# Session: Java 8 Features

# Assignment

1. 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

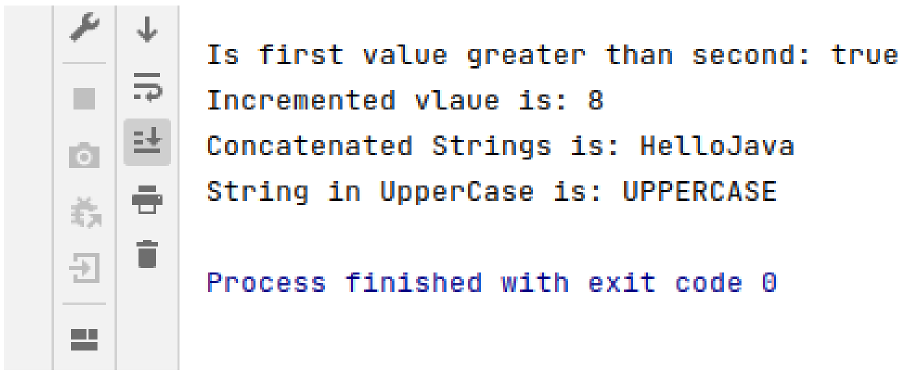
   (3) Concatenation of 2 string
   Parameter (String , String ) Return (String)

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

   **CODE**

```
import java.util.Locale;

import java.util.function.*;

public class Ques1 {

   public static void main(String[] args) {

      BiFunction<Integer, Integer, Boolean> isGreater = (x, y) -> x > y;

      Function<Integer, Integer> incrementByOne = x -> x + 1;

      BiFunction<String, String, String> stringConcate = (x, y) -> x + y;

      Function<String,String> changeCase = str -> str.toUpperCase();


      System.out.println("\nIs first value greater than second: "+isGreater.apply(7,5));

      System.out.println("Incremented vlaue is: "+ incrementByOne.apply(7));

      System.out.println("Concatenated Strings is: "+stringConcate.apply("Hello", "Java"));

      System.out.println("String in UpperCase is: "+ changeCase.apply("uppercase"));


   }

}
```

**OUTPUT**



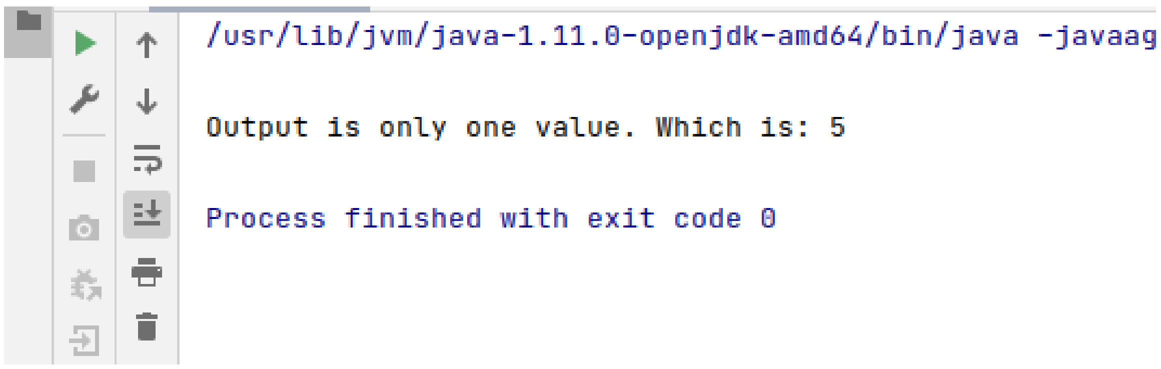
```
Is first value greater than second: true
Incremented vlaue is: 8
Concatenated Strings is: HelloJava
String in UpperCase is: UPPERCASE

Process finished with exit code 0
```

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

**CODE**

```
        import java.util.function.BiFunction;
        import java.util.function.Function;

        public class Ques2 {
           public static void main(String[] args) {

              BiFunction<Integer, Integer, Integer> result = (x,y) -> x;
              System.out.println("\nOutput is only one value. Which is: "+ result.apply(5,3));
           }
}
```

