# Comp151 Lab03

Write **three** independent applications as per descriptions below:

**Project1**

Define a class LinkedSetWithLinkedBag that represents a set and implements the SetInterface (described in Segment 1.21 of chapter 1). Use the class LinkedBag in your implementation as defined in the UML diagram below. Test your class with the test cases provided in main . See the sample run below. Load to IDEA only the classes needed for this project from the provided Lab03.zip file.

### UML Diagram:

A screenshot of a cell phone

Description automatically generated

### Sample Run:

--> Creating aSet and adding to it elements from inputData: [A, B, C, D, A, C, B, B]

The set contains 4 string(s), as follows:

D C B A

--> Clearing aSet

The set is empty

--> aSet isEmpty returns: true

--> Creating set1 and set2

--> Adding elements to set1

--> set1 after adding elements

The set contains 3 string(s), as follows:

C B A

--> Adding elements to set2

--> set2 after adding elements

The set contains 4 string(s), as follows:

D C B A

--> set1 contains "A": true

--> set1 contains "E": false

--> Removing "B" from set1

--> After removing "B" from set1,

The set contains 2 string(s), as follows:

C A

--> Removing random element from set1

--> set1.remove() returned: "C"

The set contains 1 string(s), as follows:

A

--> Removing "A" from set1

--> After removing "A" from set1,

The set is empty

--> Removing random element from set1

--> set1.remove() returned: "null"

The set is empty

--> Adding 4 elements to set1

--> After adding 4 elements to set1:

The set contains 4 string(s), as follows:

N M L K

--> Trying to add duplicate element "N" to set1

--> After adding a duplicate element "N" to set1

The set contains 4 string(s), as follows:

N M L K

Trying to add null entry

--> set1 after adding:

The set contains 4 string(s), as follows:

N M L K

Process finished with exit code 0

**Project2**

Load to the IDEA the remaining classes from the provided Lab03.zip file. Repeat the previous project inside the LinkedSetWithChainOfNodes class, but this time use a chain of linked nodes instead of the LinkedBag as defined in the UML diagram below. Test your class with the test cases provided in main .

### UML Diagram:

A screenshot of a cell phone

Description automatically generated

### Sample Run:

--> Creating aSet and adding to it elements from inputData: [A, B, C, D, A, C, B, B]

The set contains 4 element(s), as follows:

D C B A

--> Clearing aSet

The set is empty

--> aSet isEmpty returns: true

--> Creating set1 and set2

--> Adding elements to set1

--> set1 after adding elements

The set contains 3 element(s), as follows:

C B A

--> Adding elements to set2

--> set2 after adding elements

The set contains 4 element(s), as follows:

D C B A

--> set1 contains "A": true

--> set1 contains "E": false

--> Removing "B" from set1

--> After removing "B" from set1,

The set contains 2 element(s), as follows:

C A

--> Removing random element from set1

--> set1.remove() returned: "C"

The set contains 1 element(s), as follows:

A

--> Removing "A" from set1

--> After removing "A" from set1,

The set is empty

--> Removing random element from set1

--> set1.remove() returned: "null"

The set is empty

--> Adding 4 elements to set1

--> After adding 4 elements to set1:

The set contains 4 element(s), as follows:

N M L K

--> Trying to add duplicate element "N" to set1

--> After adding a duplicate element "N" to set1

The set contains 4 element(s), as follows:

N M L K

Trying to add null entry

--> set1 after adding:

The set contains 4 element(s), as follows:

N M L K

Process finished with exit code 0

**Project3**

The attached BagIterface.java contains java interface that defines the operations that can be performed on ADT bag. Notice javadoc comments that describe operation’s purpose, parameters and return values.

