CONCORDIA UNIVERSITY
COEN-448

Software Testing and Validation
Black Box Testing

Name: RIFADUL HAQUE

ID:

Section: W

Due Date:13/2/22

I certify that this submission is my original work and meets the Faculty's Expectations of Originality.

1. **Apply the Input Domain Modeling techniques to fill in the table of program characteristics and block values for the method remove() of any class that implements ADTList. Please include both interface and functionality based IDM for defining characteristics.**

Linked-List

Characteristics(interface-based-IDM)	b1	b2
List is null	True	False
List is empty	True	False

Array-List

Characteristics(interface-based-IDM)	b1	b2
List is null	True	False
List is empty	True	False

Double-Linked-List

Characteristics(interface-based-IDM)	b1	b2
List is null	True	False
List is empty	True	False

Below, I have provided the screenshot of the tests and the result as well.
I have also verified list is null and empty for all the three lists which are linked list, double-linked-list and array list.

```java
@Test
public void testRemove()
{
    //test for null and empty for the Array list
    L2.clear();
    L2.append(1);
    assertEquals( expected: "< | 1 >", L2.toString());
    assertEquals( expected: 1, (int)L2.remove());
    assertEquals( expected: "< | >", L2.toString());
    assertEquals( expected: null, L2.remove());


    //test for null and empty for the  Linked-List
      L1.clear();
      L1.append(2);
      assertEquals( expected: "< | 2 >", L1.toString());
      assertEquals( expected: 2, (int)L1.remove());
      assertEquals( expected: "< | >", L1.toString());
      assertEquals( expected: null, L1.remove());

      //test for null and empty for the Double-Linked-List
      L3.clear();
      L3.append(4);
      assertEquals( expected: "< | 4 >", L3.toString());
      assertEquals( expected: 4, (int)L3.remove());
      assertEquals( expected: "< | >", L3.toString());
      assertEquals( expected: null, L3.remove());
}
```

```
Run:  ListJUnitTest.testRemove
Tests passed: 1 of 1 test – 14 ms
Test Results                14 ms   C:\Users\rifad\.jdks\openjdk-15.0.2\bin\java.exe ...
  ListJUnitTest             14 ms
    testRemove()            14 ms   Process finished with exit code 0
```

Array-List

Characteristics (Functionality-based-IDM)	b1	b2	b3
Number of elements removed from the list	0	1	More than 1
Remove element first in list	True	False	
Remove element last in List	True	False	

```java
@Test
public void testRemoveForFunctionality(){

    //Testing for Array List

    //removing 1 element from the list and removing element from the front of the list
    L2.clear();
    L2.append(1);
    assertEquals( expected: "< | 1 >", L2.toString());
    assertEquals( expected: 1, (int)L2.remove());

    //remove 2 elements from the list
    L2.append(1);
    L2.append(2);
    assertEquals( expected: "< | 1 2 >", L2.toString());
    assertEquals( expected: 1, (int)L2.remove());
    assertEquals( expected: 2, (int)L2.remove());

    //remove 0 elements from the list
    assertEquals( expected: "< | >", L2.toString());
    assertEquals( expected: null, L2.remove());
```

Linked-List

Characteristics (Functionality-based-IDM)	b1	b2	b3
Number of elements removed from the list	0	1	More than 1
Remove element first in list	True	False	
Remove element last in List	True	False	

```java
//Testing for Linked list

//removing 1 element from the list and removing element from the front of the list
L1.clear();
L1.append(1);
assertEquals( expected: "< | 1 >", L1.toString());
assertEquals( expected: 1, (int)L1.remove());

//remove 2 elements from the list
L1.append(1);
L1.append(2);
assertEquals( expected: "< | 1 2 >", L1.toString());
assertEquals( expected: 1, (int)L1.remove());
assertEquals( expected: 2, (int)L1.remove());

//remove 0 elements from the list
assertEquals( expected: "< | >", L1.toString());
assertEquals( expected: null, L1.remove());
```

Double-Linked-List

Characteristics (Functionality-based-IDM)	b1	b2	b3
Number of elements removed from the list	0	1	More than 1
Remove element first in list	True	False	
Remove element last in List	True	False	

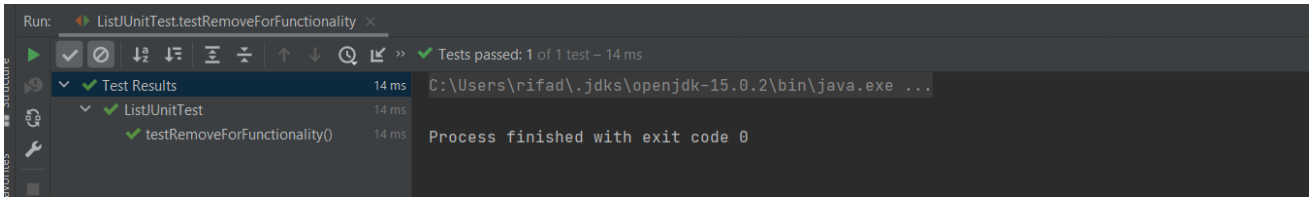

```java
//Testing for Linked list

//removing 1 element from the list and removing element from the front of the list
L1.clear();
L1.append(1);
assertEquals( expected: "< | 1 >", L1.toString());
assertEquals( expected: 1, (int)L1.remove());

//remove 2 elements from the list
L1.append(1);
L1.append(2);
assertEquals( expected: "< | 1 2 >", L1.toString());
assertEquals( expected: 1, (int)L1.remove());
assertEquals( expected: 2, (int)L1.remove());

//remove 0 elements from the list
assertEquals( expected: "< | >", L1.toString());
assertEquals( expected: null, L1.remove());
```