**OUTPUT**



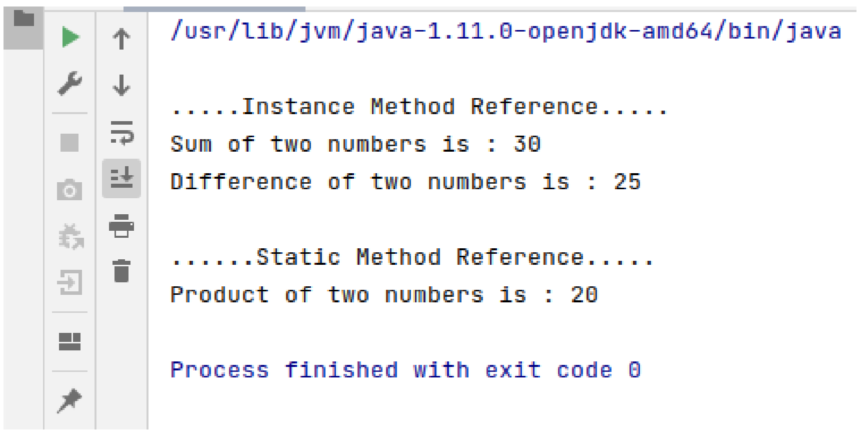
```
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -javaag

Output is only one value. Which is: 5

Process finished with exit code 0
```

3. Using (instance) Method reference created and apply add and subtract method and using (Static) Method reference create and apply multiplication method for the functional interface created.

**CODE**

```
import java.util.function.BiFunction;

public class Ques3 {
    public static void main(String[] args) {
        Ques3 obj1 = new Ques3();
        System.out.println("\n.....Instance Method Reference.....");
        BiFunction<Integer, Integer, Integer> sum = obj1::addition;
        System.out.println("Sum of two numbers is : "+sum.apply(5,25));

        BiFunction<Integer, Integer, Integer> diff = obj1::difference;
        System.out.println("Difference of two numbers is : "+diff.apply(5,30));

        System.out.println("\n......Static Method Reference.....");
        BiFunction<Integer, Integer, Integer> prod = Ques3::multiplication;
        System.out.println("Product of two numbers is : "+prod.apply(5,4));

    }
    public int addition(int num1, int num2)
    {
        return num1 + num2;
    }
    public int difference(int num1, int num2)
    {
        return num2 - num1;
    }
    public static int multiplication(int num1, int num2)
    {
        return num1 * num2;
    }
}
```

## OUTPUT



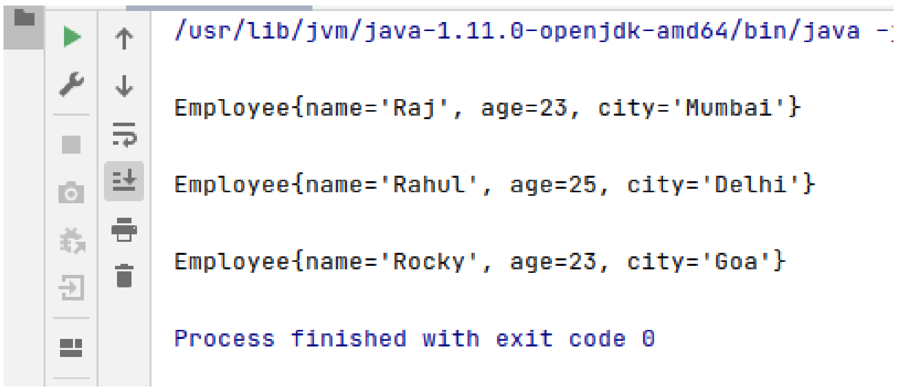
```
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java

.....Instance Method Reference.....
Sum of two numbers is : 30
Difference of two numbers is : 25

......Static Method Reference.....
Product of two numbers is : 20

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

**CODE**

```
interface empInterface
{
     Employee getEmployee(String name, int age, String city);
}

