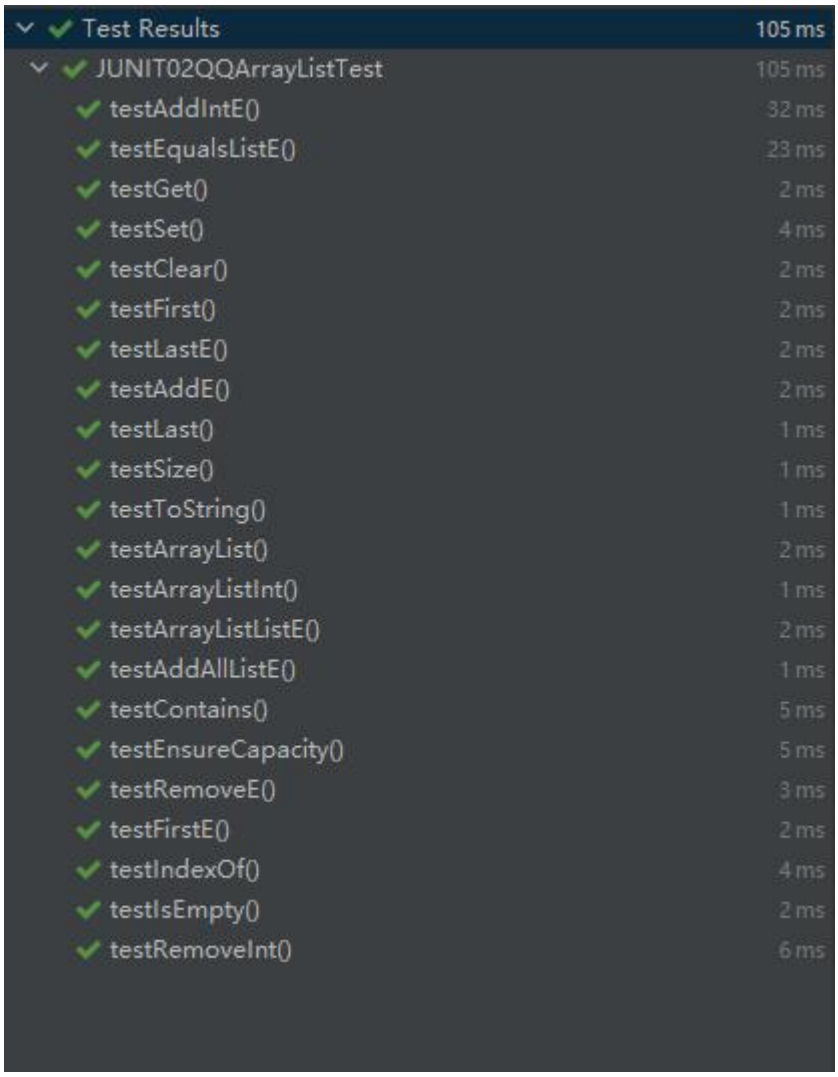
1. JUNIT


Test Results 105 ms
JUNIT02QQArrayListTest 105 ms
testAddIntE() 32 ms
testEqualsListE() 23 ms
testGet() 2 ms
testSet() 4 ms
testClear() 2 ms
testFirst() 2 ms
testLastE() 2 ms
testAddE() 2 ms
testLast() 1 ms
testSize() 1 ms
testToString() 1 ms
testArrayList() 2 ms
testArrayListInt() 1 ms
testArrayListListE() 2 ms
testAddAllListE() 1 ms
testContains() 5 ms
testEnsureCapacity() 5 ms
testRemoveE() 3 ms
testFirstE() 2 ms
testIndexOf() 4 ms
testIsEmpty() 2 ms
testRemoveInt() 6 ms

```
---------- T  E  S  T   ADD(int index, E item) INSERT ----------

[Un]

--------------- ADD IN FRONT ---------------

[Quatre, Trois, Deux, Un]

--------------- ADD AT BACK ---------------

[Quatre, Trois, Deux, Un, Cinq, Six, Sept, Huit]

--------------- ADD IN MIDDLE ---------------

[Quatre, Trois, Deux, Un, Treize, Douze, Onze, Cinq, Six, Sept, Huit]

--------------- ADD AT END ---------------

[Quatre, Trois, Deux, Un, Treize, Douze, Onze, Cinq, Six, Sept, Huit, Quinze, Seize]
size: 13

     0       1       2       3       4       5       6       7       8       9
[(6, 9), (8, 3), (3, 7), (7, 5), (2, 7), (9, 4), (5, 8), (1, 6), (4, 8), (9, 6)]
[(6, 9), (8, 3), (3, 7), (7, 5), (2, 7), (9, 4), (5, 8), (1, 6), (4, 8), (9, 6)]

     0       1       2       3       4       5       6
[(6, 5), (8, 3), (2, 7), (9, 4), (5, 8), (1, 6), (9, 6)]
[(6, 5), (8, 3), (3, 3), (2, 7), (5, 8), (1, 6), (9, 6)]


--------------- T  E  S  T   EQUALS OTHER STRING LIST ---------------

[Un, Deux, Trois, Quatre, Cinq, Six, Sept, Huit, Neuf, Dix]
[Un, Deux, Trois, Quatre, Cinq, Six, Sept, Huit, Neuf, Dix]

     0       1       2       3       4       5       6       7       8       9
[Onze, Douze, Treize, Quatorze, Quinze, Seize, Dix Sept, Dix Huit, Dix Neuf, Vingt]


--------------- T  E  S  T   GET( ) ---------------
[Un, Deux, Trois, Quatre, Cinq, Six, Sept, Huit, Neuf, Dix]
who is at 0? Un
who is at 4? Cinq
who is at 9? Dix
```

```
-------- T  E   S  T   SET(int index, String item) --------



--------------- T  E  S  T   CLEAR( ) ---------------
[Un, Deux, Trois, Quatre, Cinq, Six, Sept]
size BEFORE: 7
size AFTER: 0


--------------  T E S T  FIRST  --------------

[Neuf, Huit, Sept, Six, Un, Deux, Trois, Quatre, Cinq]

[Deux, Trois, Quatre, Cinq]



[(8, 3), (6, 9), (2, 7), (1, 6), (9, 6), (3, 7), (7, 5), (9, 4), (5, 8)]

[(8, 3), (6, 9), (2, 7), (1, 6), (9, 6)]


--------------- T  E  S  T   ADD(E item) APPEND ---------------
[Un, Deux, Trois, Quatre, Cinq, Six, Sept, Huit, Neuf, Dix]
size: 10


==  T  E  S  T    E  X  P  A  N  D    C  A  P  A  C  I  T  Y ==
     Should expand beyond this point or will CRASH
[Un, Deux, Trois, Quatre, Cinq, Six, Sept, Huit, Neuf, Dix, Onze, Douze, Treize, Quatorze, Quinze, Seize]
size AFTER: 16

--------------  T E S T  LAST  --------------

[Un, Deux, Trois, Quatre, Cinq, Six, Sept, Huit, Neuf]

[Un, Deux, Trois, Quatre]
```

```
---------- T  E  S  T     S I Z E () ----------