The following Screenshot below shows that the tests has passed for the functionality-based IDM



2. **Apply Each Choice Coverage (ECC) and list the generated combination in a table or a list.**

Ans: So here in ECC which is Each choice Coverage we basically must do a combination off each block for each characteristic for the interface-base-IDM. And there should be at least one test case that should meet the combination.

So, for the **interface-based-IDM**, I have 2 options which are list is null or list is empty.
So, for me to create a combination in my term, I would first name them for creating feasibility of creating the combinations.

For linked List and for the option List is Empty let's name it as LinkT1 and LinkF1 and for List is Null let's name it as LinkT2 and NullF2. For Array list and for the option List is Empty let's name it as ArrayT1 and ArrayF1 and for List is Null let's name it as ArrayT2 abd Array F2. I would go for same pattern for the Double-Lined List and lets name it as DoubleT1 and DoubleF1 for List is Empty and DoubleT2 and DoubleF2 for List is Null.

So, the Combination for the Lists in my case would be:

1.(LinkF1, LinkT2)	3.(ArrayF1, ArrayT2)	5.(DoubleF1, Double T2)
2.(LinkT1, LinkF2)	4.(ArrayT1, ArrayF2)	6.(DoubleT1, Double F2)

All the test cases can be seen in the code that I have provided along with the report. But I have only included the each from the common type in the report.
Since case 1,3 and 5 are common and 2,4 and 6 are common. So a screenshot of one of the common type is given below:

```java
public void testRemoveInterfaceECC2(){

    //LinkT1, LinkF2

    //tracker for null and empty list check
    boolean nullChecker= false;
    boolean emptyChecker= false;

    //clearing the list and then appending value to the list
    L1.clear();
    //returns true if the list is null
    assertEquals( expected: true, !nullChecker);

    L1.append(1);
    assertEquals( expected: "< | 1 >", L1.toString());

    //checks if the list is empty or not
    if(L1.toString().equals("< | >") && L1.length()==0){
        emptyChecker=true;
    }
    //returns false if the list is not empty
    assertEquals( expected: false, emptyChecker);

}
```

```
C:\Users\rifad\.jdks\openjdk-15.0.2\bin\java.exe ...

Process finished with exit code 0
```

```java
public void testRemoveInterfaceECC1(){
    //LinkF1, LinkT2
    //tracker for null and empty list check
    boolean nullChecker= false;
    boolean emptyChecker= false;
    //clearing the list and then appending value to the list
    L1.clear();
    L1.append(1);

    //if there is an item in the list and we want to check
    //if we remove the item then the list if null or not.
    if(L1.remove()==null){
        nullChecker=true;
    }
    //if it returns false then the list is not null
    assertEquals( expected: false, nullChecker);

    //adding 2 in the list for checking list empty upon removing the item.
    L1.append(2);
    assertEquals( expected: "< | 2 >", L1.toString());
    assertEquals( expected: 2, (int)L1.remove());

    //using if statment to check the length and the print statement of the list to determine if the list is empty or not.
    if(L1.toString().equals("< | >") && L1.length()==0){
        emptyChecker=true;
    }
    //expected print of the list
    assertEquals( expected: "< | >", L1.toString());
    //statement to check if the list is empty or not
    assertEquals( expected: true, emptyChecker);
}
```

The result of all the test cases are given in the answer of question 4.