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

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

   @Override
   public String toString() {
      return "\nEmployee{" +
            "name='" + name + '\'' +
            ", age=" + age +
            ", city='" + city + '\'' +
            '}';
   }
}
public class Ques4 {
   public static void main(String[] args) {

      empInterface emp = Employee::new;
      System.out.println(emp.getEmployee("Raj", 23, "Mumbai"));
      System.out.println(emp.getEmployee("Rahul", 25, "Delhi"));
      System.out.println(emp.getEmployee("Rocky", 23, "Goa"));
   }
}
```

**OUTPUT**



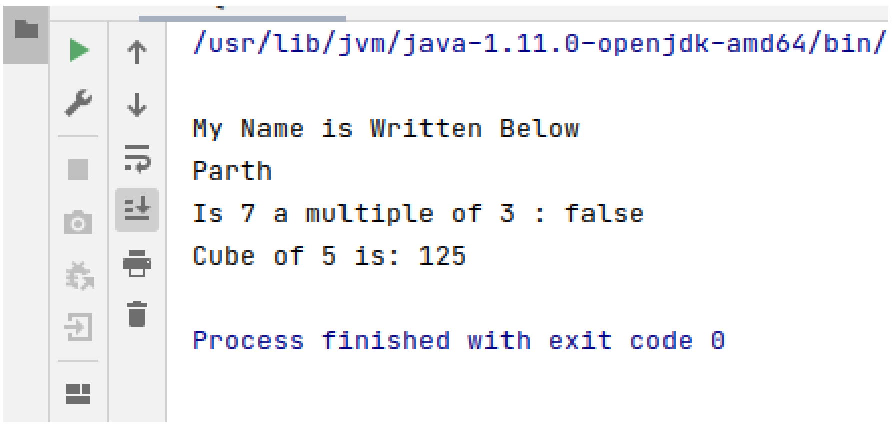
```
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -

Employee{name='Raj', age=23, city='Mumbai'}

Employee{name='Rahul', age=25, city='Delhi'}

Employee{name='Rocky', age=23, city='Goa'}

Process finished with exit code 0
```

5. Implement following functional interfaces from java.util.function using lambdas
   - Consumer
   - Supplier
   - Predicate
   - Function

**CODE**

```
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) {

        Consumer<String> name = str -> System.out.println(str);
        Supplier<String> str = () -> "Parth";
        Predicate<Integer> isMultipleofThree = x -> (x % 3 == 0);
        Function<Integer, Integer> cube = x -> (x * x * x);


        name.accept("\nMy Name is Written Below");
        System.out.println(str.get());
        System.out.println("Is 7 a multiple of 3 : "+isMultipleofThree.test(7));
        System.out.println("Cube of 5 is: "+cube.apply(5));
    }
}
```

**OUTPUT**



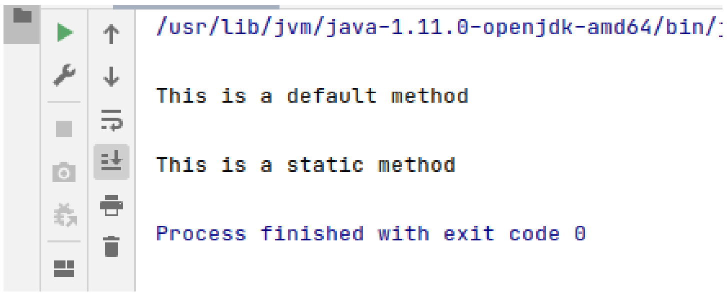
```
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/

My Name is Written Below
Parth
Is 7 a multiple of 3 : false
Cube of 5 is: 125

Process finished with exit code 0
```

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

**CODE**

```
interface dummy {
   default void dummyDefault() {
      System.out.println("\nThis is a default method");
   }

   static void dummyStatic() {
      System.out.println("\nThis is a static method");
   }
}

public class Ques6 implements dummy {
   public static void main(String[] args) {

      Ques6 obj = new Ques6();
      obj.dummyDefault();
      dummy.dummyStatic();

   }
}
```

**OUTPUT**



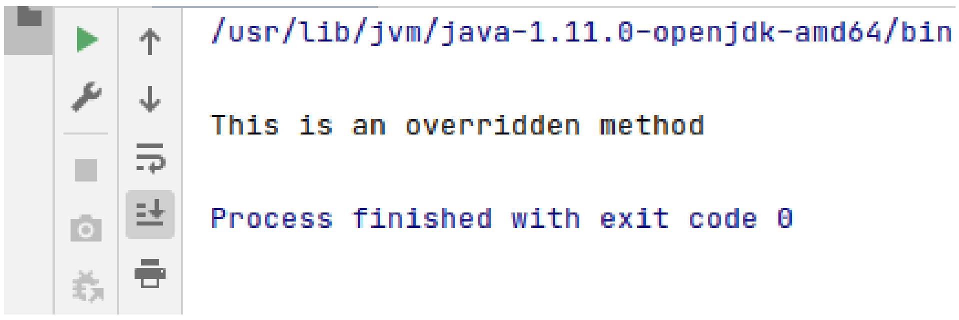
```
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/

