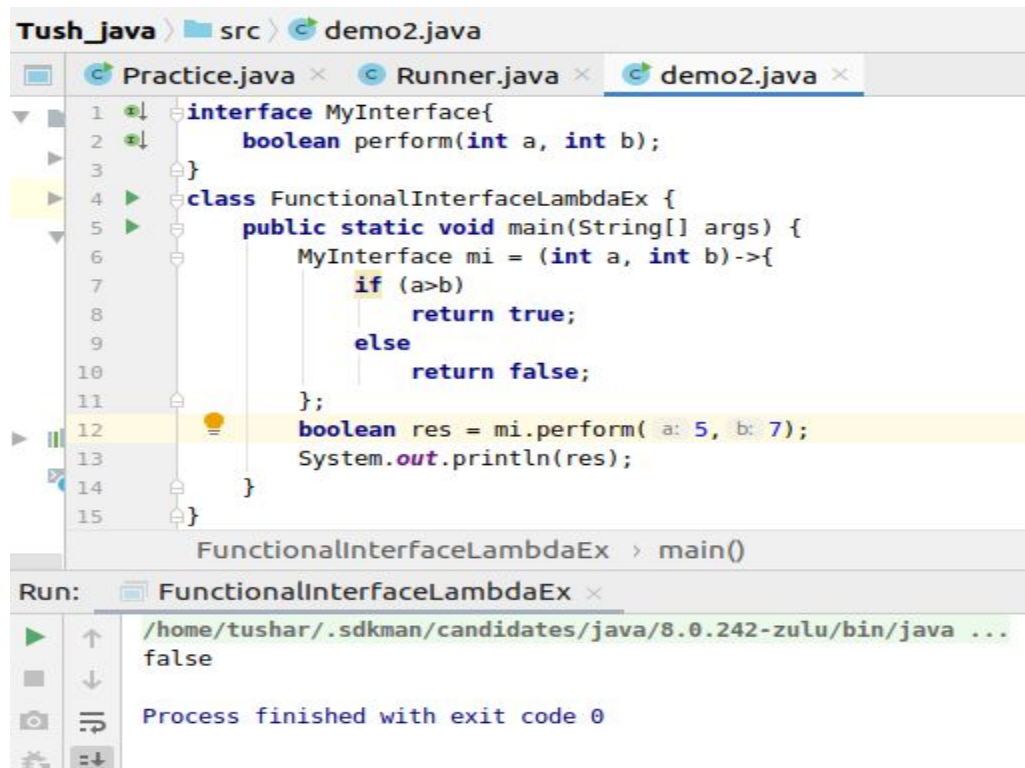


Q1. Write the following a functional interface and implement it using lambda:

- (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)

Answers:

1.



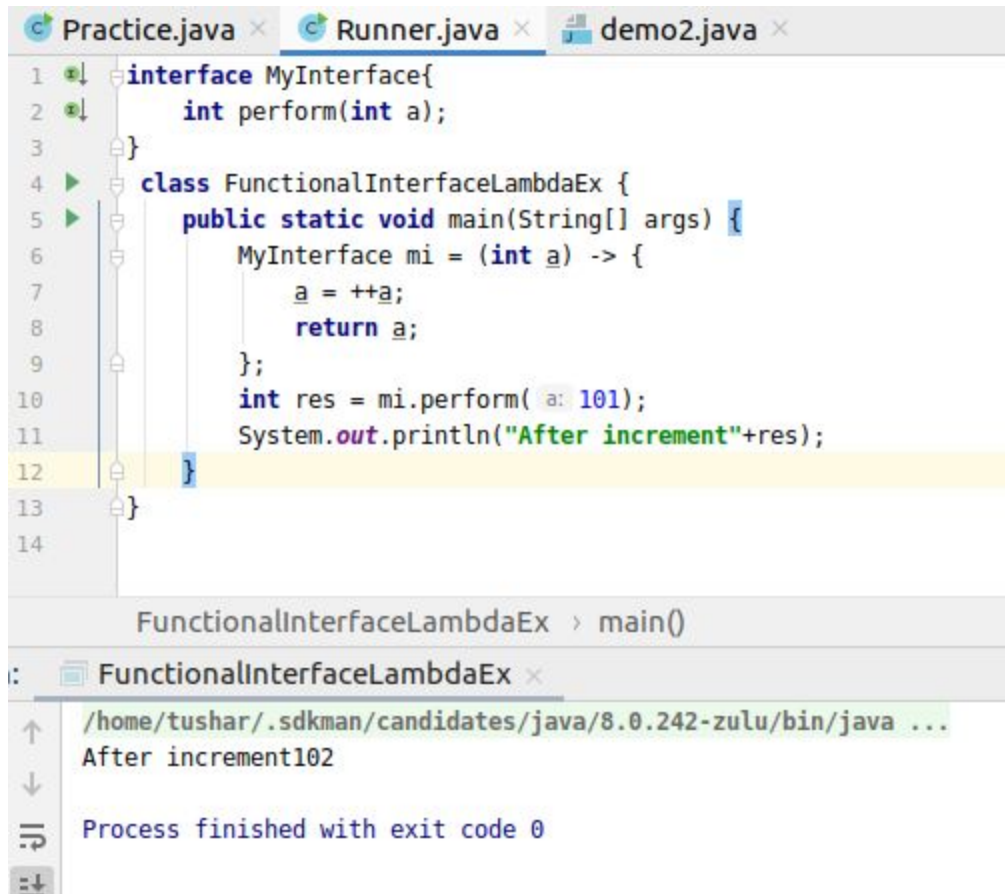
The screenshot shows an IDE with three tabs: Practice.java, Runner.java, and demo2.java. The demo2.java file contains the following code:

```
1 interface MyInterface{
2     boolean perform(int a, int b);
3 }
4 class FunctionalInterfaceLambdaEx {
5     public static void main(String[] args) {
6         MyInterface mi = (int a, int b)->{
7             if (a>b)
8                 return true;
9             else
10                return false;
11        };
12        boolean res = mi.perform( a: 5, b: 7);
13        System.out.println(res);
14    }
15 }
```

