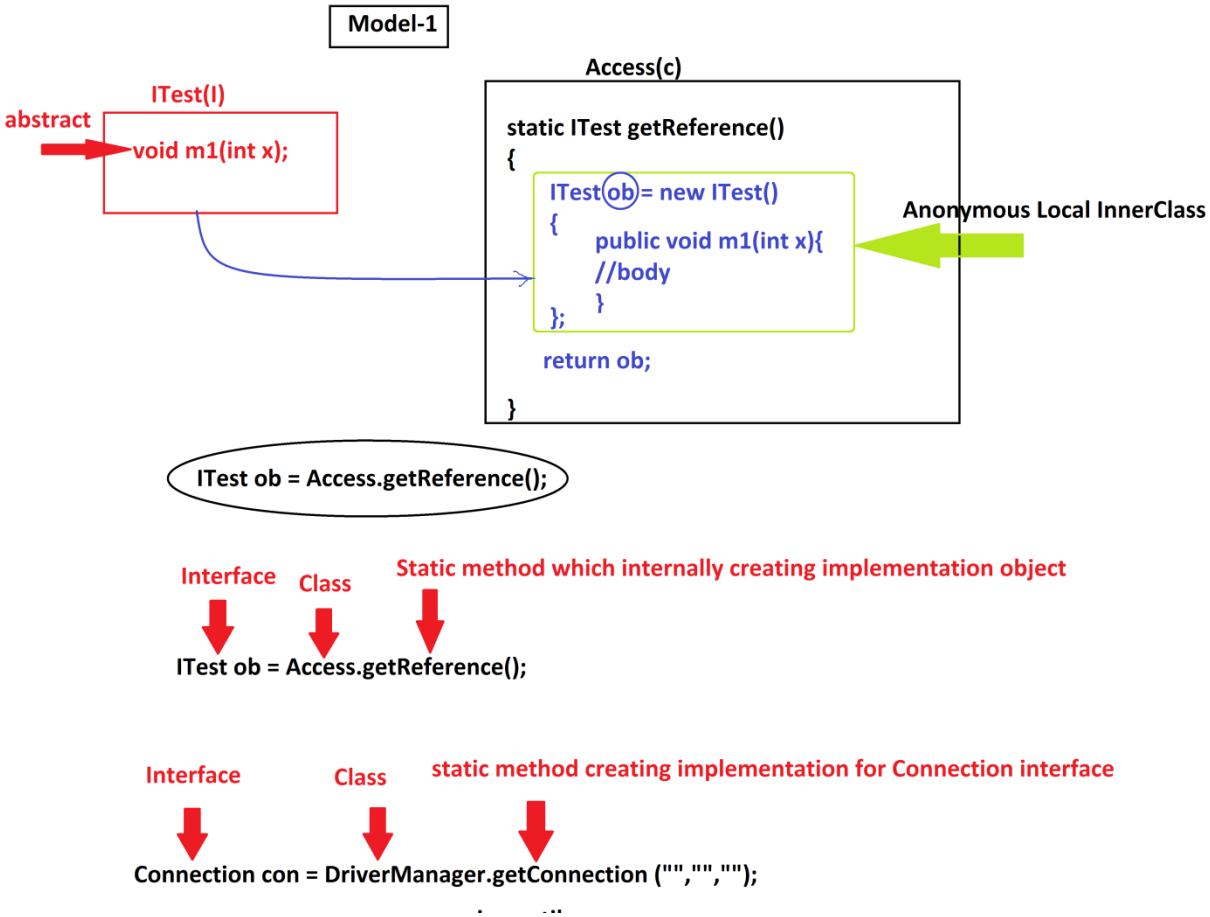
Dt : 26/10/2022

**imp*

Design Models in Applications:

Model-1 : we create implementation object for Interface by declaring implementation class as "Anonymous Local InnerClass".

Diagram:



Ex:

ITest.java

```
package test;
public interface ITest {
   public abstract void m1(int x);
}
```

Access.java

```
package test;
public class Access
{
     public static ITest getReference()
     {
      ITest ob = new ITest()
      {
         public void m1(int x)
         {
          System.out.println("===m1(x)===");
          System.out.println("The value x:"+x);
         }
      };
      return ob;
     }//OuterClass method
}//OuterClass
```

DemoDesignModel1.java(MainClass)

```
package maccess;
import test.*;
public class DemoDesignModel1 {
     public static void main(String[] args) {
       ITest ob = Access.getReference();
                //Implementation object is created using
method
       ob.m1(121);
     }
}
```

o/p:

===m1(x)===

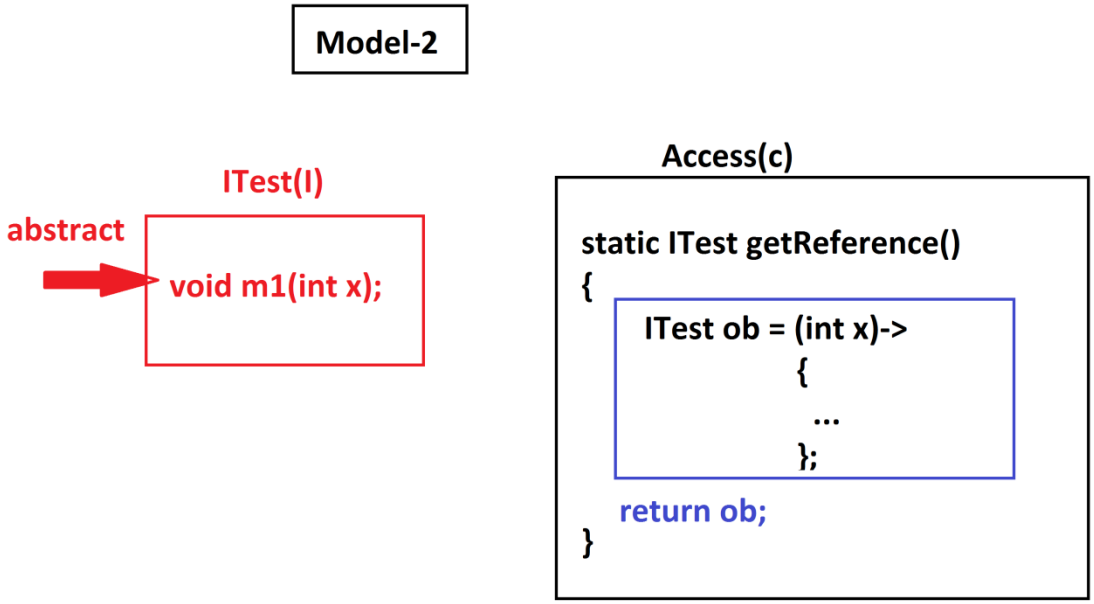
The value x:121

==========

Model-2 : we create Implementation object for Interface using

LambdaExpression

Diagram:



Ex:

ITest.java

```
package test;
public interface ITest {
   public abstract void m1(int x);
}
```

Access.java

```
package test;
public class Access
{
      public static ITest getReference()
      {
       ITest ob = (int x)->
           {
```

```
            System.out.println("===m1(x)===");
            System.out.println("The value x:"+x);
           };
       return ob;
      }//OuterClass method
}//OuterClass
```

DemoDesignModel2.java(MainClass)

```
package maccess;
import test.*;
public class DemoDesignModel2 {
     public static void main(String[] args) {
       ITest ob = Access.getReference();
                    //Implementation object is created using
method
       ob.m1(121);
     }
}
```

o/p:

===m1(x)===

The value x:121

==========

**imp*

define Spliterator<T>? (Java8 - new Version - component)

=>Spliterator<T> is an interface from java.util package introduced by Java8 version and which is used to retrieve elements from Array objects and Collection<E> objects.

=>The following is one important method of Spliterator<T>:

public default void forEachRemaining

(java.util.function.Consumer<? super T>);

=>we use spliterator() method from java.util.Arrays class to create the implementation object for Spliterator<T> interface.

syntax:

Spliterator<T> ob = Arrays.spliterator(arr_var);

define Consumer<T>?(Java8 - new version - Component)

=>Consumer<T> is a functional interface from java.util.function package introduced by Java8 version and this Consumer<T> will provide abstract method "accept(T)" to hold LambdaExpression passed as parameter to forEachRemaining() method.

structure of Consumer<T>:

```
public interface java.util.function.Consumer<T>
{
  public abstract void accept(T);
}
```

```
Consumer<T> obj = (T)->
            {
              ....
            };
```

Ex-program : DemoSpliterator.java

```
package maccess;
import java.util.*;
public class DemoSpliterator {
     public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.println("Enter the size of Array:");
        int n = s.nextInt();
        Integer a[] = new Integer[n];
        System.out.println("Enter "+n+" Integer elements:");
        for(int i=0;i<a.length;i++)
        {
         a[i] = new Integer(s.nextInt());
        }//end of loop
        System.out.println("===Old for loop===");
        for(int i=0;i<a.length;i++)
        {
         System.out.print(a[i].toString()+" ");
        }//end of loop
        System.out.println("\n====Extended for====");
        for(Integer i : a )
        {
         System.out.print(i.toString()+" ");
        }//end of loop
        System.out.println("\n====Spliterator<T>(Java8)====");
        Spliterator<Integer> ob = Arrays.spliterator(a);
        ob.forEachRemaining((k)->
        {
         System.out.print(k.toString()+" ");
        });
        s.close();
     }
}
```

o/p:

Enter the size of Array:

5

Enter 5 Integer elements:

11

12

13

14

15

===Old for loop===

11 12 13 14 15

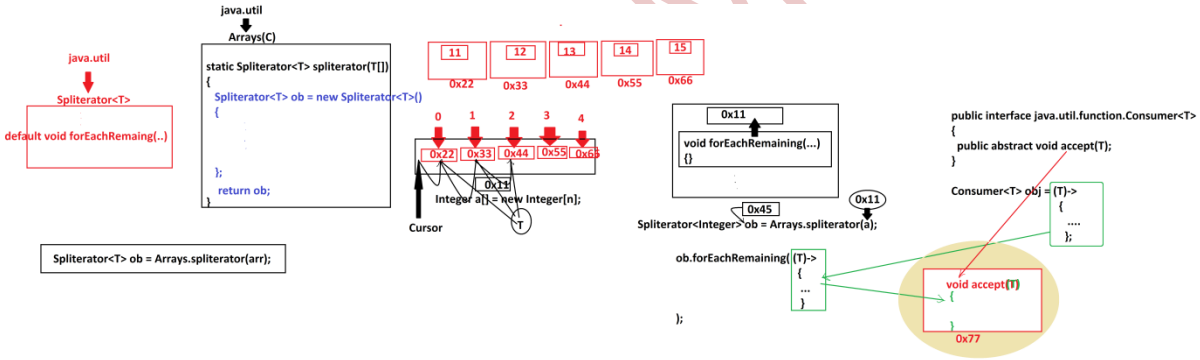
====Extended for====

11 12 13 14 15

====Spliterator<T>(Java8)====

11 12 13 14 15

Diagram:



==========