Advanced Object-Oriented Programming

CPT204 – Lab 4

Erick Purwanto

Xi'an Jiaotong-Liverpool University
西交利物浦大学

CPT204 Advanced Object-Oriented Programming

Lab 4

Testing 4, Linked List 1

Welcome !

- Welcome to Lab 4 !
  - We are going to add methods to our **MyList** implementation iteratively and recursively, either mutate the object or not

- You will find in this lab
  1. Lab Exercise 4.1 - 4.4, and their hints
  2. Exercise 4.1 - 4.4

- Download **lab4** zip files from Learning Mall
- Don't forget to import the **lab4** files and the library into an IntelliJ project
  - Read **lab1** again for reference

Lab Exercise 4.1 MyList Iterative Square Mutate

- Complete the method void iterSquareMutList(MyList list) iteratively.
  The method ***modifies/mutates*** list so that all of its elements are squared.
  Use loops.

- Test case:
  list = [1, 2, 3]
  MyList.iterSquareMutList(list);
  list → [1, 4, 9]

WARNING: Hints to the exercise on the next slide

Please try to solve the exercise by yourself first...

Lab Exercise 4.1 MyList Iterative Square Mutate Hints

- Loop using while as long as list is not null
    - square the value
    - move list to list.next

Lab Exercise 4.2 MyList Recursive Square Mutate

- Complete the method void recSquareMutList(MyList list) recursively.
  The method ***modifies/mutates*** list so that all of its elements are squared.
  Do **not** use loops.

- Test case:
  list = [1, 2, 3]
  MyList.recSquareMutList(list);
  list → [1, 4, 9]

WARNING: Hints to the exercise on the next slide

Please try to solve the exercise by yourself first...

Lab Exercise 4.2 MyList Recursive Square Mutate Hints

- Base case
  - when list is null, do nothing
  - when list.next is null, square the value
- Recursive step
  - square the value
  - call the method on list.next

Lab Exercise 4.3 MyList Iterative Square Immutate

- Complete the method MyList iterSquareList(MyList list) iteratively.
  The method ***does not mutate*** list, but create a new MyList object with all of its elements squared.
  Use loops.

- Test case 1:
  list1 = [1, 2, 3]
  MyList list2 = MyList.iterSquareList(list1);
  list1 → [1, 2, 3]
  list2 → [1, 4, 9]

the input MyList object is unchanged

WARNING: Hints to the exercise on the next slide

Please try to solve the exercise by yourself first...

Lab Exercise 4.3 MyList Iterative Square Immutate Hints

- if input list is null, return null
- create a new list result using new and MyList constructor, with squared list's value
- create a pointer ptr pointing to result
- move list to its next
- while list is not null
    - create a new MyList object with squared list value, and store the reference/address in ptr.next
    - move ptr and list to their next
- return result

Lab Exercise 4.4 MyList Recursive Square Immutate

- Complete the method MyList recSquareList(MyList list) recursively.
  The method ***does not mutate*** list, but create a new MyList object with
  all of its elements squared.
  Do **not** use loops.

- Test case 1:
  list1 = [1, 2, 3]
  MyList list2 = MyList.recSquareList(list1);
  list1 → [1, 2, 3]
  list2 → [1, 4, 9]

the input MyList object is unchanged

WARNING: Hints to the exercise on the next slide

Please try to solve the exercise by yourself first...

Lab Exercise 4.4 MyList Recursive Square Immutate Hints

- Base case
  - when list is null, no value to square so return null
- Recursive step
  - return a new MyList object,
    with square of input list's value as value
    while call the method recursively on list.next and store its result as
    the object's next

Week 4 Online Programming Assignments

- Start with creating a good set of test cases first !
  - Include the corner/boundary cases,
    such as **empty** MyList, by: MyList empty = MyList.ofEntries();
    if the input is **empty** (which is **null**), then the output is also **null**

- Use IntelliJ to write and test your code, use the ***Java Visualizer***!

Exercise 4.1 MyList Iterative Catenate Mutate

- Complete the method MyList iterCatMutList(MyList listA, MyList listB) iteratively, to return a list consisting of all elements of listA, followed by all elements of listB.
  The method ***modifies/mutates*** listA so that it is concatenated with listB, if listA is not empty/null.
  Use loops.

