Assignment - Java 8 Features - 1

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

1. To check whether the first number is greater than second number or not, Parameter (int ,int) Return type 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)

```java
package Java_8_1.Q1;

@FunctionalInterface 1 usage
interface Greater {
    boolean greater(int a, int b); 1 usage
}

@FunctionalInterface 1 usage
interface Increase {
    int increase(int a); 1 usage
}

@FunctionalInterface 1 usage
interface Concatinate {
    String concatinate(String a, String b); 1 usage
}
@FunctionalInterface 1 usage
interface UpperCase {
    String upperCase(String s); 1 usage
}

public class Q1 { 2 usages
    public static void Implement() { 1 usage
        Greater g = ( int a, int b) ->  a > b;


        if(g.greater( a: 10, b: 1)){
            System.out.println("A is greater than B");
        } else  {
            System.out.println("B is greater than A");
        }

        System.out.println("\n\n");

        Increase i = ( int a) -> a++;
        System.out.println("Value Incremented by 1 : "+ i.increase( a: 10));
```

```
        Increase i = ( int a) -> a++;
        System.out.println("Value Incremented by 1 : "+ i.increase( a: 10));

        System.out.println("\n\n");

        Concatinate c = ( String a, String b) -> a + b;
        System.out.println("Concatinated Strings : " + c.concatinate( a: "a", b: "b"));

        System.out.println("\n\n");

        UpperCase u = ( String s) -> s.toUpperCase();
        System.out.println("UpperCase : " +u.upperCase( s: "akash"));

    }
}
```

```
/usr/lib/jvm/java-1.21.0-openjdk-amd64/bin/java -javaagent:/home/akash/Downloads/idea-IU-251.26094.121/lib/idea_rt.jar=41839 -Dfile.encoding=UTF-8
A is greater than B



Value Incremented by 1 : 10



Concatinated Strings : ab



UpperCase : AKASH

Process finished with exit code 0
```

Q2)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

```java
package Java_8_1.Q2;

@FunctionalInterface  3 usages
interface Operator{
    int apply(int a,int b);  3 usages
}

class Calculator {  3 usages
  public int add(int a,int b){  1 usage
      return a+b;
  }
  public int sub(int a,int b){  1 usage
      return a-b;
  }
  public static int mul(int a,int b){  1 usage
      return a*b;
  }
}
public class Q2 {  2 usages
    public static void MethodReference() {  1 usage
        Calculator c = new Calculator();
        Operator add = c ::add;
        Operator sub = c ::sub;

        System.out.println("\n\n");
        System.out.println("Instance Method Reference");

        System.out.println("On Adding : " + add.apply( a: 10, b: 5));
        System.out.println("On Subtract : " + sub.apply( a: 10, b: 5));

        System.out.println("\n\n");

        System.out.println("Static Method Reference");
        Operator mul = Calculator::mul;

        System.out.println("On Multiplication : " + mul.apply( a: 10, b: 5));

    }
```

```
Instance Method Reference
On Adding : 15
On Subtract : 5



Static Method Reference
On Multiplication : 50

Process finished with exit code 0
```

Q3)Implement multiple inheritance with default method inside interface.

```java
package Java_8_1.Q3;

interface Parent{  2 usages  3 implementations
    default void show(){  1 usage  3 overrides
        System.out.println("Parent");
    }
}
interface Child1 extends Parent{  1 usage  1 implementation
    default void show(){  1 usage  1 override
        System.out.println("Child1");
    }
}
interface Child2 extends Parent{  1 usage  1 implementation
    default void show(){  1 usage  1 override
        System.out.println("Child2");
    }
}

public class Q3 implements Child1, Child2 {  4 usages
    @Override  1 usage
    public void show() {
        System.out.println("Child of child");
    }

    public static void MultupleInheritance() {  no usages
        Q3 q = new Q3();
        q.show();
    }
}
```

```
/usr/lib/jvm/java-1.21.0-openjdk-amd64/bin/java -javaagent:/home/akash/Downloads/idea-IU-251.26094.121
Child of child

Process finished with exit code 0
```

Q4)Write a program to implement constructor reference

```java
package Java_8_1.Q4;

class A {  3 usages
     String name;  2 usages
     int age;  2 usages
     public A(String name, int age) {  1 usage
         this.name = name;
         this.age = age;
     }
    public void show(){  1 usage
        System.out.println("name : " + name  + " age : " + age);
    }
}

@FunctionalInterface  1 usage
interface ConstRefernce {
    A getA(String name, int age);  1 usage
}

public class Q4 {  2 usages
    public static void ConstructorRefernce() {  1 usage

        ConstRefernce refernce = A::new;

        A a = refernce.getA( name: "A",  age: 18);
        a.show();

    }
}
```

```
/usr/lib/jvm/java-1.21.0-openjdk-amd64/bin/java -javaagent:/home/akash/Downloads/idea-IU-251.26094.121/lib/idea_rt.jar=39429
name : A age : 18

Process finished with exit code 0
```