JAVA FEATURES

- 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)

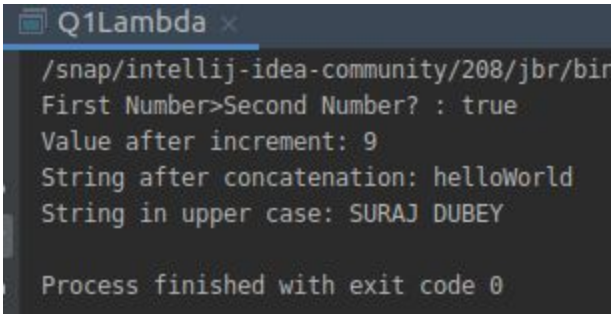
Code :

```
package com.JavaFeatures;

//part1
interface greaterNumber
{
  boolean findGreater(int a,int b);
}
//part2
interface IncrementNumber
{
  int incrementNumber(int a);
}
//part3
interface ConcatString
{
  String Concat(String s1,String s2);
}
//part4
interface ToUpper
{
  String toUpper(String s);
}
public class Q1Lambda {
  public static void main(String[] args) {
    //part1
    greaterNumber gn = (int x, int y) -> {
      if (x > y)
        return true;
      else
        return false;
    };
    System.out.println("First Number>Second Number? : " + gn.findGreater(6, 2));
    //part2
    IncrementNumber in = x -> {
```

```
        return x + 1;
    };
    System.out.println("Value after increment: " + in.incrementNumber(8));
    //part3
    ConcatString cs = (String s1, String s2) -> s1.concat(s2);
    System.out.println("String after concatenation: " + cs.Concat("hello", "World"));
    //part4
    ToUpper tu = s -> {
        return s.toUpperCase();
    };
    System.out.println("String in upper case: "+tu.toUpper("suraj dubey"));
  }
}
```

Output:



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

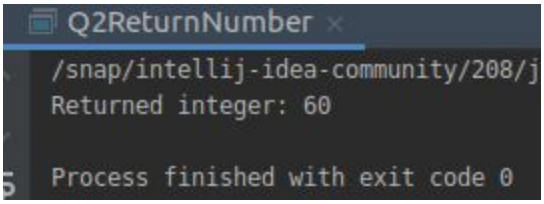
Code :

```
package com.JavaFeatures;

@FunctionalInterface
interface Number
{
   int Num(int a,int b);
}
public class Q2ReturnNumber {
   public static void main(String[] args) {
      Number number=(int a,int b)->{
         if(a>b)
            return a;
         else
            return b;
      };
      System.out.println("Returned integer: "+number.Num(60,45));
   }
```

```
}
```

Output:



- 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.

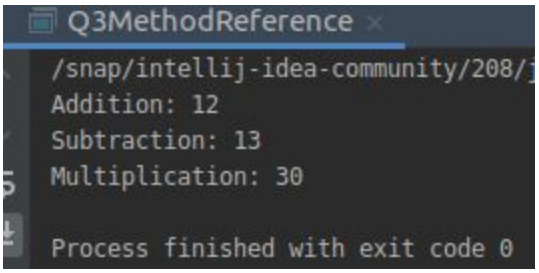
Code :

```
package com.JavaFeatures;

interface addNumber{
    void add(int a,int b);
}
interface subtractNumber{
    void subtract(int a,int b);
}
interface multiplyNumber{
    void multiply(int a);
}
public class Q3MethodReference {
    void addMethod(int a,int b)
    {
        System.out.println("Addition: "+(a+b));
    }
    void subtractMethod(int a,int b)
    {
        System.out.println("Subtraction: "+(a-b));
    }
    static void multiplyMethod(int a)
    {
        System.out.println("Multiplication: "+(a*5));
    }
    public static void main(String[] args) {
        addNumber addNum= new Q3MethodReference()::addMethod;
```

```
        addNum.add(4,8);
        subtractNumber subtractNum=new Q3MethodReference()::subtractMethod;
        subtractNum.subtract(20,7);
        multiplyNumber multiplyNum=Q3MethodReference::multiplyMethod;
        multiplyNum.multiply(6);
    }
}
```

Output:



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

Code :

