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

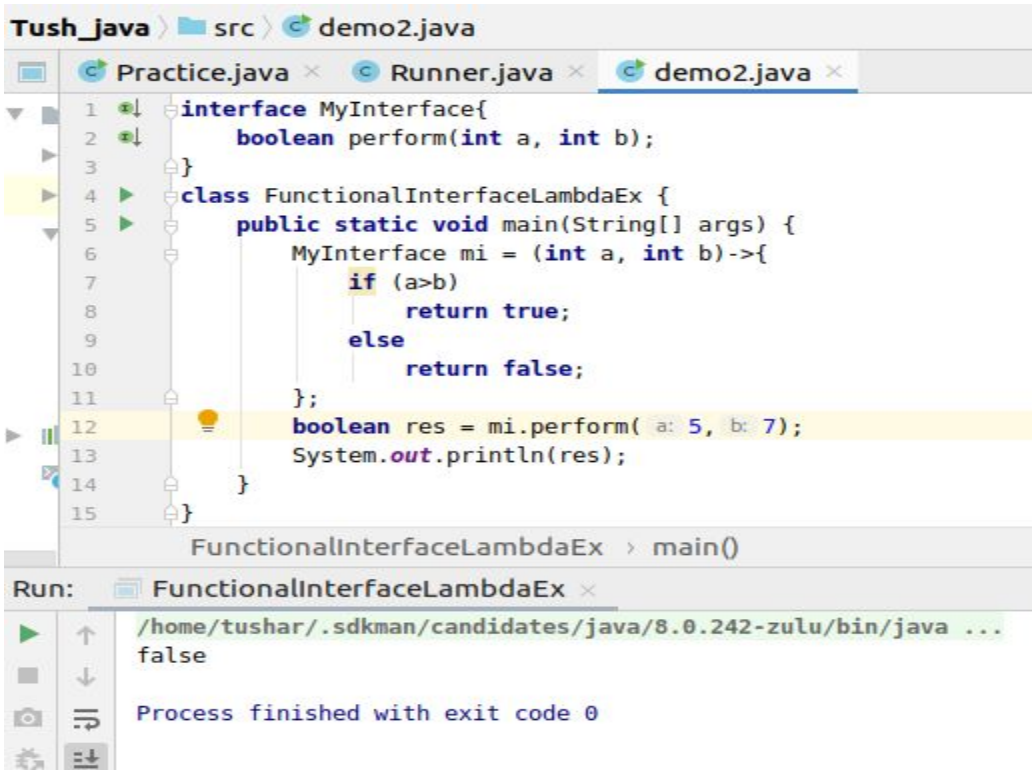
(2) Increment the number by 1 and return incremented value . Parameter (int) Return int

(3) Concatination of 2 string . Parameter (String , String) Return (String)

(4) Convert a string to uppercase and return . Parameter (String) Return (String)

Answers:

1.




```
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.

Practice.java | Runner.java | demo2.java

```java
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( a: 101);
        System.out.println("After increment"+res);
    }
}
```

FunctionalInterfaceLambdaEx › main()

FunctionalInterfaceLambdaEx

```
/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
After increment102

Process finished with exit code 0
```

```java
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.

```java
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);
  }
}
```

Tush_java 〉 src 〉 Runner.java

Practice.java | Runner.java | demo2.java

```
1  interface MyInterface{
2      String perform(String s1, String s2);
3  }
4  class FunctionalInterfaceLambdaEx {
5      public static void main(String[] args) {
6          MyInterface mi = (String s, String s1)->{
7              String s2 = s + s1;
8              return s2;
9          };
10         String res = mi.perform("I am Tushar Bansal"," My newer id is 4156");
11         System.out.println(res);
12     }
13 }
```

FunctionalInterfaceLambdaEx 〉 main()

Run: FunctionalInterfaceLambdaEx

```
/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
I am Tushar Bansal My newer id is 4156

Process finished with exit code 0
```