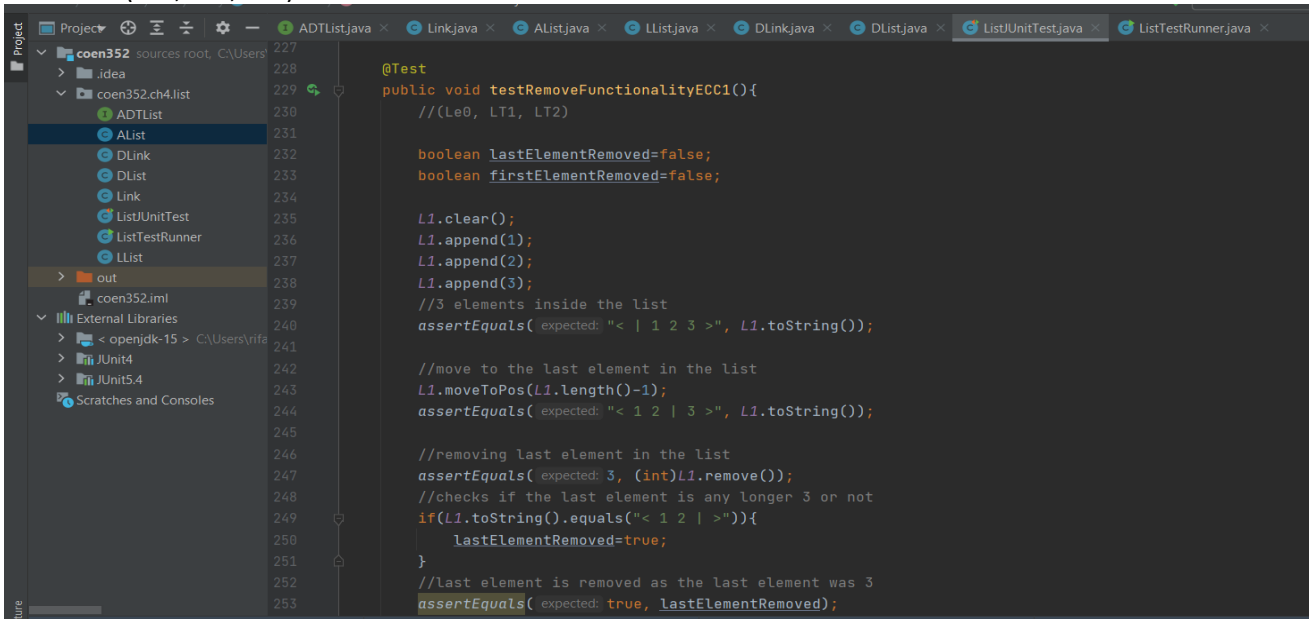
So, for the **functionality-based-IDM**, I have are Number of elements removed from the list and remove first element from the list. Same for this one also I would name them for making it easy to create combinations.

For the Linked list and for the option Number of elements removed from the list let's name it as Le0 for 0 elements, Le1 for 1 element and Le2 for more than one element and for Remove element first in list I would name them as LT1 for True and LF1 for False and to remove element from the last I would name the blocks or options as LT2 and LF2. Hence for Array List it would be Ae0 for 0 elements, Ae1 for 1 element and Ae2 for more than one element and for Remove element first in list I would name them as AT1 for True and AF1 for False and to remove element from the last I would name the blocks or options as AT2 and AF2. And finally for Double-Linked-List it would De0 for 0 elements, De1 for 1 element and De2 for more than one element and for Remove element first in list I would name them as DT1 for True and DF1 for False and to remove element from the last I would name the blocks or options as DT2 and DF2.

So, for the combinations for the list would be as follows:

1.(Le0, LT1, LT2)	2.(Ae0, AT1, AT2)	3.(De0, DT1, DT2)
4.(Le1, LF1, LF2)	5.(Ae1, AF1, AF2)	6.(De1, DF1, DF2)
7.(Le2, LT1, LF2)	8.(Ae2, LT1, AF2)	9.(De2, DT1, DF2)

All the test cases can be seen in the code that I have provided along with the report. But I have only included the each from the common type in the report.
Since case 1,2 and 3 is same and case 4,5 and 6 is same and case 7, 8 and 9 is same. So I would provide a screenshot of 1,4 and 7 and the rest cases can be found in the source code that I have provided.
Below are the screenshots:
Test 1 for (Le0, LT1, LT2)



```java
        //removing first element in the list which is 1
        L1.moveToStart();
        assertEquals( expected: 1,L1.remove());
        //checks if the first element is 1 any longer or not and returns true if not, hence first element is removed
        if(L1.toString().equals("< | 2 >")){
            firstElementRemoved=true;
        }
        //the first element is removed
        assertEquals( expected: true, firstElementRemoved);

        L1.remove();
        //checks if there are 0 elements in the list and upon trying to remove 0 elements the length of
        //the list remains 0 hence the test passed.
        assertEquals( expected: "< | >",L1.toString());
        assertEquals( expected: 0,L1.length());
        assertEquals( expected: null,L1.remove());
        assertEquals( expected: 0,L1.length());

    }
```

```
Run: ListJUnitTest.testRemoveFunctionalityECC1
Tests passed: 1 of 1 test – 14 ms
Test Results 14 ms
ListJUnitTest 14 ms
testRemoveFunctionalityECC1() 14 ms

C:\Users\rifad\.jdks\openjdk-15.0.2\bin\java.exe ...

Process finished with exit code 0
Tests passed: 1 (3 minutes ago)
```