- Test case 1:
  list1 = [1, 2, 3], list2 = [4, 5, 6]
  list = MyList.iterCatMutList(list1, list2);
  list → [1, 2, 3, 4, 5, 6]
  list1 → [1, 2, 3, 4, 5, 6]

list1 is changed

Exercise 4.1 MyList Iterative Catenate Mutate

- Test case 2:

  list1 = null,   list2 = null

  list = MyList.iterCatMutList(list1, list2);

  list → null

  list1 → null

- Test case 3:

  list1 = [5],   list2 = null

  list = MyList.iterCatMutList(list1, list2);

  list → [5]

  list1 → [5]

- Test case 4:

  list1 = null,   list2 = [5]

  list = MyList.iterCatMutList(list1, list2);

  list → [5]

  list1 → null

Exercise 4.2 MyList Recursive Catenate Mutate

- Complete the method MyList recCatMutList(MyList listA, MyList listB) recursively, to return a list consisting of all elements of listA, followed by all elements of listB.
  The method ***modifies/mutates*** listA so that it is concatenated with listB, if listA is not empty/null.
  Do **not** use loops.

- Test case 1:
  list1 = [1, 2, 3], list2 = [4, 5, 6]
  list = MyList.recCatMutList(list1, list2);
  list → [1, 2, 3, 4, 5, 6]
  list1 → [1, 2, 3, 4, 5, 6]

list1 is changed

Exercise 4.2 MyList Recursive Catenate Mutate

- Test case 2:

  list1 = null,   list2 = null

  list = MyList.recCatMutList(list1, list2);

  list → null

  list1 → null

- Test case 3:

  list1 = [5],   list2 = null

  list = MyList.recCatMutList(list1, list2);

  list → [5]

  list1 → [5]

- Test case 4:

  list1 = null,   list2 = [5]

  list = MyList.recCatMutList(list1, list2);

  list → [5]

  list1 → null

Exercise 4.3 MyList Iterative Catenate Immutate

- Complete the method MyList iterCatList(MyList listA, MyList listB) iteratively, to return a list consisting of all elements of listA, followed by all elements of listB.
  The method ***does not mutate*** listA.
  Use loops.

- Test case 1:
  list1 = [1, 2, 3], list2 = [4, 5, 6]
  list = MyList.iterCatList(list1, list2);
  list → [1, 2, 3, 4, 5, 6]
  list1 → [1, 2, 3]

list1 is unchanged

Exercise 4.3 MyList Iterative Catenate Immutate

- Test case 2:
  list1 = null,   list2 = null
  list = MyList.iterCatList(list1, list2);
  list → null
  list1 → null

- Test case 3:
  list1 = [5],   list2 = null
  list = MyList.iterCatList(list1, list2);
  list → [5]
  list1 → [5]

- Test case 4:
  list1 = null,   list2 = [5]
  list = MyList.iterCatList(list1, list2);
  list → [5]
  list1 → null

Exercise 4.4 MyList Recursive Catenate Immutate

- Complete the method MyList recCatList(MyList listA, MyList listB) recursively, to return a list consisting of all elements of listA, followed by all elements of listB.
  The method ***does not mutate*** listA.
  Do **not** use loops.

- Test case 1:
  list1 = [1, 2, 3],   list2 = [4, 5, 6]
  list = MyList.recCatList(list1, list2);
  list → [1, 2, 3, 4, 5, 6]
  list1 → [1, 2, 3]

list1 is unchanged

Exercise 4.4 MyList Recursive Catenate Immutate

- Test case 2:
  list1 = null, list2 = null
  list = MyList.recCatList(list1, list2);
  list → null
  list1 → null

- Test case 3:
  list1 = [5], list2 = null
  list = MyList.recCatList(list1, list2);
  list → [5]
  list1 → [5]

- Test case 4:
  list1 = null, list2 = [5]
  list = MyList.recCatList(list1, list2);
  list → [5]
  list1 → null

Thank you for your attention !

- In this lab, you have learned:
  - To create linked list methods *iteratively* and *recursively* that *mutate* or *do not mutate* the input list