The Run console shows the output: false. The process finished with exit code 0.

```
interface MyInterface{
    boolean perform(int a, int b);
}
class FunctionalInterfaceLambdaEx {
    public static void main(String[] args) {
        MyInterface mi = (int a, int b)->{
            if (a>b)
                return true;
            else
                return false;
        };
        boolean res = mi.perform(5,7);
        System.out.println(res);
    }
}
```

2.



The screenshot shows an IDE with three tabs: Practice.java, Runner.java, and demo2.java. The main editor displays the code for Practice.java. The code defines an interface MyInterface with a method perform(int a). A class FunctionalInterfaceLambdaEx implements this interface in its main method. It creates a MyInterface instance mi with a lambda expression (int a) -> { a = ++a; return a; }, calls mi.perform(101), and prints the result. The output console shows the command to run the program, the output 'After increment102', and the message 'Process finished with exit code 0'.

```
1 interface MyInterface{
2     int perform(int a);
3 }
4 class FunctionalInterfaceLambdaEx {
5     public static void main(String[] args) {
6         MyInterface mi = (int a) -> {
7             a = ++a;
8             return a;
9         };
10        int res = mi.perform( a: 101);
11        System.out.println("After increment"+res);
12    }
13 }
14
```

FunctionalInterfaceLambdaEx > main()

:/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...  
After increment102  
Process finished with exit code 0

```
interface MyInterface{
    int perform(int a);
}
class FunctionalInterfaceLambdaEx {
    public static void main(String[] args) {
        MyInterface mi = (int a) -> {
            a = ++a;
            return a;
        };
        int res = mi.perform(101);
        System.out.println("After increment"+res);
    }
}
```

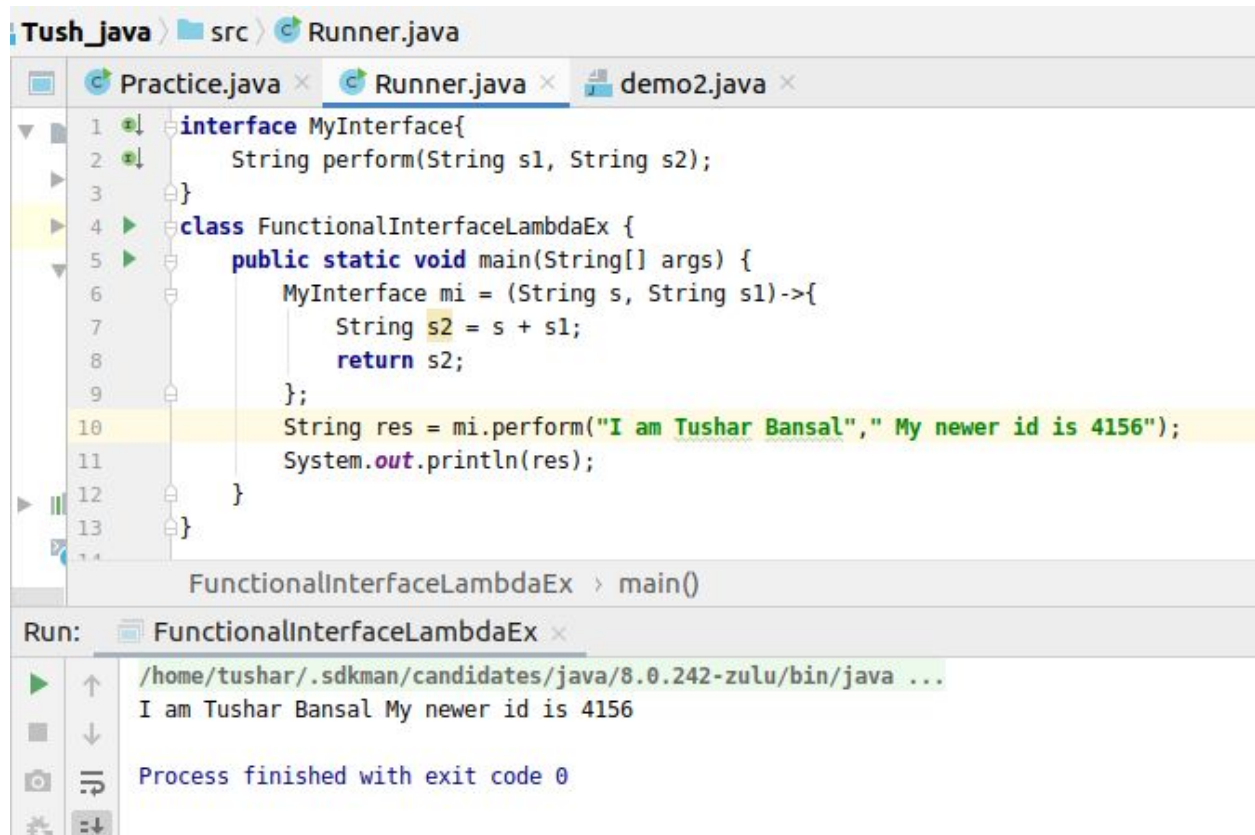
3.

```
interface MyInterface{
    String perform(String s1, String s2);
}
class FunctionalInterfaceLambdaEx {
    public static void main(String[] args) {
        MyInterface mi = (String s, String s1)->{
            String s2 = s + s1;
        };
    }
}
```

```

        return s2;
    };
    String res = mi.perform("I am Tushar Bansal"," My newer id is 4156");
    System.out.println(res);
}
}