Test for (Le1, LF1, LF2) is given below

```java
    @Test
    public void testRemoveFunctionalityECC4(){

        boolean lastElementRemoved=false;
        boolean firstElementRemoved=false;

        //(Le1, LF1, LF2)
        L1.clear();
        L1.append(1);
        //Test for removing 1 element from the list and checking it.
        assertEquals( expected: "< | 1 >",L1.toString());
        assertEquals( expected: 1,L1.length());
        assertEquals( expected: 1,L1.remove());
        assertEquals( expected: 0,L1.length());
        assertEquals( expected: "< | >",L1.toString());

        //Test for returning false upon when the first element from the list is not removed
        L1.append(1);
        L1.append(2);
        L1.append(3);
        L1.append(4);
        L1.next();
        assertEquals( expected: "< 1 | 2 3 4 >",L1.toString());
        assertEquals( expected: 2,L1.remove());
        if(L1.toString().equals("< | 2 3 4 >")){
            firstElementRemoved=true;
        }
```

```
        assertEquals( expected: false, firstElementRemoved);

        //Test for returning false when the last element from the list is not removed
        assertEquals( expected: 3, L1.remove());
        if(L1.toString().equals("< 1 | 3 >")){
            lastElementRemoved=true;
        }
        assertEquals( expected: false, lastElementRemoved);
    }
```

```
Run: ListJUnitTest.testRemoveFunctionalityECC4
Tests passed: 1 of 1 test – 12 ms
Test Results 12 ms
ListJUnitTest 12 ms
testRemoveFunctionalityECC4() 12 ms

C:\Users\rifad\.jdks\openjdk-15.0.2\bin\java.exe ...

Process finished with exit code 0
```

Test for (Le2, LT1, LF2) and the screenshot is given below:

```
    @Test
    public void testRemoveFunctionalityECC7(){
        //(Le2, LT1, LF2)
        boolean lastElementRemoved=false;
        boolean firstElementRemoved=false;

        L1.clear();
        L1.append(1);
        L1.append(2);
        L1.append(3);
        L1.append(4);
        L1.append(5);
        L1.append(6);

        //Test for removing first element in the list
        assertEquals( expected: "< | 1 2 3 4 5 6 >",L1.toString());
        assertEquals( expected: 1,L1.remove());
        if(L1.toString().equals("< | 2 3 4 5 6 >")){
            firstElementRemoved=true;
        }
        assertEquals( expected: "< | 2 3 4 5 6 >", L1.toString());
        assertEquals( expected: true, firstElementRemoved);

        //Test for removing last element from the list
        L1.remove();
        if(L1.toString().equals("< | 2 3 4 5 >")){
            lastElementRemoved=true;
        }
        assertEquals( expected: false, lastElementRemoved);

        //Test for removing more than one element from the list
        assertEquals( expected: "< | 3 4 5 6 >", L1.toString());
        assertEquals( expected: 3, L1.remove());
        assertEquals( expected: 4,L1.remove());
        assertEquals( expected: "< | 5 6 >",L1.toString());

    }
```

```
Run: ListJUnitTest.testRemoveFunctionalityECC7
Tests passed: 1 of 1 test – 15 ms
Test Results 15 ms
ListJUnitTest 15 ms
testRemoveFunctionalityECC7() 15 ms

C:\Users\rifad\.jdks\openjdk-15.0.2\bin\java.exe ...

Process finished with exit code 0
```

The result of all the test cases is given in the answer of question 4.

3. **Apply Basic Choice Coverage (BCC) and list the generated combination in a table or a list.**

   Ans: So, in the Basic Choice Coverage, it mainly to select a base case which at least one element from all the characteristics. And then we keep that base case and keep on changing the elements. He basically we keep the base choice constant and keep on changing the non-base choices or the options in the base choice. I will provide my coverage as an example for better demonstration of the project.