This is a default method

This is a static method

Process finished with exit code 0
```

7. Override the default method of the interface.

**CODE**

```
interface demo
{
   default void demoDefault()
   {
      System.out.println("\nThis is a default method for this interface");
   }
}
public class Ques7 implements demo{
   @Override
   public void demoDefault() {

      System.out.println("\nThis is an overridden method");
   }

   public static void main(String[] args) {

      Ques7 obj1 = new Ques7();
      obj1.demoDefault();

   }
}
```

**OUTPUT**



```
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin

This is an overridden method

Process finished with exit code 0
```

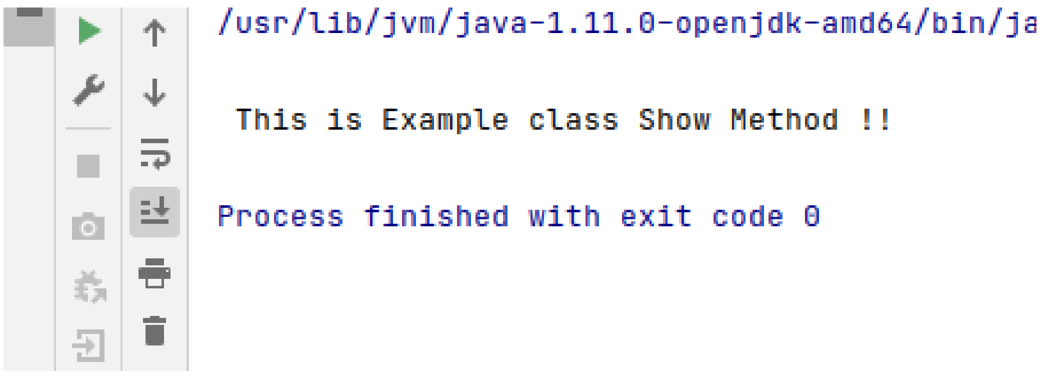
8. Implement multiple inheritance with default method inside interface.

**CODE**

```
package com.company;

interface first
{
   default void show()
   {
      System.out.println("\n This is First Interface Show Method !!");
```

```
        }
    }

    interface second
    {
        default void show()
        {
            System.out.println("\n This is Second Interface Show Method !!");
        }
    }

    class Example implements first, second
    {

        @Override
        public void show() {
            System.out.println("\n This is Example class Show Method !!");
        }
    }
    public class Ques8 {
        public static void main(String[] args) {
            Example ex = new Example();
            ex.show();
        }
}
```

**OUTPUT**



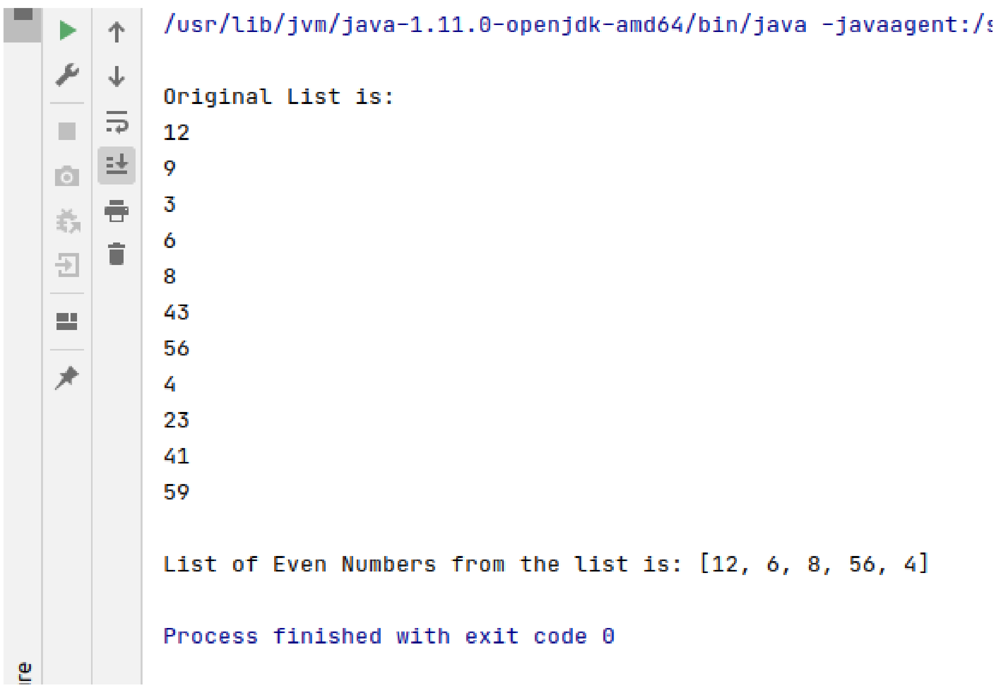
```
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/ja