```



4.

```

interface MyInterface{
    String perform(String s);
}
class FunctionalInterfaceLambdaEx {
    public static void main(String[] args) {
        MyInterface mi = (String s)->{
            String s1 = s.toUpperCase();
            return s1;
        };
        String res = mi.perform("Tushar");
        System.out.println(res);
    }
}

```

```
tushar_java > src > Runner.java
Practice.java x Runner.java x demo2.java x
1 interface MyInterface{
2     String perform(String s);
3 }
4 class FunctionalInterfaceLambdaEx {
5     public static void main(String[] args) {
6         MyInterface mi = (String s)->{
7             String s1 = s.toUpperCase();
8             return s1;
9         };
10        String res = mi.perform( s: "Tushar");
11        System.out.println(res);
12    }
13 }
FunctionalInterfaceLambdaEx > main()
Run: FunctionalInterfaceLambdaEx x
/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
TUSHAR
Process finished with exit code 0
```

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

```
interface Functional{
    int print(int a, int b);
}
class FunctionalEx {
    public static void main(String[] args) {
        Functional fun=(int x, int y)->{
            return (x+y);
        };
        int val = fun.print(100,20);
        System.out.println(val);
    }
}
```

```

Tush_java > src > Runner.java
Practice.java x Runner.java x demo2.java x
1 interface Functional{
2     int print(int a, int b);
3 }
4 class FunctionalEx {
5     public static void main(String[] args) {
6         Functional fun=(int x, int y)->{
7             return (x+y);
8         };
9         int val = fun.print(a: 100, b: 20);
10        System.out.println(val);
11    }
12 }
13
FunctionalEx > main()
Run: FunctionalEx x
/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java
120
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.

**Answer:**

```

interface Reference{
    void display();
}
class MethodRefernceEx {
    void add(){
        int a=7;
        int b=3;
        int c=a+b;
        int d=a-b;
        int e = a * b;
        System.out.println("Add "+c);
        System.out.println("Subtract "+d);
        System.out.println("Multiply "+e);
    }
    public static void main(String[] args) {
        Reference ref = new MethodRefernceEx();
        ref.display();
    }
}

```

```

Tush_Java > src > Runner.java
Practice.java x Runner.java x demo2.java x
1 interface Reference{
2     void display();
3 }
4 class MethodRefernceEx {
5     void add(){
6         int a=7;
7         int b=3;
8         int c=a+b;
9         int d=a-b;
10        int e = a * b;
11        System.out.println("Add "+c);
12        System.out.println("Subtract "+d);
13        System.out.println("Multiply "+e);
14    }
15    public static void main(String[] args) {
16        Reference ref = new MethodRefernceEx().add;
17        ref.display();
18    }
19 }
MethodRefernceEx > add()

Run: MethodRefernceEx x
/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Add 10
Subtract 4
Multiply 21
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 .

**Answer:**

```

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

    public Employee(String s, int i, String c) {
        System.out.println(s+" "+i+" "+c);
    }
}
interface EmployeeRef{
    public abstract Employee getEmpDetails(String n, int a, String c);
}
class EmployeeReference {
    public static void main(String[] args) {

```



```

EmployeeRef er = Employee::new;
er.getEmpDetails("Tushar Bansal",20,"Ghaziabad");
}
}

```

The screenshot shows an IDE with three tabs: Practice.java, Runner.java, and demo2.java. The Runner.java tab is active, showing the following code:

```

1  class Employee {
2      String name;
3      int age;
4      String city;
5
6      public Employee(String s, int i, String c) {
7          System.out.println(s+" "+i+" "+c);
8      }
9  }
10 interface EmployeeRef{
11     public abstract Employee getEmpDetails(String n, int a, String c);
12 }
13 class EmployeeReference {
14     public static void main(String[] args) {
15         EmployeeRef er = Employee::new;
16         er.getEmpDetails( n: "Tushar Bansal", a: 20, c: "Ghaziabad");
17     }
18 }

```

The output window shows the following text:

```

Run: EmployeeReference x
/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
Tushar Bansal 20 Ghaziabad
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

**Answer:**

1.