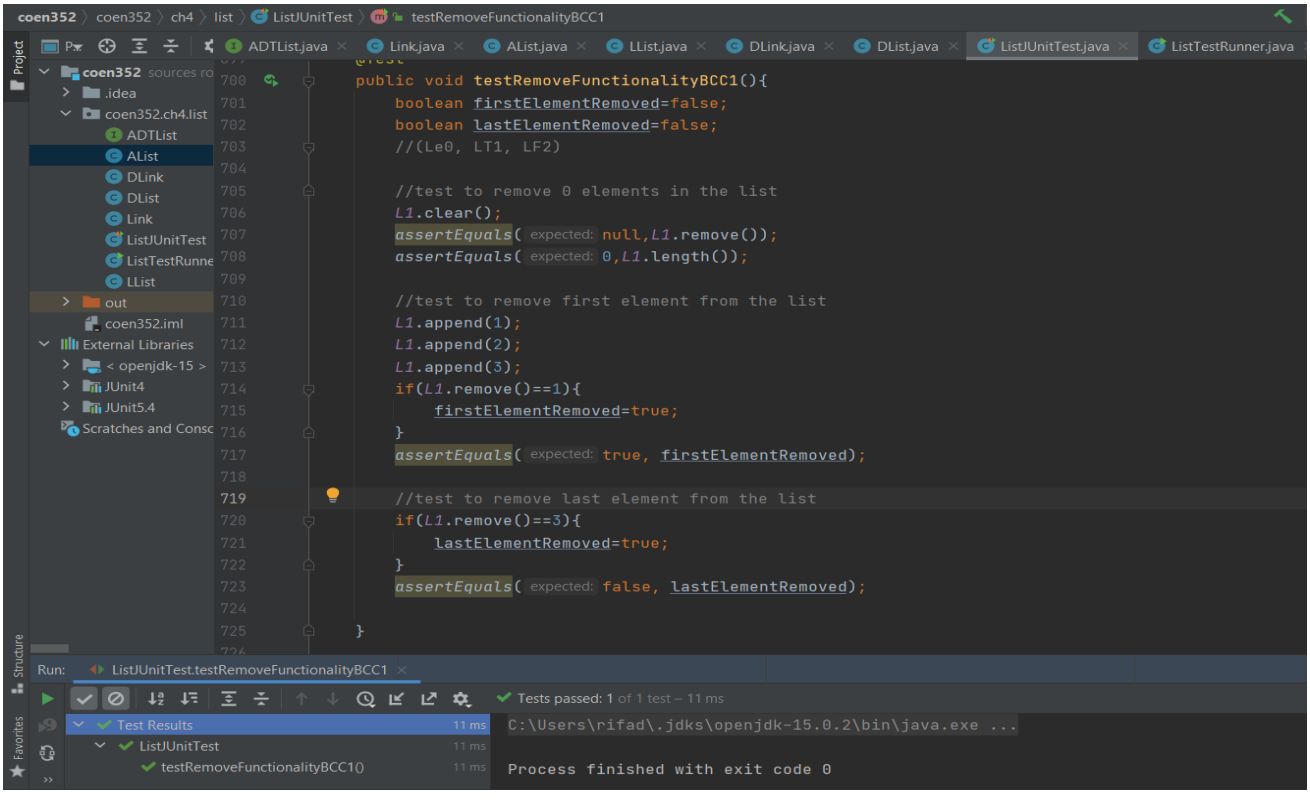
   Functionality-based-IDM:
   So it would be better to name the characteristics and the options at the beginning.
   I would name them the same as I did for BCC and proceed with it.

   So, at first my base case would be for Linked List: (Le0, LT1, LT2), the screenshot for the Array List is given which is same as Linked-List.

   So, the combinations for Linked Lists of functionality-based-IDM:

   1.(Le0, LT1, LF2) 2.(Le1, LT1, LT2)
   3.(Le0, LF1, LT2) 4.(Le2, LT1, LT2)

Test→ (Le0, LT1, LF2—it is same as for Array list and Double linked List



Test-→ (Le1, LT1, LT2)

```java
@Test
public void testRemoveFunctionalityBCC2(){
    boolean firstElementRemoved=false;
    boolean lastElementRemoved=false;
    //(Le1, LT1, LT2)

    //test to remove 1 item from the list and remove the first element from the list
    L1.clear();
    L1.append(1);
    L1.append(2);
    L1.append(3);
    if(L1.remove()==1){
        firstElementRemoved=true;
    }
    assertEquals( expected: "< | 2 3 >",L1.toString());
    assertEquals( expected: true, firstElementRemoved);

    //test to remove last element in the list
    L1.moveToPos(L1.length()-1);
    if(L1.remove()==3){
        lastElementRemoved=true;
    }
    assertEquals( expected: "< 2 | >",L1.toString());
    assertEquals( expected: true, lastElementRemoved);
}
```

```
Run: ListJUnitTest.testRemoveFunctionalityBCC2
Tests passed: 1 of 1 test – 10 ms
Test Results 10 ms
ListJUnitTest 10 ms
testRemoveFunctionalityBCC2() 10 ms
C:\Users\rifad\.jdks\openjdk-15.0.2\bin\java.exe ...

Process finished with exit code 0
```

Test--→(Le0, LF1, LT2)

```java
@Test
public void testRemoveFunctionalityBCC3(){
    boolean firsElementRemoved=false;
    boolean lastElementRemoved=false;
    //(Le0, LF1, LT2)
    //test to remove 0 elements from the list
    L1.clear();
    assertEquals( expected: null,L1.remove());

    //test to remove first element from the list
    L1.append(1);
    L1.append(2);
    L1.append(3);
    L1.append(4);
    L1.next();
    if(L1.remove()==1){
        firsElementRemoved=true;
    }
    assertEquals( expected: false,firsElementRemoved);
    assertEquals( expected: "< 1 | 3 4 >",L1.toString());

    //test to remove last element in the list
    L1.moveToPos(L1.length()-1);
    if(L1.remove()==4){
        lastElementRemoved=true;
    }
    assertEquals( expected: true, lastElementRemoved);
```

```
Run: ListJUnitTest.testRemoveFunctionalityBCC3
Tests passed: 1 of 1 test – 10 ms
Test Results 10 ms
ListJUnitTest 10 ms
testRemoveFunctionalityBCC3() 10 ms
C:\Users\rifad\.jdks\openjdk-15.0.2\bin\java.exe ...

Process finished with exit code 0
```