[Un, Deux, Trois, Quatre, Cinq, Six, Sept, Huit, Neuf, Dix]
size end: 10

---------- T  E  S  T     S I Z E () ----------
[]
[Un]
[Un, Deux]
[Un, Deux, Trois]
[Un, Deux, Trois, Quatre]
[Un, Deux, Trois, Quatre, Cinq]

---------- T  E  S  T   DEFAULT CONSTRUCTOR( ) ----------
size BEFORE: 0
---------- T  E  S  T   EXPANDING CAPACITY ----------
[Un, Deux, Trois, Quatre, Cinq, Six, Sept, Huit, Neuf, Dix, Onze, Douze, Treize, Quatorze, Quinze, Seize, Dix Sept, Dix Huit, Dix Neuf, Vingt]
size AFTER: 20

---------- T  E  S  T   CONSTRUCTOR CAPACITY( ) ----------
size BEFORE: 0
---------- T  E  S  T   EXPANDING CAPACITY ----------
[Un, Deux, Trois, Quatre, Cinq, Six, Sept, Huit, Neuf, Dix, Onze, Douze, Treize, Quatorze, Quinze, Seize, Dix Sept, Dix Huit, Dix Neuf, Vingt]
size AFTER: 20

[(6, 5), (8, 6), (3, 7), (7, 9), (2, 7), (9, 4), (5, 8), (1, 6), (4, 8), (9, 6)]
[(6, 5), (8, 6), (3, 7), (7, 9), (2, 7), (9, 4), (5, 8), (1, 6), (4, 8), (9, 6)]

---------- T  E  S  T   OTHER STRING LIST ----------

[Un, Deux, Trois, Quatre, Cinq, Six, Sept, Huit, Neuf, Dix]
[Un, Deux, Trois, Quatre, Cinq, Six, Sept, Huit, Neuf, Dix]


[(9, 4), (5, 8), (1, 6), (4, 8), (9, 6), (6, 5), (8, 6), (3, 7), (7, 9), (2, 7)]

---------- T  E  S  T   Add ALL Strings ----------

[Un, Deux, Trois, Quatre, Cinq, Six, Sept, Huit, Neuf, Dix]
```

```
------------ T  E  S  T  CONTAINS(String item) ------------
     0       1       2       3       4       5       6       7       8       9
[Un, Deux, Trois, Quatre, Cinq, Six, Sept, Huit, Neuf, Dix]

has Un? true
has Cinq? true
has Onze? false
has Douze?false

---------- T  E  S  T   CONTAINS(Point item) ----------
     0       1       2       3       4       5       6       7       8       9
[(6, 5), (3, 7), (7, 9), (9, 4), (2, 5), (5, 6), (1, 8)]

contains (6, 5)? true
contains (9, 4)? true
contains (1, 8)? true
contains (1, 2)? false
contains (9, 3)? false

---------- T  E  S  T   EnsureCapacity( ) ----------
size BEFORE: 0
----------  T  E  S  T   EXPANDING CAPACITY ----------
     REACHED List LIMIT ....AT Capacity
     Should expand beyond this point or will CRASH
[Un, Deux, Trois, Quatre, Cinq, Six, Sept, Huit, Neuf, Dix, Onze, Douze, Treize, Quatorze, Quinze, Seize, Dix Sept, Dix Huit, Dix Neuf, Vingt]
size AFTER: 20

--------------------------------------------------
TEST 1: size: 110, list-size: 110
TEST 2: size: 20, list-size: 20
TEST 3: size: 10020, list-size: 10020
--------------------------------------------------

---------- T  E  S  T   REMOVE(String item) ----------
size BEFORE: 0

---------- REMOVE FROM MIDDLE ----------
---------- REMOVE FROM FRONT ----------
---------- REMOVE FROM BACK ----------
---------- REMOVE ITEM NOT IN LIST ----------

---------- T  E  S  T   REMOVE(Point item) ----------
---------- REMOVE FROM FRONT ----------
---------- REMOVE FROM BACK ----------
---------- REMOVE ITEM NOT IN LIST ----------
[(7, 9), (2, 7), (9, 4), (5, 8)]
POINT LIST SIZE: 4
```

```
[(5, 8), (9, 4), (7, 5), (3, 7), (8, 3), (6, 9), (2, 7), (1, 6), (9, 6)]