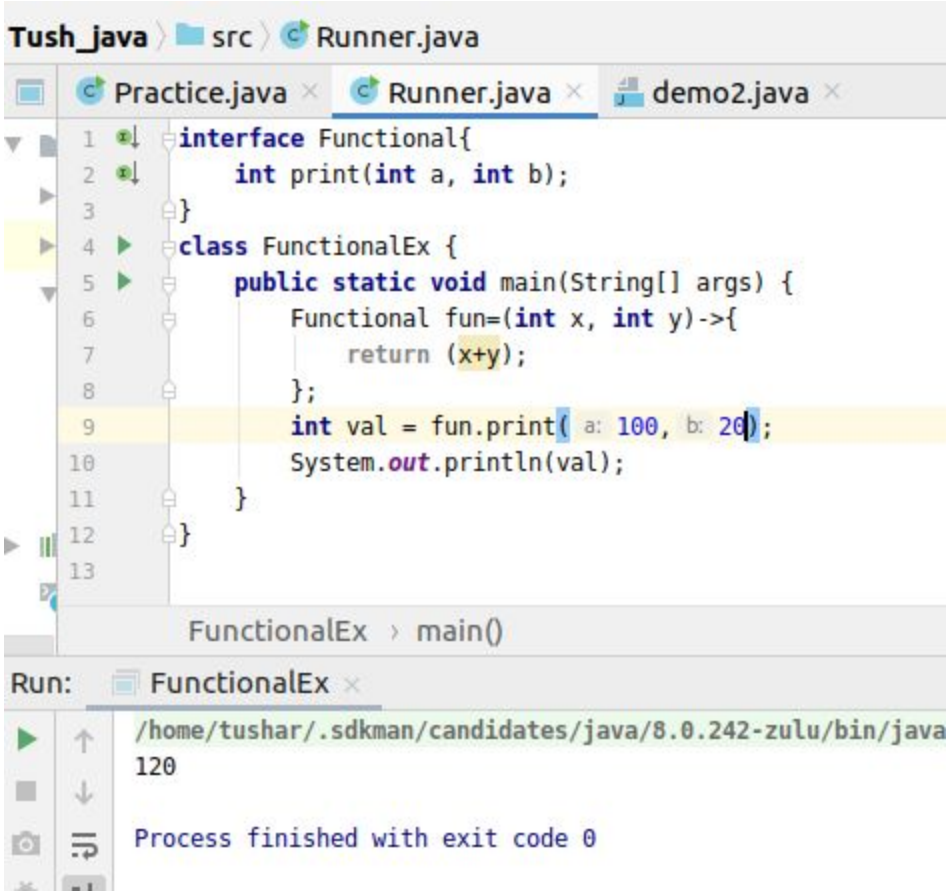
4.

```
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);
  }
}
```

```
rush_java > src > Runner.java
Practice.java   Runner.java   demo2.java
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
/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);
  }
}
```



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()::add;
    ref.display();
  }
}
```

Tush_java 〉 src 〉 Runner.java

Practice.java | Runner.java | demo2.java

```java
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()::add;
        ref.display();
    }
}
```

MethodRefernceEx 〉 add()

Run: MethodRefernceEx

```
/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:

```java
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");
  }
}
```

Tush_java 〉 src 〉 Runner.java

Practice.java | Runner.java | demo2.java

```
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  }
```

EmployeeReference 〉 main()

Run: EmployeeReference

```
/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);
    }
}
```

2.