```

import java.util.function.Consumer;
class ConsumerInterface {

    public static void main(String[] args) {

        Consumer consumer = (i)->{

            System.out.println(i);

```

```

    };
    consumer.accept(10);
}
}

```

The screenshot shows an IDE with the following components:

- Project Explorer:** Shows a project named 'Tush\_java' with a source folder 'src' containing a file 'ConsumerInterface'.
- Editor:** Displays the code for 'ConsumerInterface.java'. The code is as follows:
 

```

import java.util.function.Consumer;
class ConsumerInterface {
    public static void main(String[] args) {
        Consumer consumer = (i)->{
            System.out.println(i);
        };
        consumer.accept(10);
    }
}

```
- Run Console:** Shows the execution of the 'ConsumerInterface' class. The command executed is:
 

```

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

```

 The output is:
 

```

10

```

 The message 'Process finished with exit code 0' is also displayed.

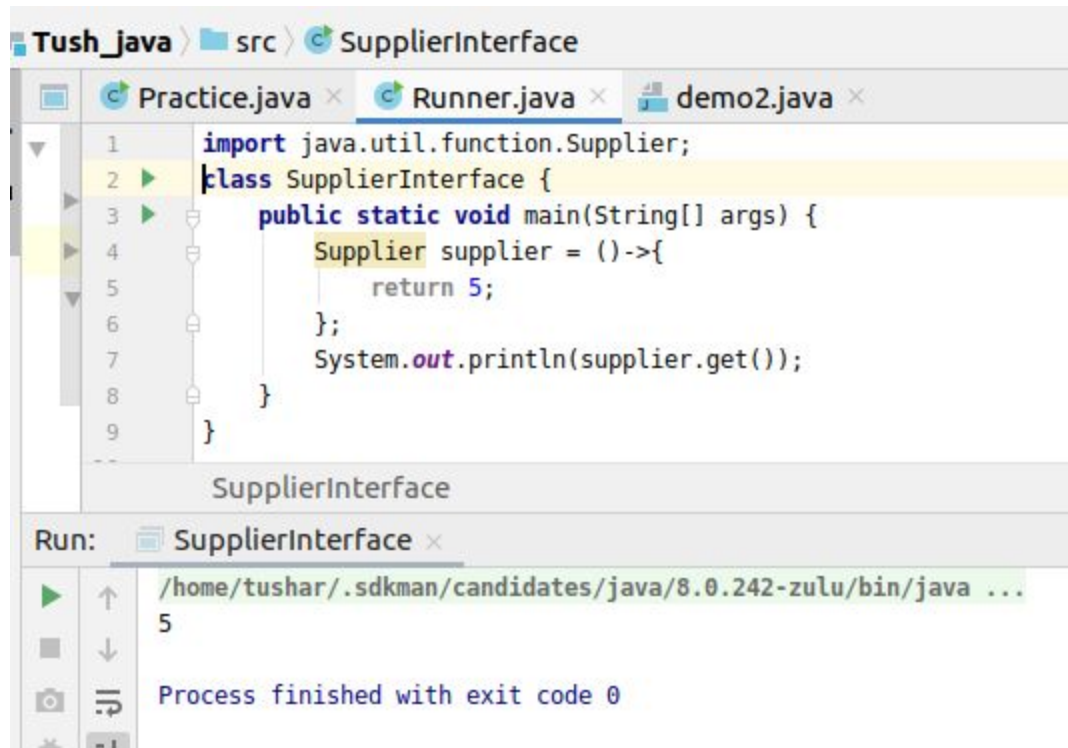
2.

```

import java.util.function.Supplier;
class SupplierInterface {
    public static void main(String[] args) {
        Supplier supplier = ()->{
            return 5;
        };
        System.out.println(supplier.get());
    }
}

```





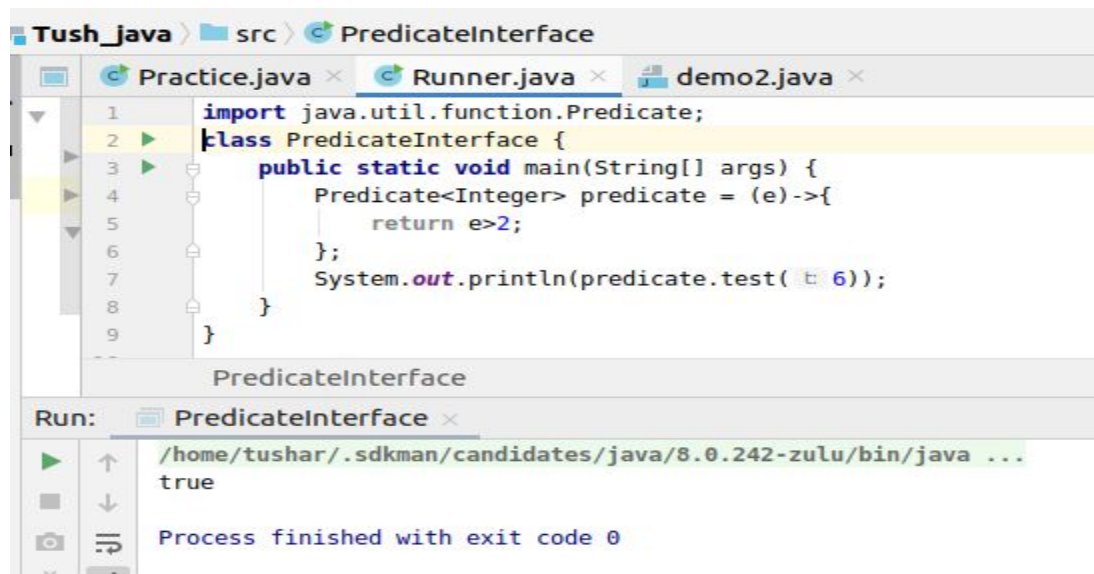
The screenshot shows an IDE window titled "Tush\_java" with a project "src" containing a class "SupplierInterface". The code in "SupplierInterface.java" is as follows:

```
1 import java.util.function.Supplier;
2 class SupplierInterface {
3     public static void main(String[] args) {
4         Supplier supplier = ()->{
5             return 5;
6         };
7         System.out.println(supplier.get());
8     }
9 }
```

The output window shows the command: `/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...` and the output: `5`. The process finished with exit code 0.

3.

```
import java.util.function.Predicate;
class PredicateInterface {
    public static void main(String[] args) {
        Predicate<Integer> predicate = (e)->{
            return e>2;
        };
        System.out.println(predicate.test(6));
    }
}
```



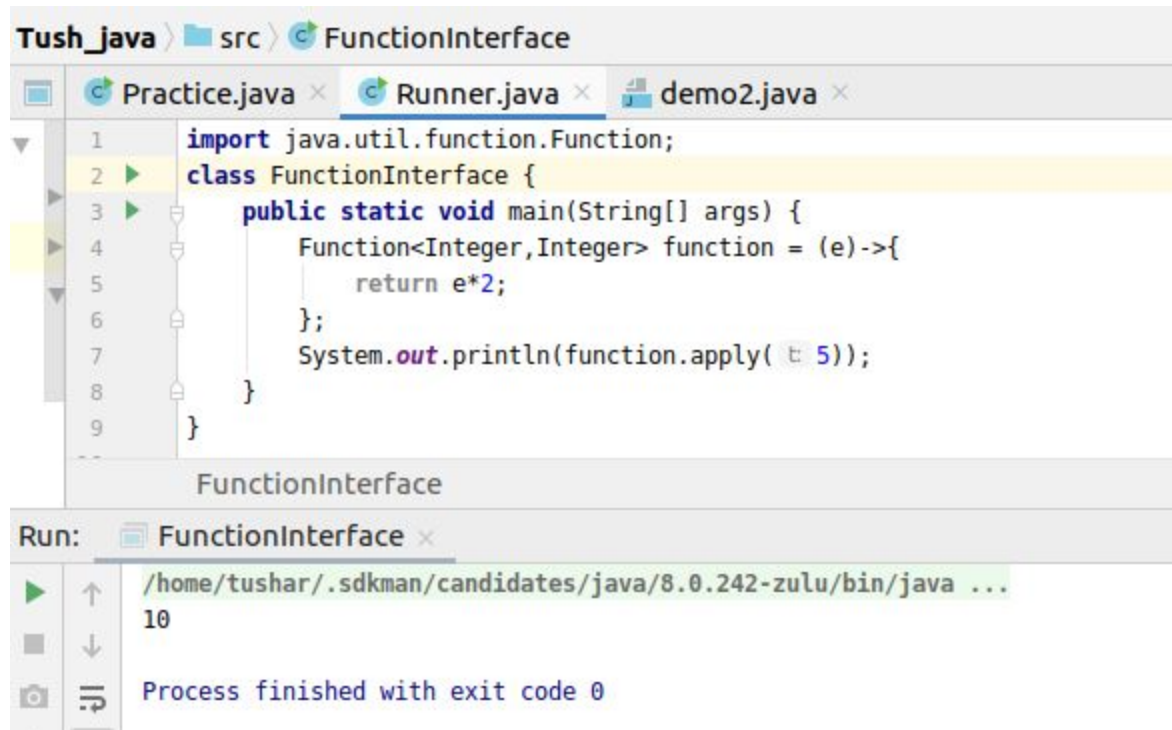
The screenshot shows an IDE window titled "Tush\_java" with a project "src" containing a class "PredicateInterface". The code in "PredicateInterface.java" is as follows:

```
1 import java.util.function.Predicate;
2 class PredicateInterface {
3     public static void main(String[] args) {
4         Predicate<Integer> predicate = (e)->{
5             return e>2;
6         };
7         System.out.println(predicate.test(6));
8     }
9 }
```

The output window shows the command: `/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...` and the output: `true`. The process finished with exit code 0.

4.