Test-→(Le2, LT1, LT2)

```java
@Test
public void testRemoveFunctionalityBBC4(){
    boolean firstElementRemoved=false;
    boolean lastElementRemoved=false;

    //(Le2, LT1, LT2)
    L1.clear();

    //test to remove element from the first and last and remove more than 1 element from the list
    L1.append(1);
    L1.append(2);
    L1.append(3);

    if(L1.remove()==1){
        firstElementRemoved=true;
    }
    assertEquals( expected: true,firstElementRemoved);
    L1.moveToPos(L1.length()-1);
    if(L1.remove()==3){
        lastElementRemoved=true;
    }
    assertEquals( expected: true, lastElementRemoved);

    //hence more than one element is removed from the list
}
```

```
Tests passed: 1 of 1 test – 11 ms
C:\Users\rifad\.jdks\openjdk-15.0.2\bin\java.exe ...

Process finished with exit code 0
```

So, at first my base case would be for Linked List: (Ae0, AT1, AT2)

```java
@Test
public void testRemoveFunctionalityBCCArrayChoice(){
    //(Le0, LT1, LT2)
    boolean firstElementRemoved=false;
    boolean lastElementRemoved=false;
    L2.append(1);
    L2.remove();
    //checks if there are 0 elements in the list and upon trying to remove 0 elements the length of
    //the list remains 0 hence the test passed.
    assertEquals( expected: "< | >",L2.toString());
    assertEquals( expected: 0,L2.length());
    assertEquals( expected: null,L2.remove());
    assertEquals( expected: 0,L2.length());

    //test to return true of the first element is removed
    L2.append(1);
    L2.append(2);
    L2.append(3);
    L2.append(4);
    assertEquals( expected: 1,L2.remove());
    //hence it is proved that the first element is removed
    if(L2.toString().equals("< | 2 3 4 >")){
        firstElementRemoved=true;
    }
    assertEquals( expected: true, firstElementRemoved);
```

```java
    //hence it is proved that the first element is removed
    if(L2.toString().equals("< | 2 3 4 >")){
        firstElementRemoved=true;
    }
    assertEquals( expected: true, firstElementRemoved);

    //test to remove last element in the list
    L2.moveToPos(L2.length()-1);
    assertEquals( expected: 4, L2.remove());
    if(L2.toString().equals("< 2 3 | >")){
        lastElementRemoved=true;
    }
```

```
Run: ListJUnitTest.testRemoveFunctionalityBCCArrayChoice
Tests passed: 1 of 1 test – 9 ms
Test Results 9 ms
ListJUnitTest 9 ms
testRemoveFunctionalityBCCArrayChoice() 9 ms
C:\Users\rifad\.jdks\openjdk-15.0.2\bin\java.exe ...

Process finished with exit code 0
Tests passed: 1
```

So, the combinations for Array List of functionality-based-IDM:

5.(Ae0, AT1, LF2) 6.(Ae1, AT1, AT2)
7.(Ae0, AF1, LT2) 8.(Ae2, AT1, AT2)

The source code contains the tests

So, at first my base case would be for Double-Linked List: (De0, DT1, DT2)
So, the combinations for Double Linked List of functionality-based-IDM:

9.(De0, DT1, DF2) 10.(De1, DT1, DT2)
11.(De0, DF1, DT2) 12.(De2, DT1, DT2)

The source code contains the tests

For characteristics-based-IDM:

My base case for linked-list would be: (LinkT1, LinkT2)

```
}

@Test
public void testRemoveCharacteristicsBCCLinkedListChoice(){
    //(LinkT1, LinkT2)

    boolean listEmpty=false;
    boolean listNull=false;

    //list is empty test
    L1.clear();
    L1.append(1);
    L1.remove();
    if(L1.toString().equals("< | >")){
        listEmpty=true;
    }
    assertEquals( expected: true,listEmpty);

    //test for null
    if(L1.remove()==null){
        listNull=true;
    }
    assertEquals( expected: true,listNull);
}
```

```
Run: ListJUnitTest.testRemoveCharacteristicsBCCLinkedListChoice
Tests passed: 1 of 1 test – 9 ms
Test Results 9 ms
ListJUnitTest 9 ms
testRemoveCharacteristicsBCCLinkedListChoice() 9 ms

C:\Users\rifad\.jdks\openjdk-15.0.2\bin\java.exe ...

Process finished with exit code 0
```

The possible combinations would be:

1.(LinkT1, LinkF2) 2.(LinkF1, LinkT2)