In addition to these basic operations, the following are included:

* **union** operation that combines the contents of two bags into a third bag (see Exercise 5 in Chapter 1)
* **intersection** operation that creates a third bag of only those items that occur in both two bags (see Exercise 6 in Chapter 1)
* **difference** operation that creates a third bag of the items that would be left in the given bag after removing those that also occur in another bag (see Exercise 7 in Chapter 1)
* **equals** - returns true if the content of two bags are the same. Note that two equal bags contain the same number of entries, each entry occurs in each bag the same number of times and in the same position in the “collection of objects”
* **display** - outputs the content of the bag (implemented for testing only)
* **getMax() -** finds the largest item in the given bag
* **removeMin** – removes and returns the smallest element from the bag
* **moveFirstToEnd -** moves the first item in the bag to be the last item in the same bag
* **findMiddleElementInOnePass** – finds the element in the middle of chain-of-nodes in one pass (HINT: uses two pointers that move at different speed – MUST NOT rely on the number of entries)
* **checkIfLoopExists –** checks in one pass if the chain-of-nodes has a loop (HINT: uses two pointers that move at different speed)

### Your Task:

1. Implement the ADT bag as the class LinkedBag. The class LinkedBag should implement the interface BagInterface. Represent the bag as a chain of linked nodes.
2. Analyze provided interface including javadoc comments that describe the purpose of each method, its parameters and return values. UML diagram is also provided for your reference.
3. Analyze the implementation of the methods provided in the LinkedBag class. Note that the main is also provided.
4. Implement the remaining methods that are “stubs” at this moment. **Your code must not assume that the bag is not empty**:
   1. public boolean equals(BagInterface<T> other);
   2. public void removeEvery(T anEntry);
   3. public T replace(T replacement);
   4. public BagInterface <T> union(BagInterface <T> other);
   5. public BagInterface <T> intersection(BagInterface <T> other);
   6. public BagInterface <T> difference(BagInterface <T> other);
   7. public T getMax();
   8. public void moveFirstToEnd()
   9. public T removeMin();
   10. public boolean checkIfLoopExists();
   11. public T findMiddleElementInOnePass();
5. Make sure that the output is correct (see Sample Run below).

### UML Diagram:

### A screenshot of a cell phone Description automatically generated

### Sample Run:

RUNNING TEST CASES

\*\*\*Testing display method\*\*\*

emptyBag:

The bag is empty.

bag1:

B C A B A

There are 5 element(s) in the bag.

bag2:

The bag is empty.

After removing the first element "B" from bag1, it contains

C A B A

There are 4 element(s) in the bag.

\*\*\*Testing equals method\*\*\*

bag1:

C A B A

There are 4 element(s) in the bag.

Are bag1 and emptyBag equal? --> NO

Are emptyBag and emptyBag equal? --> YES

Are emptyBag and bag1 equal? --> NO

bag2:

X B A C A

There are 5 element(s) in the bag.

Are bag1 and bag2 equal? --> NO

Removed "X" from bag2.

B A C A

There are 4 element(s) in the bag.

Are bag1 and bag2 equal now? --> NO

Created bagCopyOfBag1:

C A B A

There are 4 element(s) in the bag.

Are bag1 and bagCopyOfBag1 equal? --> YES

Created bagCopyOfBag1PlusOne:

C A B A D

There are 5 element(s) in the bag.

Are bagCopyOfBag1PlusOne and bagCopyOfBag1 equal? --> NO

Are bagCopyOfBag1 and bagCopyOfBag1PlusOne equal? --> NO

\*\*\*Testing getMax method\*\*\*

The largest item in emptyBag is: null

bag1:

A C A X B A A

There are 7 element(s) in the bag.

The largest item in bag1 is: X

bag2:

D C C A B B A

There are 7 element(s) in the bag.

The largest item in bag2 is: D

\*\*\*Testing union, removeMin, intersection, difference and subset methods\*\*\*

bag1:

A C A X B A A

There are 7 element(s) in the bag.

bag2:

D C C A B B A

There are 7 element(s) in the bag.

\*\*\*Testing union method\*\*\*

The union of bag1 and bag2 is

A B B A C C D A A B X A C A

There are 14 element(s) in the bag.

The union of bag1 and emptyBag is

A A B X A C A

There are 7 element(s) in the bag.

The union of emptyBag and bag1 is

A A B X A C A

There are 7 element(s) in the bag.

\*\*\*Testing removeMin method\*\*\*

Removed the smallest element "A" from the union bag; the current content is:

A B X A C A

There are 6 element(s) in the bag.

Removed the smallest element "A" from the union bag; the current content is:

B X A C A

There are 5 element(s) in the bag.

Removed the smallest element "A" from the union bag; the current content is:

X B C A

There are 4 element(s) in the bag.

Removed the smallest element "A" from the union bag; the current content is:

B C X

There are 3 element(s) in the bag.

Removed the smallest element "B" from the union bag; the current content is:

C X

There are 2 element(s) in the bag.

Removed the smallest element "C" from the union bag; the current content is:

X

There are 1 element(s) in the bag.

Removed the smallest element "X" from the union bag; the current content is:

The bag is empty.

The union bag is empty and removeMin returned null - CORRECT

\*\*\*Testing intersection method\*\*\*

The intersection of bag1 and bag2 is

B A C A

There are 4 element(s) in the bag.

The intersection of bag1 and emptyBag is

The bag is empty.

The intersection of emptyBag and bag1 is

The bag is empty.

\*\*\*Testing difference method\*\*\*

The difference of bag1 and bag2 is

X A A

There are 3 element(s) in the bag.

The difference of bag2 and bag1 is

B C D

There are 3 element(s) in the bag.

The difference of bag1 and emptyBag is

A A B X A C A

There are 7 element(s) in the bag.

The difference of emptyBag and bag1 is

The bag is empty.

\*\*\*Testing replace method\*\*\*

Bag1 contains:

A C A X B A A

There are 7 element(s) in the bag.

Replaced "A" with "X"

Now bag1 contains:

X C A X B A A

There are 7 element(s) in the bag.

Calling replace on emptyBag

The bag is empty and replace returned null - CORRECT

Now emptyBag contains:

The bag is empty.

\*\*\*Testing removeEvery method\*\*\*

Bag1 contains:

X C A X B A A

There are 7 element(s) in the bag.

Removing all "Z"

After removing all "Z" bag1 contains:

X C A X B A A

There are 7 element(s) in the bag.

Removing all "X"

After removing all "X" bag1 contains:

C A B A A

There are 5 element(s) in the bag.

After adding two "A" bag1 contains:

A A C A B A A

There are 7 element(s) in the bag.

Removing all "A"

After removing all "A" bag1 contains:

C B

There are 2 element(s) in the bag.

Removing all "B"

After removing all "B" bag1 contains:

C

There are 1 element(s) in the bag.

After removing all "C" emptyBag contains:

The bag is empty.

\*\*\* TESTING moveFirstToEnd \*\*\*

List before:

A B C

There are 3 element(s) in the bag.

List after:

B C A

There are 3 element(s) in the bag.

Calling moveFirstToEnd three times

List before:

A C B

There are 3 element(s) in the bag.

List after:

A C B

There are 3 element(s) in the bag.

Calling moveFirstToEnd on a list of length 0

List before:

The bag is empty.

List after:

The bag is empty.

Calling moveFirstToEnd on a list of length 1

List before:

B

There are 1 element(s) in the bag.

List after:

B

There are 1 element(s) in the bag.

Calling moveFirstToEnd on a list of length 2

List before:

B A

There are 2 element(s) in the bag.

List after:

A B

There are 2 element(s) in the bag.

\*\*\* TESTING findMiddleElementInOnePass \*\*\*

The bag is empty.

middle: null

A

There are 1 element(s) in the bag.

middle: A

B A

There are 2 element(s) in the bag.

middle: B

F E D C B A

There are 6 element(s) in the bag.

middle: D

G F E D C B A

There are 7 element(s) in the bag.

middle: D

\*\*\* TESTING checkIfLoopExists \*\*\*

testBag does not have a loop - CORRECT

bagWithLoop does have a loop - CORRECT

Process finished with exit code 0