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

Tush_java 〉 src 〉 SupplierInterface

Practice.java | Runner.java | demo2.java

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

SupplierInterface

Run: SupplierInterface

```
/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
5

Process finished with exit code 0
```

3.

```java
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));
  }
}
```

Tush_java 〉 src 〉 PredicateInterface

Practice.java | Runner.java | demo2.java

```java
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( t: 6));
    }
}
```

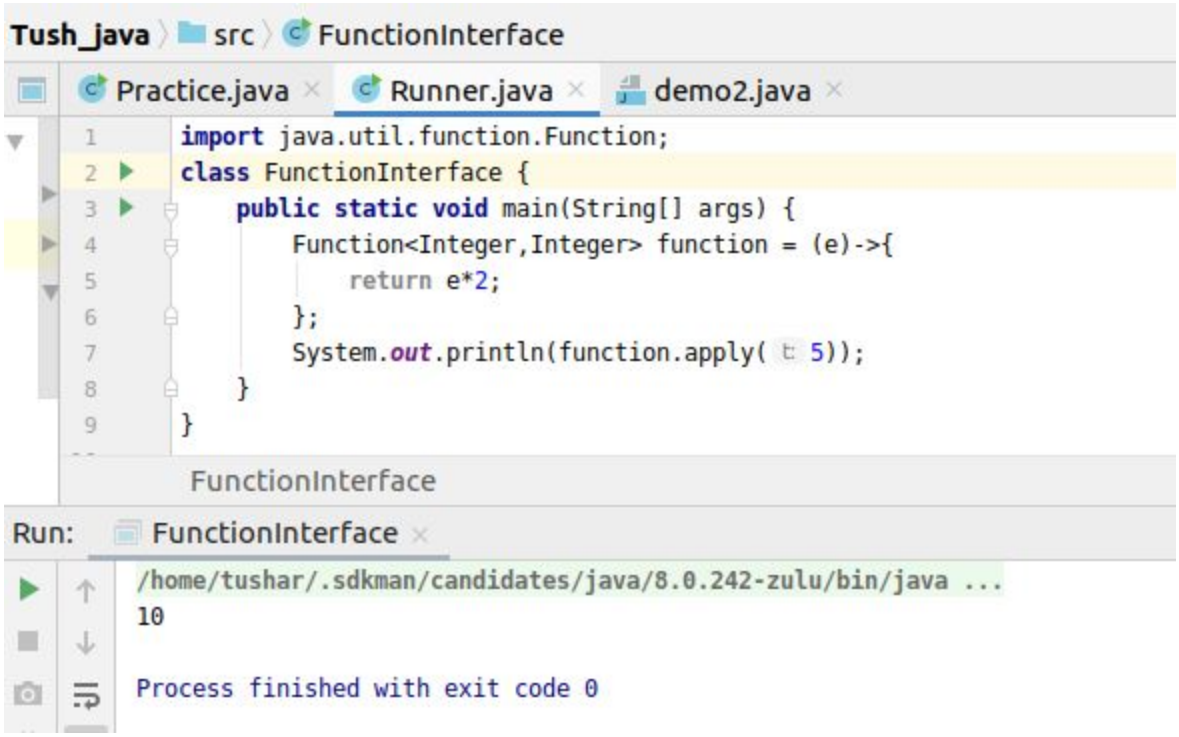
PredicateInterface

Run: PredicateInterface

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

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));
  }
}
```



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();
  }
}
```

Tush_java > src > Runner.java

Practice.java | Runner.java | demo2.java

```java
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();
    }
}
```

MethodEx > main()

Run: MethodEx

```
/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:

```java
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");
    }
}
```

Tush_java > src > Runner.java

Practice.java | Runner.java | demo2.java

```
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();
```

Run: DefaultOverrideEx

```
/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 | Runner.java | demo2.java

```java
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();
    }
}
```

Two > print()