[(8, 3), (6, 9), (2, 7), (1, 6), (9, 6)]


---------- T  E  S  T   INDEX OF(String item) ----------
     0       1       2       3       4       5       6       7       8       9
[Un, Deux, Trois, Quatre, Cinq, Six, Sept, Huit, Neuf, Dix]

location of Un: 0
location of Cinq: 3
location of Dix: 6

[Un, Deux, Trois, Quatre, Cinq, Six, Sept, Huit, Neuf, Dix]

---------- T  E  S  T   INDEX OF(Point item) ----------
     0       1       2       3       4       5       6       7       8       9
[(6, 5), (3, 7), (7, 9), (9, 4), (2, 5), (5, 6), (1, 8), (2, 7), (1, 6), (9, 6)]


location of (6, 5) 0
location of (9, 4) 3
location of (1, 8) 6
location of (1, 2) -1
location of (9, 3) -1


---------- T  E  S  T   IS EMPTY() ----------

[Un, Deux, Cinq, Quatre, Sept, Six]

size before clearing: 6

---------- C  L  E  A  R  ----------
[Un, Deux, Cinq, Quatre, Sept, Six]
Un

[Deux, Cinq, Quatre, Sept, Six]
Deux

[Cinq, Quatre, Sept, Six]
Cinq
```

```
[Quatre, Sept, Six]
Quatre

[Sept, Six]
Sept

[Six]
Six

[]

----------  T  E  S  T   REMOVE(int index) ----------

---------- REMOVE FROM MIDDLE ----------

[Un, Deux, Trois, Quatre, Cinq, Six, Sept, Huit, Neuf, Dix, Onze, Douze, Treize, Quatorze, Quinze, Seize, Dix Sept, Dix Huit, Dix Neuf, Vingt]
[Un, Deux, Trois, Quatre, Cinq, Six, Sept, Huit, Dix, Onze, Douze, Treize, Quatorze, Quinze, Seize, Dix Sept, Dix Huit, Dix Neuf, Vingt]
[Un, Deux, Trois, Quatre, Cinq, Six, Sept, Huit, Onze, Douze, Treize, Quatorze, Quinze, Seize, Dix Sept, Dix Huit, Dix Neuf, Vingt]
[Un, Deux, Trois, Quatre, Cinq, Six, Sept, Huit, Douze, Treize, Quatorze, Quinze, Seize, Dix Sept, Dix Huit, Dix Neuf, Vingt]
---------- REMOVE FROM FRONT ----------
[Deux, Trois, Quatre, Cinq, Six, Sept, Huit, Douze, Treize, Quatorze, Quinze, Seize, Dix Sept, Dix Huit, Dix Neuf, Vingt]
[Trois, Quatre, Cinq, Six, Sept, Huit, Douze, Treize, Quatorze, Quinze, Seize, Dix Sept, Dix Huit, Dix Neuf, Vingt]
[Quatre, Cinq, Six, Sept, Huit, Douze, Treize, Quatorze, Quinze, Seize, Dix Sept, Dix Huit, Dix Neuf, Vingt]
[Cinq, Six, Sept, Huit, Douze, Treize, Quatorze, Quinze, Seize, Dix Sept, Dix Huit, Dix Neuf, Vingt]
---------- REMOVE FROM BACK ----------
[Cinq, Six, Sept, Huit, Douze, Treize, Quatorze, Quinze, Seize, Dix Sept, Dix Huit, Dix Neuf]
[Cinq, Six, Sept, Huit, Douze, Treize, Quatorze, Quinze, Seize, Dix Sept, Dix Huit]
[Cinq, Six, Sept, Huit, Douze, Treize, Quatorze, Quinze, Seize, Dix Sept]

<---------- T E S T  REMOVE(int index) POINTS ---------->

---------- REMOVE FROM FRONT ----------
[(8, 6), (3, 7), (7, 9), (2, 7), (9, 4), (5, 8), (1, 6), (4, 8), (9, 6)]
[(3, 7), (7, 9), (2, 7), (9, 4), (5, 8), (1, 6), (4, 8), (9, 6)]
[(7, 9), (2, 7), (9, 4), (5, 8), (1, 6), (4, 8), (9, 6)]
[(2, 7), (9, 4), (5, 8), (1, 6), (4, 8), (9, 6)]
---------- REMOVE FROM BACK ----------
[(2, 7), (9, 4), (5, 8), (1, 6), (4, 8)]
[(2, 7), (9, 4), (5, 8), (1, 6)]
[(2, 7), (9, 4), (5, 8)]
[(2, 7), (9, 4)]

TEST for throwing IndexOutOfBoundsException -> PASSED
```