Test-→ (LinkT1,LinkF2)

```java
@Test
public void testRemoveCharacteristicsBCC1(){
    //(LinkT1, LinkF2)
    boolean emptyList=false;
    boolean nullList=false;

    L1.clear();
    //null list check
    if(L1.remove()==null){
        nullList=true;
    }
    assertEquals( expected: true,nullList);

    //empty list check
    L1.append(1);
    if(L1.length()==0){
        emptyList=true;
    }
    assertEquals( expected: 1,L1.length());
    assertEquals( expected: false, emptyList);
}
```

```
Run: ListJUnitTest.testRemoveCharacteristicsBCC1
Tests passed: 1 of 1 test – 10 ms
Test Results 10 ms
ListJUnitTest 10 ms
testRemoveCharacteristicsBCC1() 10 ms
C:\Users\rifad\.jdks\openjdk-15.0.2\bin\java.exe ...

Process finished with exit code 0
```

Test→ (LinkF1, LinkT2)

```java
@Test
public void testRemoveCharacteristicsBCC2(){
    boolean emptyCheck=false;
    boolean nullCheck=false;

    //(LinkF1, LinkT2)

    //list is null test
    L1.clear();
    L1.append(1);
    L1.append(2);
    if(L1.remove()==null){
        nullCheck=true;
    }
    assertEquals( expected: false, nullCheck);

    //list is empty
    L1.remove();
    if(L1.toString().equals("< | >")){
        emptyCheck=true;
    }
    assertEquals( expected: true, emptyCheck);

}
```

```
Run: ListJUnitTest.testRemoveCharacteristicsBCC2
Tests passed: 1 of 1 test – 12 ms
Test Results 12 ms
ListJUnitTest 12 ms
testRemoveCharacteristicsBCC2() 12 ms
C:\Users\rifad\.jdks\openjdk-15.0.2\bin\java.exe ...

Process finished with exit code 0
```

My base case for Array-list would be: (ArrayT1, ArrayT2)→ test is same as linked List
The possible combinations would be:

3.(ArrayT1, ArrayF2) 4.(ArrayF1, ArrayT2)

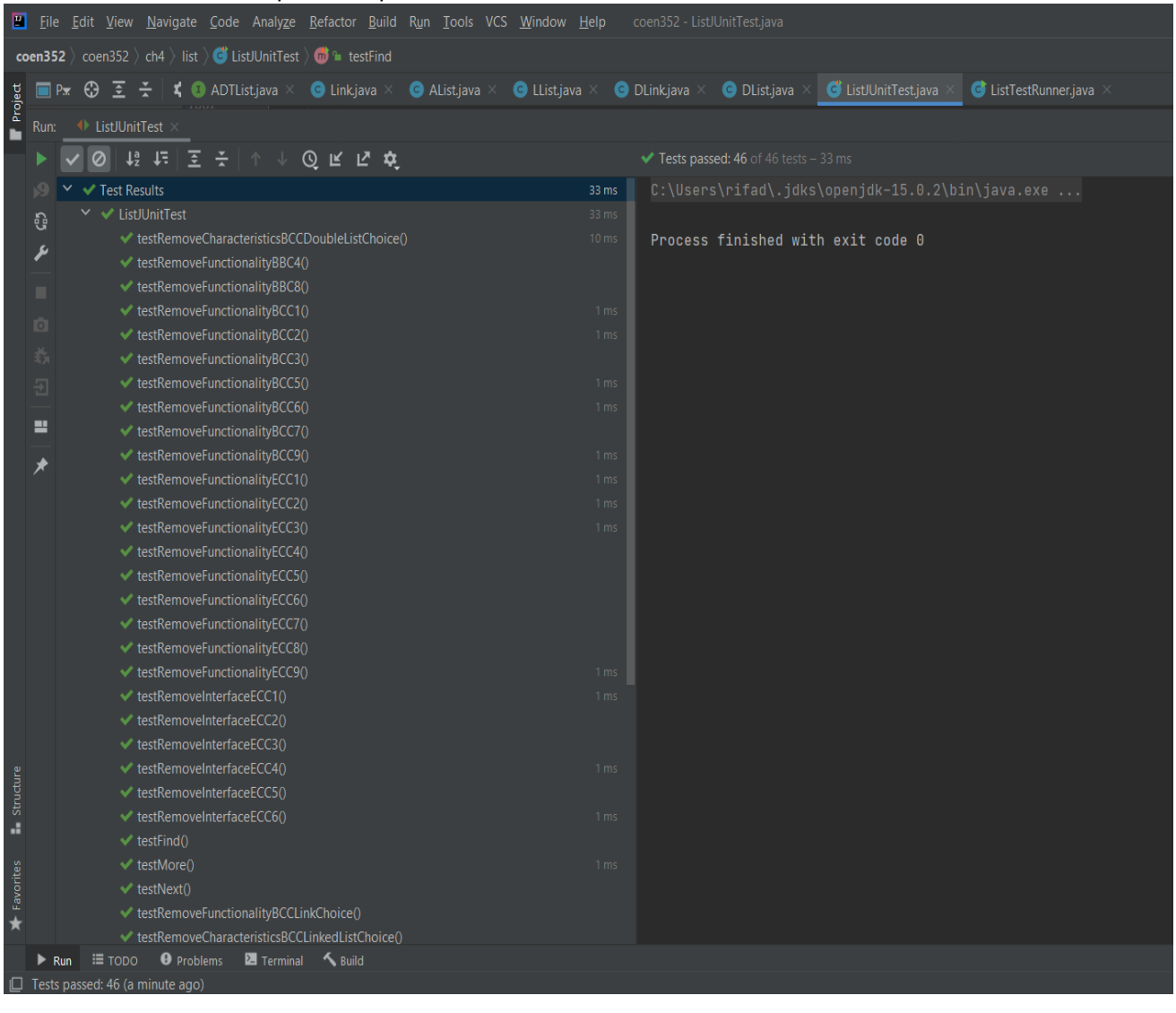
The tests can be found in the source code.