 This is Example class Show Method !!

Process finished with exit code 0
```

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

## CODE

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

public class Ques9 {
    public static void main(String[] args) {
        List<Integer> numbers = List.of(12,9,3,6,8,43,56,4,23,41,59);
        System.out.println("\nOriginal List is: ");
        numbers.stream().forEach(System.out::println);
        List<Integer> evenNumbers = numbers.stream()
                                    .filter(x -> x % 2 == 0)
                                    .collect(Collectors.toList());
        System.out.println("\nList of Even Numbers from the list is: "+evenNumbers);
    }
}
```

## OUTPUT



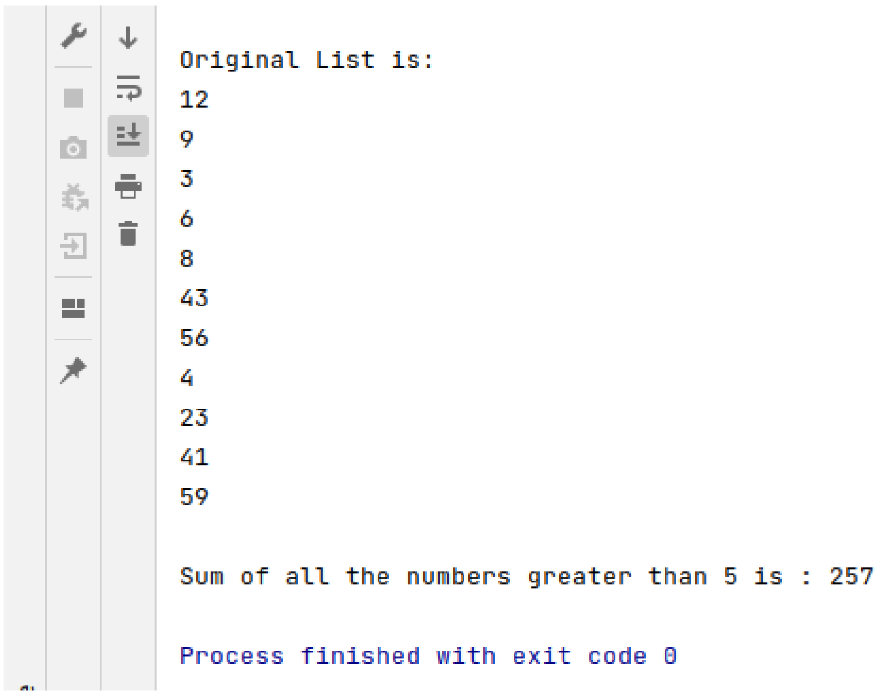
```
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -javaagent:/

Original List is:
12
9
3
6
8
43
56
4
23
41
59

List of Even Numbers from the list is: [12, 6, 8, 56, 4]

Process finished with exit code 0
```

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

## CODE

```
import java.util.List;

public class Ques10 {
    public static void main(String[] args) {

        List<Integer> numbers = List.of(12,9,3,6,8,43,56,4,23,41,59);
        System.out.println("\nOriginal List is: ");
        numbers.stream().forEach(System.out::println);
        System.out.println("\nSum of all the numbers greater than 5 is : "+numbers.stream()
                .filter(x -> x > 5)
                .reduce(0, Integer::sum));
    }
}
```

## OUTPUT



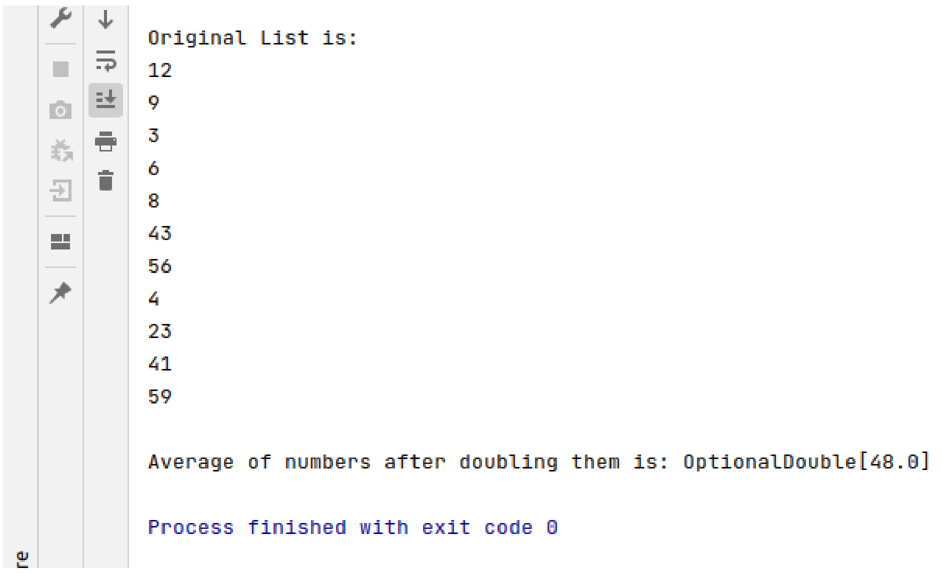
```

Original List is:
12
9
3
6
8
43
56
4
23
41
59

Sum of all the numbers greater than 5 is : 257

Process finished with exit code 0
```

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

**CODE**

```
import java.util.List;

public class Ques11 {
   public static void main(String[] args) {
      List<Integer> number = List.of(12,9,3,6,8,43,56,4,23,41,59);
      System.out.println("\nOriginal List is: ");
      number.stream().forEach(System.out::println);
      System.out.println("\nAverage of numbers after doubling them is: "+number.stream()
            .mapToDouble( x -> x + x)
            .average());
   }
}
```

**OUTPUT**



```
Original List is:
12
9
3
6
8
43
56
4
23
41
59

Average of numbers after doubling them is: OptionalDouble[48.0]

Process finished with exit code 0
```

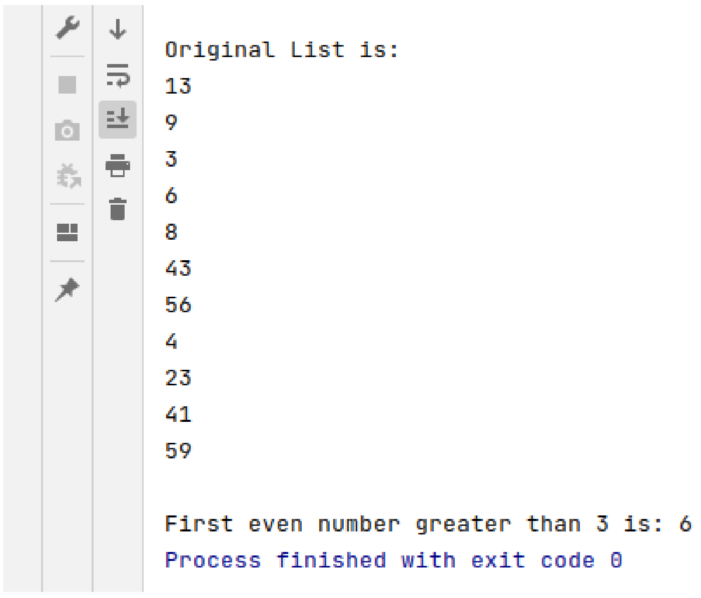
12. Find the first even number in the integer list which is greater than 3.

**CODE**

```
import java.util.List;

public class Ques12 {

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

```
                List<Integer> number = List.of(13,9,3,6,8,43,56,4,23,41,59);
                System.out.println("\nOriginal List is: ");
                number.stream().forEach(System.out::println);
                System.out.print("\nFirst even number greater than 3 is: ");
                number.stream()
                        .filter(x -> x % 2 == 0 && x >3).limit(1)
                        .forEach(System.out::print);
        }
}
```

## OUTPUT



```
Original List is:
13
9
3
6
8
43
56
4
23
41
59

First even number greater than 3 is: 6
Process finished with exit code 0
```