Employee Class

```
package com.JavaFeatures.Q4Employee;

interface ConstRef{
    public Employee getEmp(String name,int age,String city);
}
public 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;
    }
```

```
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Integer getAge() {
        return age;
    }

    public void setAge(Integer age) {
        this.age = age;
    }

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
    }

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

Main Class

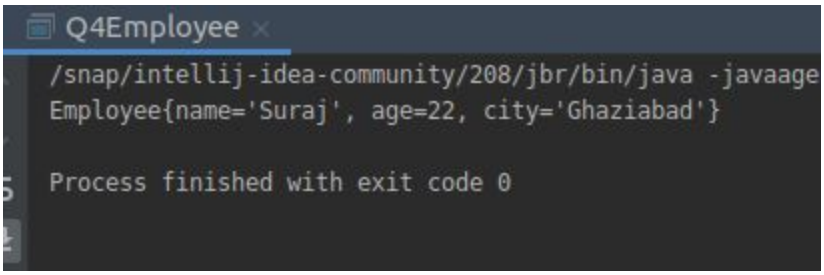
```
package com.JavaFeatures.Q4Employee;

public class Q4Employee {
    public static void main(String[] args) {
        ConstRef constRef=Employee::new;
        Employee e1=constRef.getEmp("Suraj",22,"Ghaziabad");
        System.out.println(e1);
```

```
    }
}
package com.JavaFeatures.Q4Employee;

public class Q4Employee {
    public static void main(String[] args) {
        ConstRef constRef=Employee::new;
        Employee e1=constRef.getEmp("Suraj",22,"Ghaziabad");
        System.out.println(e1);
    }
}
```

Output:



- Implement following functional interfaces from java.util.function using lambdas:
    - (1) Consumer
    - (2) Supplier
    - (3) Predicate
    - (4) Function

Code :

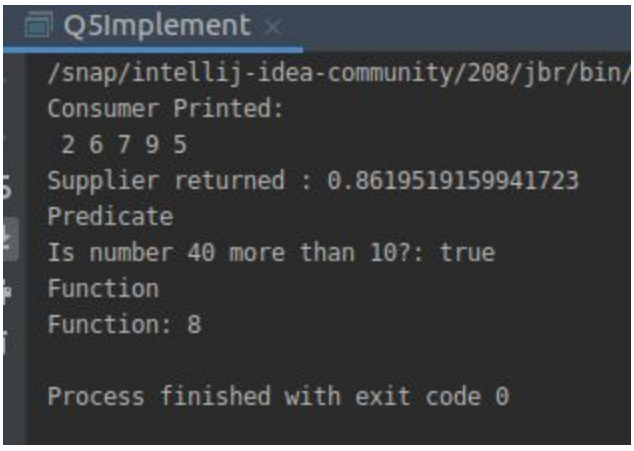
```
package com.JavaFeatures;

import java.util.Arrays;
import java.util.List;
import java.util.function.Consumer;
import java.util.function.Function;
import java.util.function.Predicate;
import java.util.function.Supplier;
public class Q5Implement {
    public static void printList(List<Integer> list1,Consumer<Integer> con)
    {
```

```
        for(int i: list1)
            con.accept(i);
    }
    public static void main(String[] args) {
        //part1
        System.out.println("Consumer Printed: " );
        Consumer<Integer> consumer=i-> System.out.print(" "+i);
        List<Integer> list= Arrays.asList(2,6,7,9,5);
        printList(list,consumer);
        System.out.println();
        //part2
        Supplier<Double> supplier=()->Math.random();
        System.out.println("Supplier returned : "+supplier.get());
        //part3
        System.out.println("Predicate");
        Predicate<Integer> predicate=i->(i>10);
        System.out.println("Is number 40 more than 10?: "+predicate.test(40));
        //part 4
        System.out.println("Function");
        Function<Integer,Integer> add = x -> x + 1;
        Integer two = add.apply(7);
        System.out.println("Function: "+two);
    }

}
```

Output:



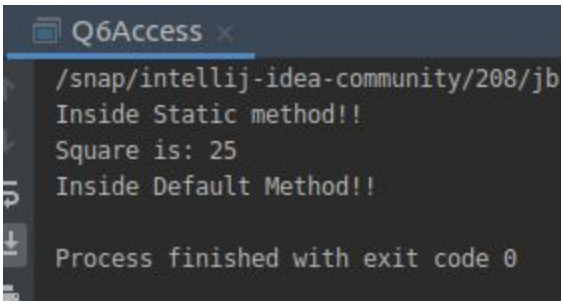
- Create and access default and static method of an interface.

Code :

```
package com.JavaFeatures;

interface TestInterface
{
   static void square(int a)
   {
      System.out.println("Inside Static method!!");
      System.out.println("Square is: "+a*a);
   }
   default void show()
   {
      System.out.println("Inside Default Method!!");
   }
}
public class Q6Access implements TestInterface{
   public static void main(String[] args) {
      Q6Access q6=new Q6Access();
      TestInterface.square(5);
      q6.show();
   }
}
```

Output:



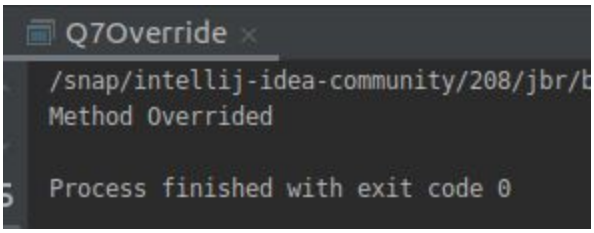
- Override the default method of the interface.

Code :