```
import java.util.function.Function;
class FunctionInterface {
    public static void main(String[] args) {
        Function<Integer,Integer> function = (e)->{
            return e*2;
        };
        System.out.println(function.apply(5));
    }
}
```

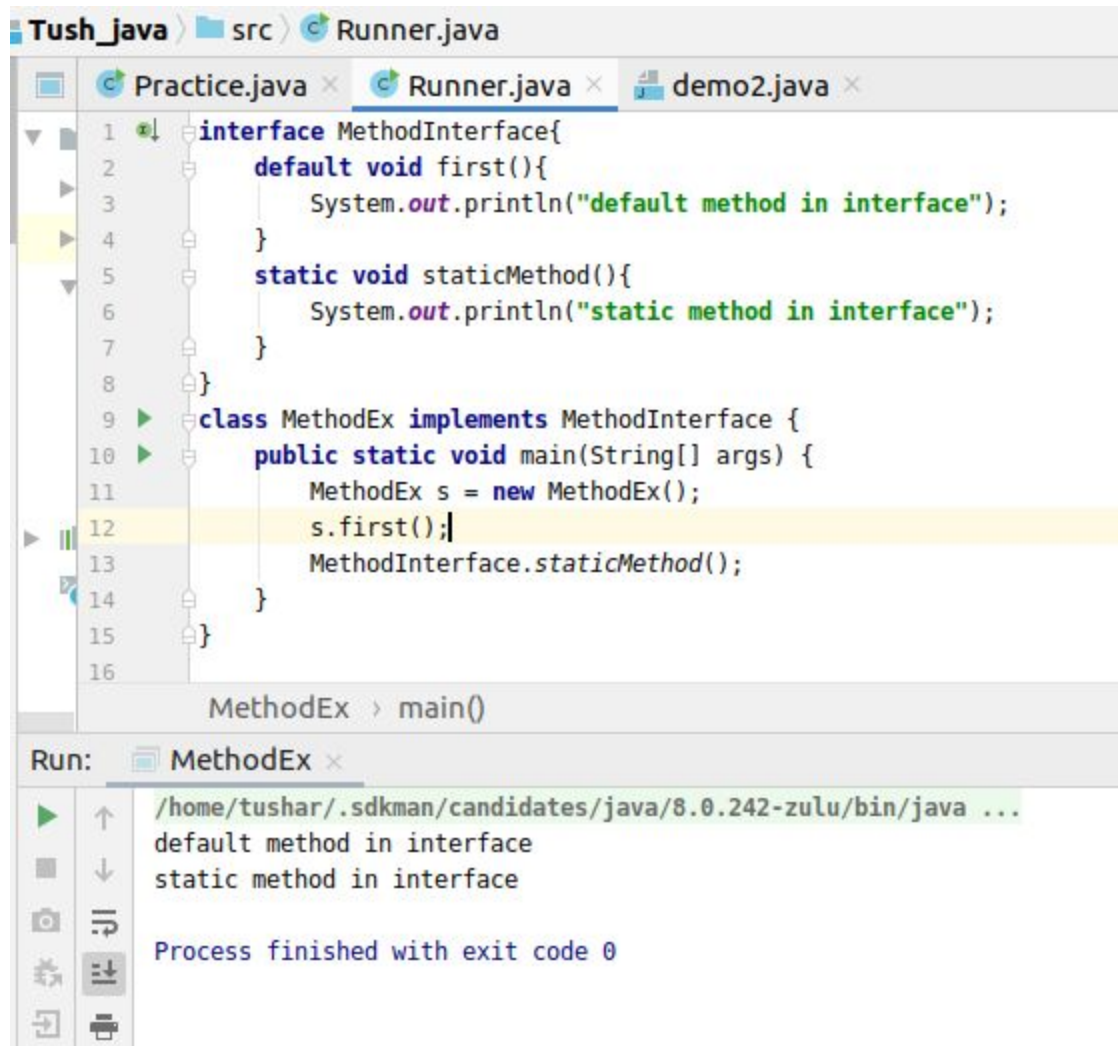


```
Tushar_java > src > FunctionInterface
Practice.java x Runner.java x demo2.java x
1 import java.util.function.Function;
2 class FunctionInterface {
3     public static void main(String[] args) {
4         Function<Integer,Integer> function = (e)->{
5             return e*2;
6         };
7         System.out.println(function.apply(5));
8     }
9 }
FunctionInterface
Run: FunctionInterface x
/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
10
Process finished with exit code 0
```

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

Answer:

```
interface MethodInterface{
    default void first(){
        System.out.println("default method in interface");
    }
    static void staticMethod(){
        System.out.println("static method in interface");
    }
}
class MethodEx implements MethodInterface {
    public static void main(String[] args) {
        MethodEx s = new MethodEx();
        s.first();
        MethodInterface.staticMethod();
    }
}
```



The screenshot shows an IDE with three tabs: Practice.java, Runner.java, and demo2.java. The Runner.java tab is active, displaying the following code:

```
1 interface MethodInterface{
2     default void first(){
3         System.out.println("default method in interface");
4     }
5     static void staticMethod(){
6         System.out.println("static method in interface");
7     }
8 }
9 class MethodEx implements MethodInterface {
10     public static void main(String[] args) {
11         MethodEx s = new MethodEx();
12         s.first();
13         MethodInterface.staticMethod();
14     }
15 }
16
```

Below the code editor, the Run tab is selected, showing the command and output:

```
Run: MethodEx x
/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
default method in interface
static method in interface
Process finished with exit code 0
```

Q7. Override the default method of the interface.

Answer:

```
interface One{
    default void print(){
        System.out.println("In one interface");
    }
    void norm();
}
interface Two{
    default void print(){
        System.out.println("In two interface");
    }
    void nor();
}
class DefaultOverrideEx implements One, Two{
    public static void main(String[] args) {
        DefaultOverrideEx def = new DefaultOverrideEx();
        def.print();
    }

    @Override
    public void nor() {
```

```

}

@Override
public void norm() {

}

@Override
public void print(){
    System.out.println("In class");
}
}

```

The screenshot shows an IDE window titled 'Tush\_java' with a project 'src' containing a file 'Runner.java'. The code defines two interfaces, 'One' and 'Two', each with a default 'print()' method and a 'norm()' method. A class 'DefaultOverrideEx' implements both interfaces and has a 'main' method that creates an instance of 'DefaultOverrideEx' and calls 'print()'. The output window shows the execution result: 'In class'.

```

Tush_java > src > Runner.java

Practice.java x Runner.java x demo2.java x

1 interface One{
2     default void print(){
3         System.out.println("In one interface");
4     }
5     void norm();
6 }
7 interface Two{
8     default void print(){
9         System.out.println("In two interface");
10    }
11    void nor();
12 }
13 class DefaultOverrideEx implements One, Two{
14     public static void main(String[] args) {
15         DefaultOverrideEx def = new DefaultOverrideEx();
16         def.print();
17     }
18 }

Run: DefaultOverrideEx x

/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
In class

Process finished with exit code 0

```

**Q8.Implement multiple inheritance with default method inside interface.**  
**Answer:**

```
Tush_java > src > Runner.java
Practice.java x Runner.java x demo2.java x
1 interface One{
2     default void print(){
3         System.out.println("In one");
4     }
5 }
6 interface Two{
7     default void print(){
8         System.out.println("In two");
9     }
10 }
11 class DefaultOverrideEx implements One, Two{
12     public static void main(String[] args) {
13         DefaultOverrideEx def = new DefaultOverrideEx();
14         def.print();
15     }
16     @Override
17     public void print(){
18         One.super.print();
19         Two.super.print();
20     }
21 }
Two > print()

Run: DefaultOverrideEx x
/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
In one
In two
Process finished with exit code 0
```

```
interface One{
    default void print(){
        System.out.println("In one");
    }
}
interface Two{
    default void print(){
        System.out.println("In two");
    }
}
class DefaultOverrideEx implements One, Two{
    public static void main(String[] args) {
        DefaultOverrideEx def = new DefaultOverrideEx();
        def.print();
    }
    @Override
```



```

    public void print(){
        One.super.print();
        Two.super.print();
    }
}

```

**Q9. Collect all the even numbers from an integer list.**

**Answer:**

```

import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
class CollectEx {
    public static void main(String[] args) {
        List<Integer> list = Arrays.asList(1,2,3,4,5,5,6,7,8,9,0);
        System.out.println(
            list
                .stream()
                .filter(e->e%2==0)
                .collect(Collectors.toList())
        );
    }
}

```

The screenshot shows an IDE with a project named 'tushar\_java' and a source folder 'src' containing a class 'CollectEx'. The code in 'CollectEx.java' is as follows:

```

1  import java.util.Arrays;
2  import java.util.List;
3  import java.util.stream.Collectors;
4  class CollectEx {
5      public static void main(String[] args) {
6          List<Integer> list = Arrays.asList(1,2,3,4,5,5,6,7,8,9,0);
7          System.out.println(
8              list
9                  .stream()
10                 .filter(e->e%2==0)
11                 .collect(Collectors.toList())
12             );
13     }
14 }
15

```

The IDE shows the execution of the 'main()' method of 'CollectEx'. The output is:

```

Run: CollectEx
/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
[2, 4, 6, 8, 0]
Process finished with exit code 0

