5. Functional Interfaces

- Interface which contains Only Single Abstract Method (SAM) is called Functional Interface.
- You can mark the Functional Interface optionally with **@FunctionalInterface**
- Functional Interface can have the following
  - One Abstract Method
  - Multiple default methods
  - Multiple static methods
  - Multiple private methods **(from Java9)**
  - Multiple private static methods **(from Java9)**

- You need a Functional Interface to write the Lambda Expressions.
- You can define the Functional Interface in your own or you can make use of Built-In Functional Interface
- Following are the List of most important of Built-In Functional Interfaces
  - Predicate
  - BiPredicate
  - Function
  - UnaryOperator
  - BiFunction
  - BinaryOperator
  - Consumer
  - BiConsumer
  - Suppiler
  - Primitive Functional Interfaces

Demo1: Files Required:

1. Hello.java	2. Demo1.java

1)Hello.java

```
package com.jlcindia.java8.demos;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
@FunctionalInterface
public interface Hello {

        void test(int a,int b) throws ArithmeticException;

}
```

2)Demo1.java

```
package com.jlcindia.java8.demos;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Demo1 {
public static void main(String[] args) {

Hello hello=  (a,b) -> {
System.out.println("Lambda Code Starts");

try {
int result = a/b;
System.out.println("Result is "+ result);
}catch(Exception ex) {
ex.printStackTrace();
}

System.out.println("Lambda Code Ends");
};

hello.test(50, 0);
}
}
```

Demo2: Files Required:

1. Hello.java	2. Demo2.java

1)Hello.java

```
package com.jlcindia.java8.demos;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
@FunctionalInterface
public interface Hello {
        void test(int a,int b) throws ArithmeticException;
}
```

2)Demo2.java

```
package com.jlcindia.java8.demos;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Demo2 {
public static void main(String[] args) {
System.out.println("main Begin");

Hello hello=  (a,b) -> {
System.out.println("Lambda Begin");
int result = a/b;
System.out.println("Result is "+ result);
System.out.println("Lambda End");
};

//hello.test(50, 0);

try{
hello.test(50, 0);
}catch(Exception ex) {
ex.printStackTrace();
}
System.out.println("main End");
}
}
```

5.1. Predicate Functional Interfaces

- **Predicate** Functional Interface
    - Takes One Input Parameter
    - Returns Boolean after processing.
- **Predicate** Functional Interface has the following methods.

abstract boolean **test**(T); //SAM

static <T> Predicate<T> **isEqual**(Object); //Static

Predicate<T> **negate**(); //Default

Predicate<T> **and**(Predicate<? super T>);//Default

Predicate<T> **or**(Predicate<? super T>);//Default

Demo3: Files Required:

1. Demo3.java	

Demo3.java

```
package com.jlcindia.java8.demos;

import java.util.function.Predicate;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Demo3 {
public static void main(String[] args) {

Predicate<Integer> predicate1 = (num) -> {
System.out.println(num);
return num % 2 == 0;
};

boolean mybool = predicate1.test(19);
System.out.println(mybool);

mybool = predicate1.test(28);
System.out.println(mybool);

Predicate<Integer> predicate2 = (num) -> num % 2 != 0;
mybool = predicate2.test(19);
System.out.println(mybool);
mybool = predicate2.test(28);
System.out.println(mybool);
}
}
```

Demo4: Files Required:

1. Demo4.java	

Demo4.java

```
package com.jlcindia.java8.demos;

import java.util.ArrayList;
import java.util.List;
import java.util.function.Predicate;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Demo4 {
public static void main(String[] args) {

Predicate<Integer> predicate1 = (num) -> {
System.out.println(num);
return num % 2 == 0;
};

Predicate<Integer> predicate2 = (num) -> num % 2 != 0;

List<Integer> mylist1 = new ArrayList<>();
mylist1.add(20);     mylist1.add(21);     mylist1.add(22);   mylist1.add(23);
mylist1.add(24);     mylist1.add(25);     mylist1.add(26);

System.out.println(mylist1);
mylist1.removeIf(predicate1);
System.out.println(mylist1);

System.out.println("------------------------");
List<Integer> mylist2 = new ArrayList<>();
mylist1.add(20);     mylist1.add(21);     mylist1.add(22);   mylist1.add(23);
mylist1.add(24);     mylist1.add(25);     mylist1.add(26);
System.out.println(mylist2);
mylist2.removeIf(predicate2);
System.out.println(mylist2);
}
}
```

Demo5: Files Required:

1. Demo5.java	

Demo5.java

```
package com.jlcindia.java8.demos;

import java.util.ArrayList;
import java.util.List;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Demo5 {
public static void main(String[] args) {

List<Integer> mylist1 = new ArrayList<>();
mylist1.add(20);      mylist1.add(21);       mylist1.add(22);   mylist1.add(23);
mylist1.add(24);       mylist1.add(25);       mylist1.add(26);

System.out.println(mylist1);
mylist1.removeIf((num) -> num %2 ==0); //IMP
System.out.println(mylist1);

System.out.println("-------------------------");
List<Integer> mylist2 = new ArrayList<>();
mylist1.add(20);      mylist1.add(21);       mylist1.add(22);   mylist1.add(23);
mylist1.add(24);       mylist1.add(25);       mylist1.add(26);

System.out.println(mylist2);
mylist2.removeIf((num) -> num %2 !=0);  //IMP
System.out.println(mylist2);

}
}
```

Demo6: Files Required:

1. Demo6.java	

Demo6.java

```
package com.jlcindia.java8.demos;

import java.util.function.Predicate;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Demo6 {
public static void main(String[] args) {

Predicate<String>  predicate1 = Predicate.isEqual("Hello Guys");

boolean mybool = predicate1.test("Hello Guys");
System.out.println(mybool);
mybool = predicate1.test("Hai Guys");
System.out.println(mybool);

Predicate<Integer>  predicate2 = Predicate.isEqual(99);

mybool = predicate2.test(99);
System.out.println(mybool);
mybool = predicate2.test(88);
System.out.println(mybool);
System.out.println("--------------");

Predicate<Integer> predicate3 = (num) -> num % 2 == 0;
Predicate<Integer> predicate4 = (num) -> num % 2 != 0;

mybool = predicate3.test(28);
System.out.println(mybool); //T
mybool = predicate3.negate().test(28);
System.out.println(mybool); //F

mybool = predicate4.test(19);
System.out.println(mybool); //T
mybool = predicate4.negate().test(19);
System.out.println(mybool); //F
}
}
```

Demo7: Files Required:

1. Demo7.java	

Demo7.java

```
package com.jlcindia.java8.demos;

import java.util.function.Predicate;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Demo7 {
public static void main(String[] args) {

Predicate<Integer> predicate1 =  (num) ->{
System.out.println("Predicate 1");
return num % 2 == 0;
};

Predicate<Integer> predicate2 = (num) -> {
System.out.println("Predicate 2");
return num % 2 != 0;
};

Predicate<Integer> predicate3 = (num) -> {
System.out.println("Predicate 3");
return  num >= 25 && num <= 50;
};

// Check whether Number is Even and between 25 and 50
boolean mybool = predicate1.and(predicate3).test(28);
System.out.println(mybool);

mybool = predicate1.and(predicate3).test(19);
System.out.println(mybool);

// Check whether Number is Odd  and between 25 and 50
mybool = predicate2.and(predicate3).test(29);
System.out.println(mybool);

mybool = predicate2.and(predicate3).test(19);
System.out.println(mybool);
// Check whether Number is Even  or  between 25 and 50

mybool = predicate1.or(predicate3).test(29);
System.out.println(mybool);

mybool = predicate1.or(predicate3).test(28);
System.out.println(mybool);

mybool = predicate1.and(predicate3).test(18);
System.out.println(mybool);

// Check whether Number is Odd  or  between 25 and 50

mybool = predicate2.or(predicate3).test(29);
System.out.println(mybool);

mybool = predicate2.or(predicate3).test(28);
System.out.println(mybool);

mybool = predicate2.and(predicate3).test(18);
System.out.println(mybool);

}

}
```

<u>**Interview Questions:**</u>

Q1) What is a Function Interface?

Ans:

Q2) What is the annottaion to mark on the Function Interface?

Ans:

Q3) Is it mandatory to mark @FunctionalInterface?

Ans:

Q4) Can I write two abstract methods in One Function Interface?

Ans:

Q5) Can I have default methods in Function Interface?

Ans:

Q6) Can I have static methods in Function Interface?

Ans:

Q7) Can I have private methods in Function Interface? (Java 9)

Ans:

Q8) Can I have private static methods in Function Interface? (Java 9)

Ans:

Q9) Can I write Lambda exp without Function Interface?

Ans:

Q10) Can we change the return type or parameters of SAM during implementation?

Ans:

Q11) How can I throws exception at method level using Functional Interface?

Ans:

Q12) What are the built-IN functional Interfaces?

Ans:

Q13) Which of the following **Hello** are Valid Function Interfaces?

1)	interface **Hello** {
	
	}	

2)	interface **Hello** {
	void show();
	}	

3)	interface Hai {
	void show();
	}
	
	Interface **Hello** extends Hai{
	
	}	

4)	interface **Hello** {
	void show();
	default void display(){
	System.out.println("OK");
	}	

5)	interface **Hello** {
	void m1();
	void m2();
	}	