Run: DefaultOverrideEx

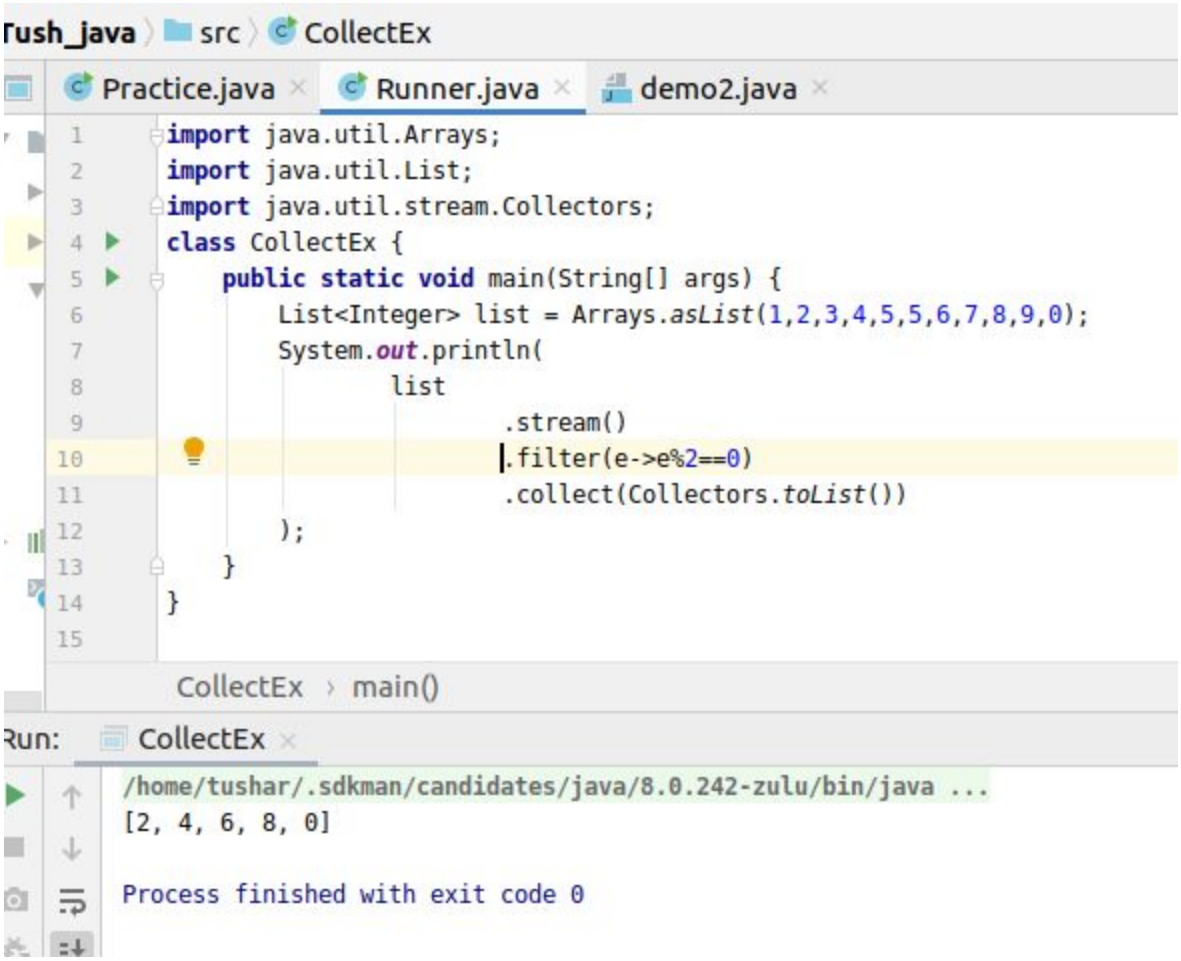
```
/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
In one
In two

Process finished with exit code 0
```

```java
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())
    );
  }
}
```



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))
     );
  }
}
```

Tush_java 〉 src 〉 CollectSumEx

Practice.java | Runner.java | demo2.java

```
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))
        );
    }
}
```

CollectSumEx 〉 main()

Run: CollectSumEx

```
/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
13

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))
```

```
    );
  }
}
```

Tush_java 〉 src 〉 CollectAverageEx

Practice.java | Runner.java | demo2.java

```
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))
        );
    }
}
```

Run: CollectAverageEx

```
/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
6.0

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 × Runner.java × demo2.java ×

```java
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()
        );
    }
}
```

FirstEvenEx

Run: FirstEvenEx ×

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

Process finished with exit code 0
```