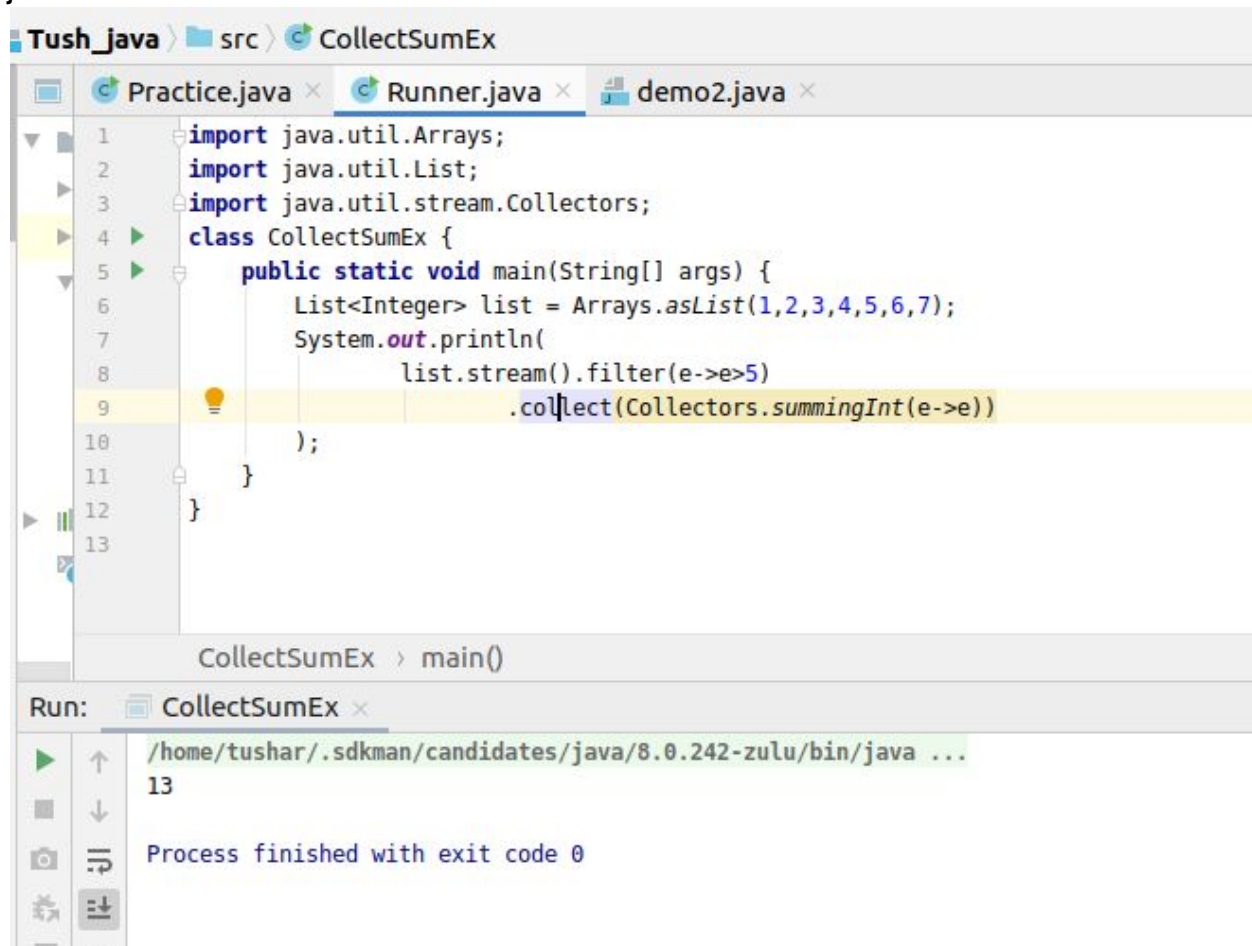
```



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

**Answer:**

```
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
class CollectSumEx {
    public static void main(String[] args) {
        List<Integer> list = Arrays.asList(1,2,3,4,5,6,7);
        System.out.println(
            list.stream().filter(e->e>5)
                .collect(Collectors.summingInt(e->e))
        );
    }
}
```



The screenshot shows an IDE with a project named 'Tush\_java' and a source folder 'src'. Inside 'src', there is a file 'CollectSumEx'. The code in 'CollectSumEx' is as follows:

```
1 import java.util.Arrays;
2 import java.util.List;
3 import java.util.stream.Collectors;
4 class CollectSumEx {
5     public static void main(String[] args) {
6         List<Integer> list = Arrays.asList(1,2,3,4,5,6,7);
7         System.out.println(
8             list.stream().filter(e->e>5)
9                 .collect(Collectors.summingInt(e->e))
10        );
11    }
12 }
13
```

The IDE also shows a 'Run' window for 'CollectSumEx'. The command executed is:

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

The output is:

```
13
```

The process finished with exit code 0.

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

**Answer:**

```
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
class CollectAverageEx {
    public static void main(String[] args) {
        List<Integer> list = Arrays.asList(1,2,3,4,5);
        System.out.println(
            list.stream()
                .collect(Collectors.averagingInt(e->e*2))
        );
    }
}
```

```

    );
}
}

```

The screenshot shows an IDE with a project named 'Tush\_java' and a source folder 'src'. The file 'CollectAverageEx' is open, showing the following code:

```

1  import java.util.Arrays;
2  import java.util.List;
3  import java.util.stream.Collectors;
4  class CollectAverageEx {
5      public static void main(String[] args) {
6          List<Integer> list = Arrays.asList(1,2,3,4,5);
7          System.out.println(
8              list.stream()
9                  .collect(Collectors.averagingInt(e->e*2))
10             );
11      }
12  }
13

```

Below the code editor, the 'Run' tab is active, showing the command executed: `/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...` and the output: `6.0`. The status bar indicates 'Process finished with exit code 0'.

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

**Answer:**

```

import java.util.Arrays;
import java.util.List;
class FirstEvenEx {
    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()
        );
    }
}

```

Tush\_java > src > FirstEvenEx

Practice.java x Runner.java x demo2.java x

```
1 import java.util.Arrays;
2 import java.util.List;
3 class FirstEvenEx {
4     public static void main(String[] args) {
5         List<Integer> list = Arrays.asList(1,2,3,4,5,6,7,8,9,10);
6         System.out.println(
7             list
8                 .stream()
9                 .filter(e->e>3)
10                .filter(e->e%2==0)
11                .findFirst()
12        );
13    }
14 }
```

FirstEvenEx

Run: FirstEvenEx x

/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...  
Optional[4]

Process finished with exit code 0