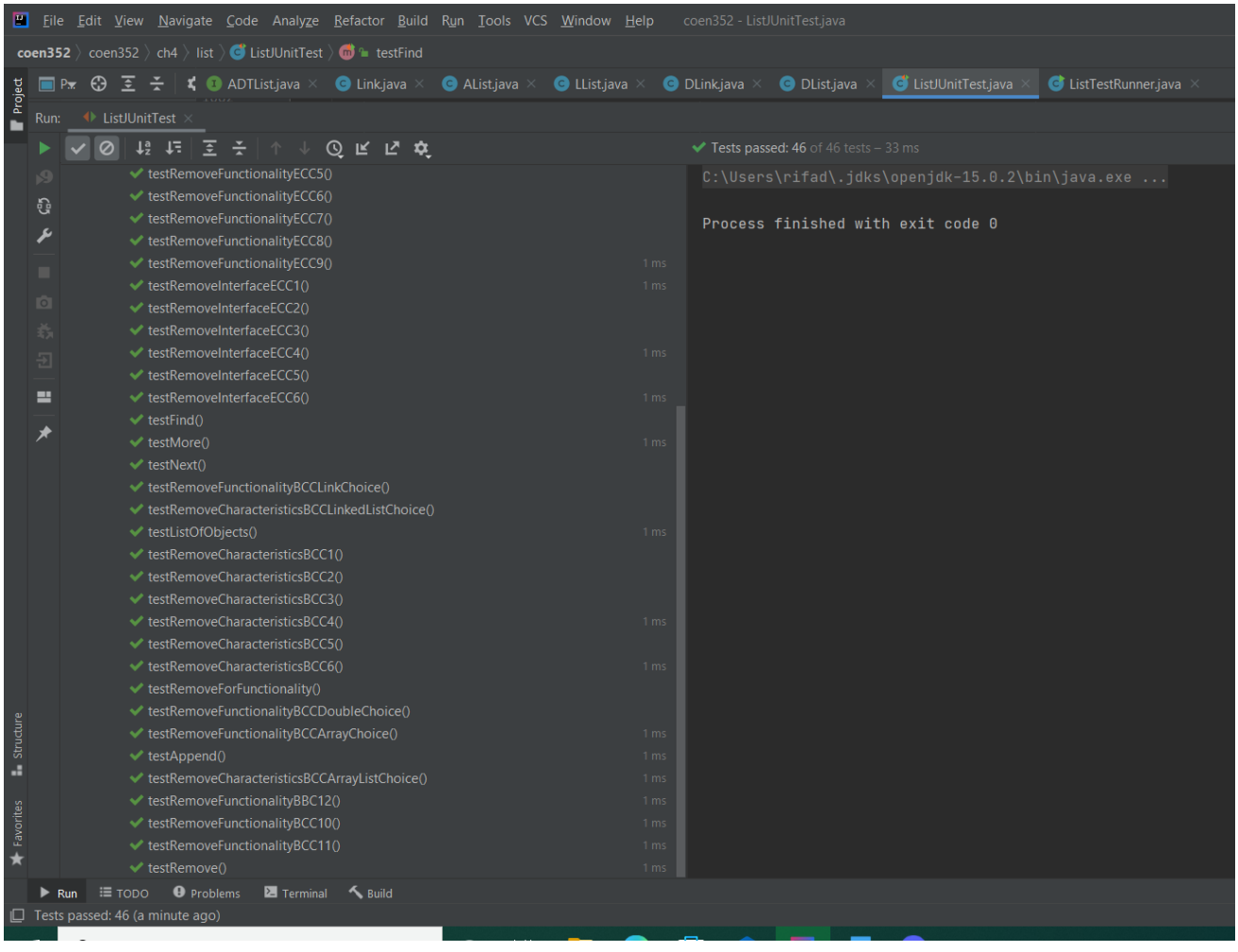
My base case for Double-linked-list would be: (DoubleT1, DoubleT2)→ test is same as Double List
The possible combinations would be:

5.(DoubleT1, DoubleF2) 6.(DoubleF1, DoubleT2)

The tests can be found in the source code.

4. Screenshot of all the test passed is provided below:





The source code is provided along with the report.