element = input.nextInt();
            list.add(element);
        }
        System.out.println("List of Even numbers :");
        System.out.println( list.stream().filter(e-> e%2 == 0).collect(Collectors.toList()) );
    }
}
```

Output-

```
/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Enter number of elements :
5
Enter numbers :
2
8
9
11
15
List of Even numbers :
[2, 8]

Process finished with exit code 0
```

Q10 Sum all the numbers greater than 5 in the integer list



.

Q11 Find average of the number inside integer list after doubling it.

Code-

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


public class AverageInt {
    public static void main(String[] args) {
        List<Integer> list = Arrays.asList(1,2,3,4,5,6,7,8,9);
        System.out.print("Average before doing double: ");
        System.out.println(list.stream().mapToInt(e->e).average().orElse( other: -1));
        System.out.print("Average after doing double: ");
        System.out.print(list
                .stream()
                .mapToInt(e->e*2)
                .average().orElse( other: -1));
    }
}
```

Output-

```
/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Average before doing double: 5.0
Average after doing double: 10.0
Process finished with exit code 0
```

Q12 Find the first even number in the integer list which is greater than 3.

Code-

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

public class FindFirst {
    public static void main(String[] args) {
        List<Integer> list = Arrays.asList(1,2,3,4,5,6,7,8,9);
        System.out.print("First Number Greater then 3 is: ");
        System.out.println(list
                .stream()
                .filter(e->e>3)
                .findFirst().orElse( other: -1));
    }
}
```

Output

```
/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
First Number Greater then 3 is: 4

Process finished with exit code 0
```