2. Summary

Through their own implementation of the List list, understand the underlying principle of Java list implementation, or through the array, if the current array is not enough to store new data, need to expand the array, and then the expanded array assigned to the array property; In addition, it also implements fetch, replace, delete and so on operations, the following is my understanding of each method:

2.1 add method

1) First verify whether the array capacity is sufficient, if the capacity is insufficient, then it needs to be expanded; If the array size is sufficient, then the element is inserted into the specified position;
2) For the array that needs to be expanded, we first determine twice the capacity and the passed parameter, and take the maximum value; Then create a new array and transfer the data from the original array to the new array, and finally assign the new array to the original array;
3) For the expanded array, we insert the array into the specified position, and move the data after the position backward;
4) For the last element, put it in the position of Size, and Size+1 is enough;

2.2 clear method

1) Reset the array to a new array, and the array length is the default length 10;
2) The Size attribute of the list is reset to zero;

2.3 get method

1) Check whether the subscript parameter is valid. If the parameter is less than 0 or greater than or equal to Size, throw an IndexOutOfBoundException;
2) Through the for loop through the number group, find the corresponding element of the subscript, and then return;

2.4 remove method

1) Check whether the subscript parameter is valid. If the parameter is less than 0 or greater than or equal to Size, throw an IndexOutOfBoundException;
2) Find the first occurrence of the subscript according to the subscript or through the parameter value;
3) Delete the element at that position, and move the other elements to the left until the last element;

2.5 set method

1) Check whether the subscript parameter is valid. If the parameter is less than 0 or greater than or equal to Size, throw an IndexOutOfBoundException;
2) Declare generic variables;
3) Through the for loop through the number group, replacing the element of the pointer;