3. Lambda Expressions

- Functional Programming importance has been increasing in the recent days because it is well suited for concurrent and event driven (or Reactive) programming.
- That doesn't mean that objects are bad Instead, the winning strategy is to blend object oriented and functional programming
- Lambda Expressions are introduced to support functional programming in Java
- Lambda Expression is an anonymous functions
- .i.e Lambda Expression is a function which doesn't have the name, return type and access modifiers
- Lambda Expressions are used heavily inside the Collections, Streams libraries from Java 8
- We need **Functional interfaces** to write lambda expressions.

Why Lambda Expressions:

- Reduce length of the code
- Readability will be improved
- Complexity of anonymous inner classes can be avoided

Demo1: Files Required:

1. Hello.java	2. Demo1.java

1)Hello.java

```
package com.jlcindia.demo1;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
@FunctionalInterface
public interface Hello {

        void display();

        default void m1() {
                System.out.println("Hello - m1()");
                display();
        }

        static void m2() {
                System.out.println("Hello - m2()");
        }
}
```

2)Demo1.java

```
package com.jlcindia.demo1;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Demo1 {

public static void main(String[] args) {

Hello hello1 =  () -> {
System.out.println("Hello Guys!!!");
};

hello1.display();
hello1.m1();
//hello1.m2();
Hello.m2();
Hello hello2 =  () -> System.out.println("Welcome to Lambda World!!!");

hello2.display();
hello2.m1();
//hello2.m2();
Hello.m2();

}
}
```

Demo2: Files Required:

1. Hello.java	2. Demo2.java

1)Hello.java

```
package com.jlcindia.demo2;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
@FunctionalInterface
public interface Hello {

        void display(String name);
}
```

2)Demo2.java

```
package com.jlcindia.demo2;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Demo2 {

public static void main(String[] args) {

Hello hello1=  (name) -> {
System.out.println("Hello "+name +"  Welcome to Lambda World!!!");
};

hello1.display("Srinivas");
Hello hello2=  (name) -> System.out.println("Hello "+name +"  Welcome to Lambda
World!!!");

hello2.display("Sri");

Hello hello3=  name -> System.out.println("Hello "+name +"  Welcome to Lambda
World!!!");

hello3.display("Vas");
}
}
```

Demo3: Files Required:

1. Hello.java	2. Demo3.java

1)Hello.java

```
package com.jlcindia.demo3;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
@FunctionalInterface
public interface Hello {

        void test(int a,int b);
}
```

2)Demo3.java

```
package com.jlcindia.demo3;

/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Demo3 {

public static void main(String[] args) {

Hello hello1=  (a,b) -> {
int sum = a+b;
System.out.println("Sum :  "+sum);
};
hello1.test(100,50);

Hello hello2=  (a,b) -> System.out.println("Sum :  "+ ( a+b));

hello2.test(95,45);

Hello hello3=  (a,b) -> {
int sub = a-b;
System.out.println("Sub :  "+sub);
};

hello3.test(100,50);

Hello hello4=  (a,b) -> System.out.println("Sub :  "+ ( a-b));

hello4.test(95,45);

}
}
```

Demo4: Files Required:

1. Hello.java	2. Demo4.java

1)Hello.java

```
package com.jlcindia.demo4;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
@FunctionalInterface
public interface Hello {
	int test(int a,int b);
}
```

2)Demo4.java

```
package com.jlcindia.demo4;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Demo4 {

public static void main(String[] args) {

        Hello hello1 = (a, b) -> {
        int sum = a + b;
        return sum;
        };

        int sum1 = hello1.test(100, 50);
        System.out.println("Sum :  " + sum1);

        Hello hello2 = (a, b) -> {
        return a + b;
        };

        int sum2= hello2.test(95, 45);
        System.out.println("Sum :  " + sum2);

        Hello hello3 = (a, b) ->  a + b;

        int sum3= hello3.test(90, 40);
        System.out.println("Sum :  " + sum3);

}
}
```

Interview Questions:

Q1) What is Lambda Expression?

Ans:

Q2) What is Annonymous Method?

Ans:

Q3) What is the Functional Interface?

Ans:

Q4) What are the uses Lambda Expressions?

Ans:

Q5) How Lambda Expressions are better than Annonymous Inner Classes?

Ans:

Q6) How Can I Reuse the Lambda Expressions?

Ans:

Q7) How Can I use lambda expression with functional interface?

Ans:

Q8) How the Lambda Expression Parameter Type will be verified?

Ans:

Q9) How the Lambda Expression Return Type will be verified?

Ans:

Q10) How target type is inferred for the lambda expression?

Ans:

Q11) Which of the following Lambda Expressions are Valid for Hello Functuon Interface given?

```
@FunctionalInterface
public interface Hello {
        int test(int a,int b);
}
```

1)	Hello hello = (int a,int b) -> {
	int sum = a + b;
	return sum;
	};	

2)	Hello hello = (a, b) -> {
	System.out.println(a + b);
	};	

3)	Hello hello = (a, b) -> {
	return a + b;
	};	

4)	Hello hello = (a, b) -> a + b;	
5)	Hello hello = (a, b,c) -> a + b-c;	
6)	Hello hello = (int a, b) -> a + b;	
7)	Hello hello = a, b -> a + b;	

8)	Hello hello = (a, b) => {
	return a + b;
	};	