```
package com.JavaFeatures;

interface Myinterface
{
   default void show()
   {
      System.out.println("Default interface of interface1");
   }
}
public class Q7Override implements Myinterface{
   public void show()
   {
      System.out.println("Method Overrided");
   }
   public static void main(String[] args) {
      Q7Override q7=new Q7Override();
      q7.show();
   }
}
```

Output:



- Implement multiple inheritance with default method inside interface.

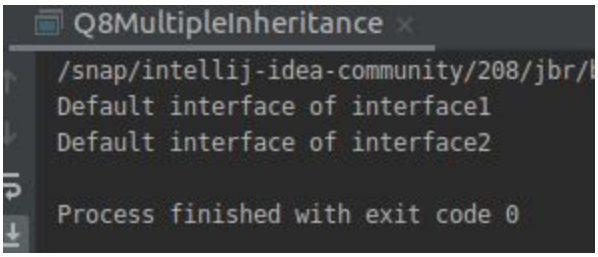
Code :

```
package com.JavaFeatures;

interface interface1
{
   default void show()
   {
```

```
        System.out.println("Default interface of interface1");
    }
}
interface interface2
{
    default void show()
    {
        System.out.println("Default interface of interface2");
    }
}
public class Q8MultipleInheritance implements interface1,interface2{
    public void show()
    {
        interface1.super.show();
        interface2.super.show();
    }
    public static void main(String[] args) {
        Q8MultipleInheritance q8=new Q8MultipleInheritance();
        q8.show();
    }
}
```

Output:



- Collect all the even numbers from an integer list.

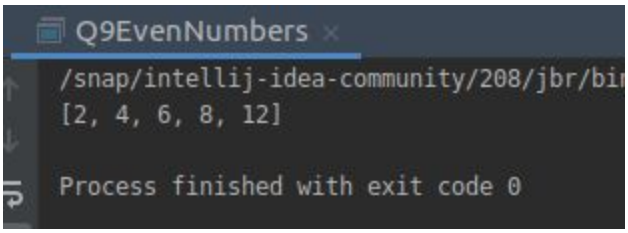
Code :

```
package com.JavaFeatures;

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

```
public class Q9EvenNumbers {
    public static void main(String[] args) {
        List<Integer> list= Arrays.asList(1,2,3,3,4,5,6,8,9,12);
        System.out.println(list
                .stream()
                .filter(e->e%2==0).collect(Collectors.toList()));
    }
}
```

Output:



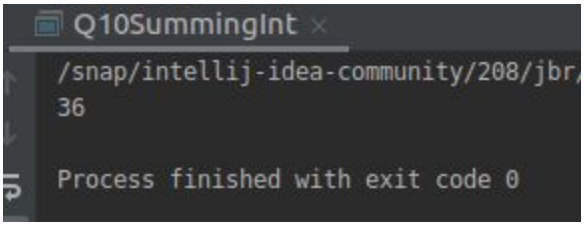
- Sum all the numbers greater than 5 in the integer list.

Code :

```
package com.JavaFeatures;

import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;
public class Q10SummingInt {
    public static void main(String[] args) {
        List<Integer> list4= Arrays.asList(1,20,3,5,10,6);
        System.out.println(list4
                .stream()
                .filter(e->e>5)
                .collect(Collectors.summingInt(e->e)));
    }
}
```

Output:



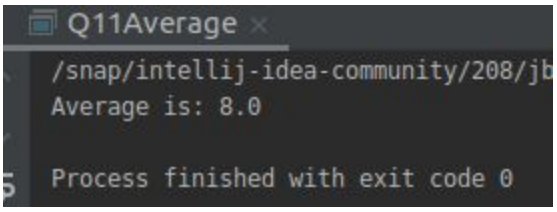
- Find average of the number inside integer list after doubling it.

Code :

```
package com.JavaFeatures;

import java.util.Arrays;
import java.util.List;
import java.util.OptionalDouble;
public class Q11Average {
    public static void main(String[] args) {
        List<Integer> list= Arrays.asList(1,2,4,6,8,3);
        OptionalDouble op=list.stream().map(e->e*2).mapToInt(e->e).average();
        System.out.println("Average is: "+op.getAsDouble());
    }
}
```

Output:



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

Code :

```
package com.JavaFeatures;

import java.util.Arrays;
import java.util.List;
import java.util.Optional;
public class Q12EvenGREATERthree {
```

```
    public static void main(String[] args) {
        List<Integer> list= Arrays.asList(1,2,4,6,7,2,7,1);
        Optional<Integer> op=
list.stream().filter((e)->e%2==0).filter((e)->e>3).findFirst();
        System.out.println(op.get());
    }
}
```

Output: