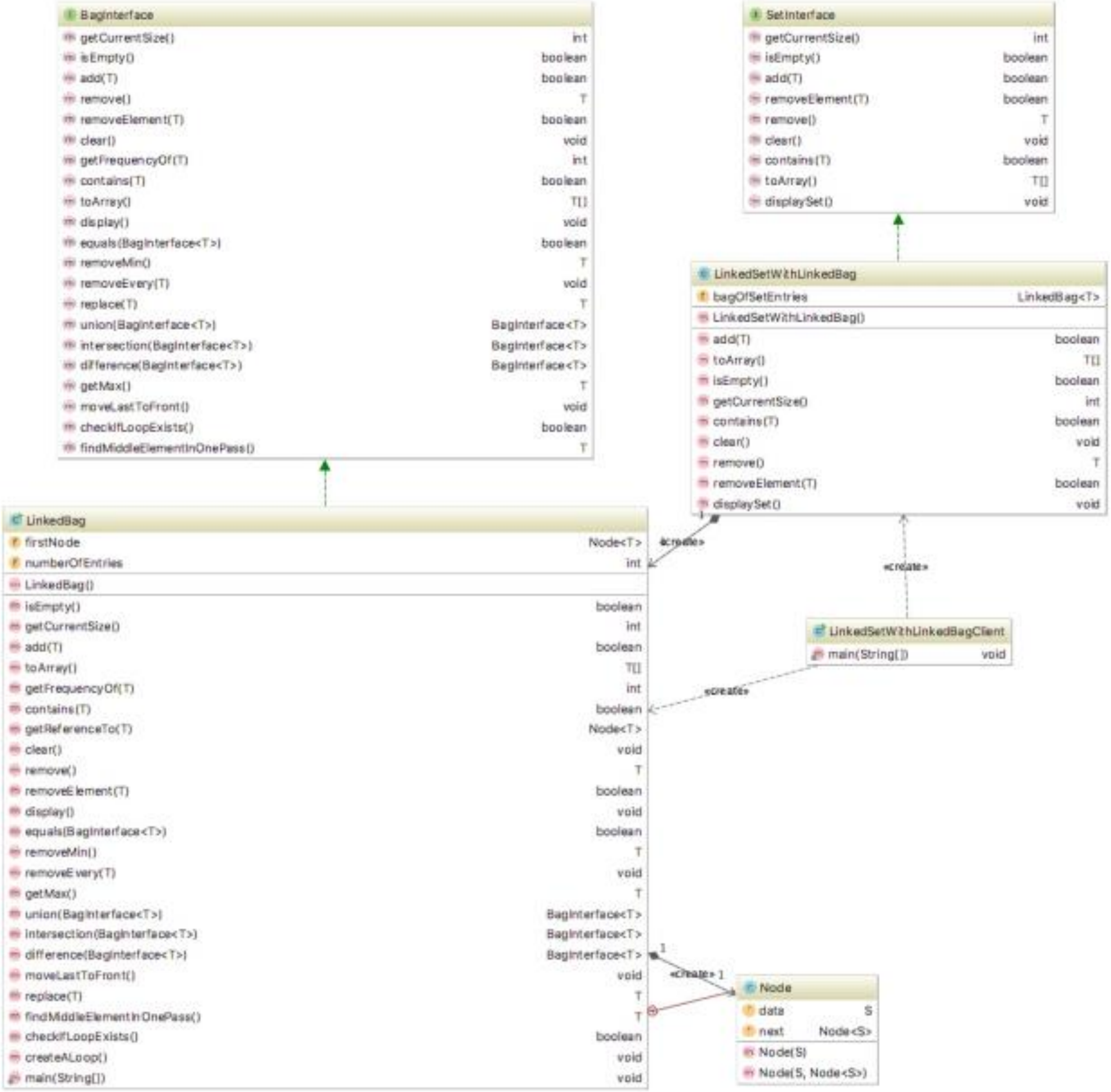
# Comp151 Lab03

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 provided client `LinkedSetWithLinkedBagClient`. See the sample run below. Load to IDEA only the classes needed for this project from the provided `Lab03.zip` file

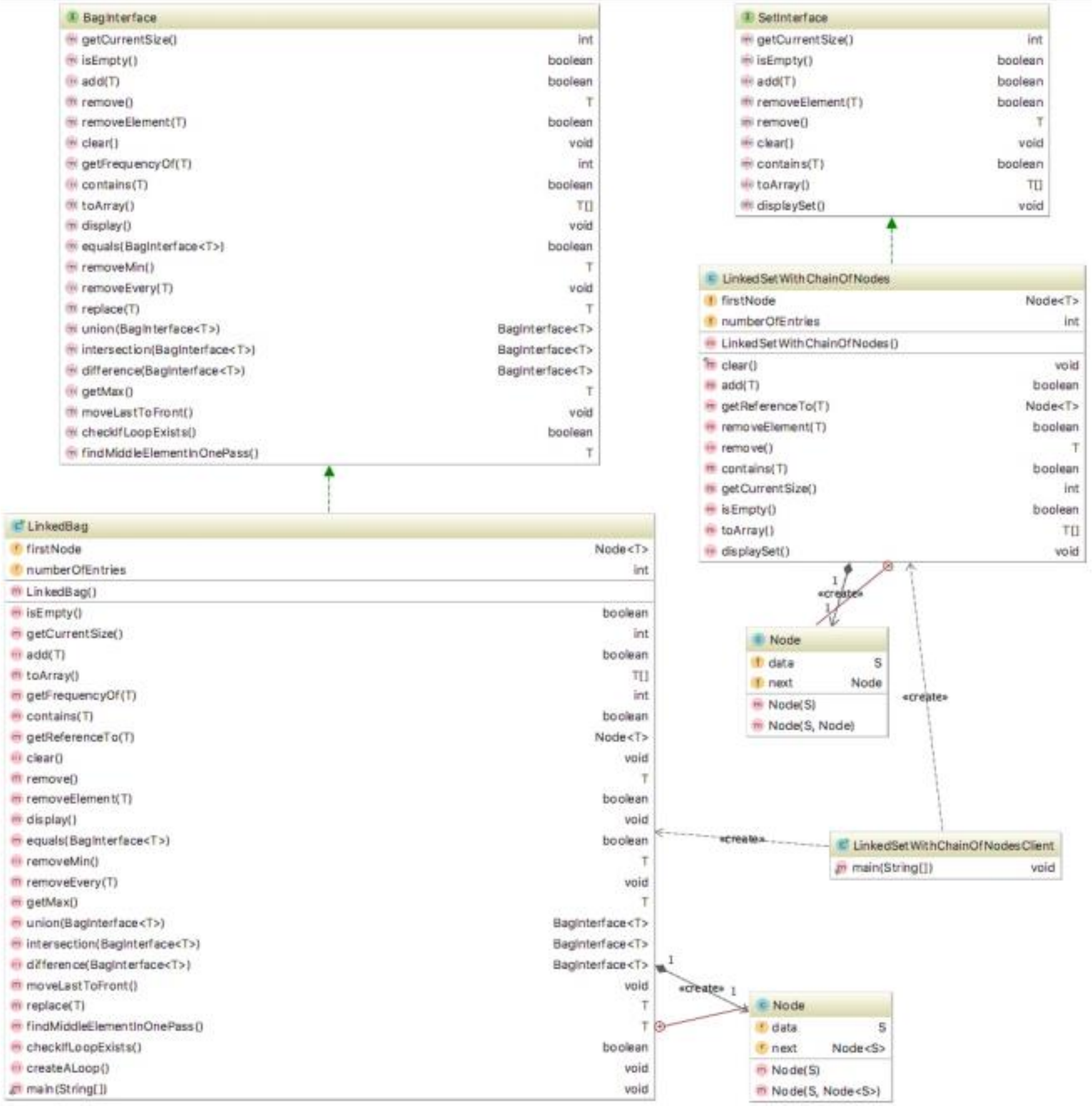


## Sample Run:

```
Creating aBag and aSet objects and adding elements from contentsOfBag to them
There are 8 element(s): B B C A D C B A
The set contains 4 string(s), as follows:
D C B A

Clearing aSet
The set is empty
aSet isEmpty returns true
The size of aSet is 0

Creating set1 and set2
The size of set1 is 0
The size of set2 is 0

Adding elements to set1
The size of set1 is 3
set1 is
The set contains 3 string(s), as follows:
C B A

Adding elements to set2
The size of set2 is 4
set2 is
The set contains 4 string(s), as follows:
D C B A

set1 contains A: true
set1 contains E: false
After removing B from set1,
The set contains 2 string(s), as follows:
C A
After removing C from set1,
The set contains 1 string(s), as follows:
A

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 provided client `LinkedSetWithChainOfNodesClient.`



## Sample Run:

```
Creating aBag and aSet objects and adding elements from contentsOfBag to them
There are 8 element(s): B B C A D C B A
The set contains 4 string(s), as follows:
D C B A

Clearing aSet
The set is empty
aSet isEmpty returns true
The size of aSet is 0

Creating set1 and set2
The size of set1 is 0
The size of set2 is 0

Adding elements to set1
The size of set1 is 3
set1 is
The set contains 3 string(s), as follows:
C B A

Adding elements to set2
The size of set2 is 4
set2 is
The set contains 4 string(s), as follows:
D C B A

set1 contains A: true
set1 contains E: false
After removing B from set1,
The set contains 2 string(s), as follows:
C A
After removing C from set1,
The set contains 1 string(s), as follows:
A

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
- **moveLastToFront -** moves the last item in the bag to be the first 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)
- **checkIfLoopExists –** checks if the chain-of-nodes has a loop in one pass (HINT: uses two pointers that move at different speed)

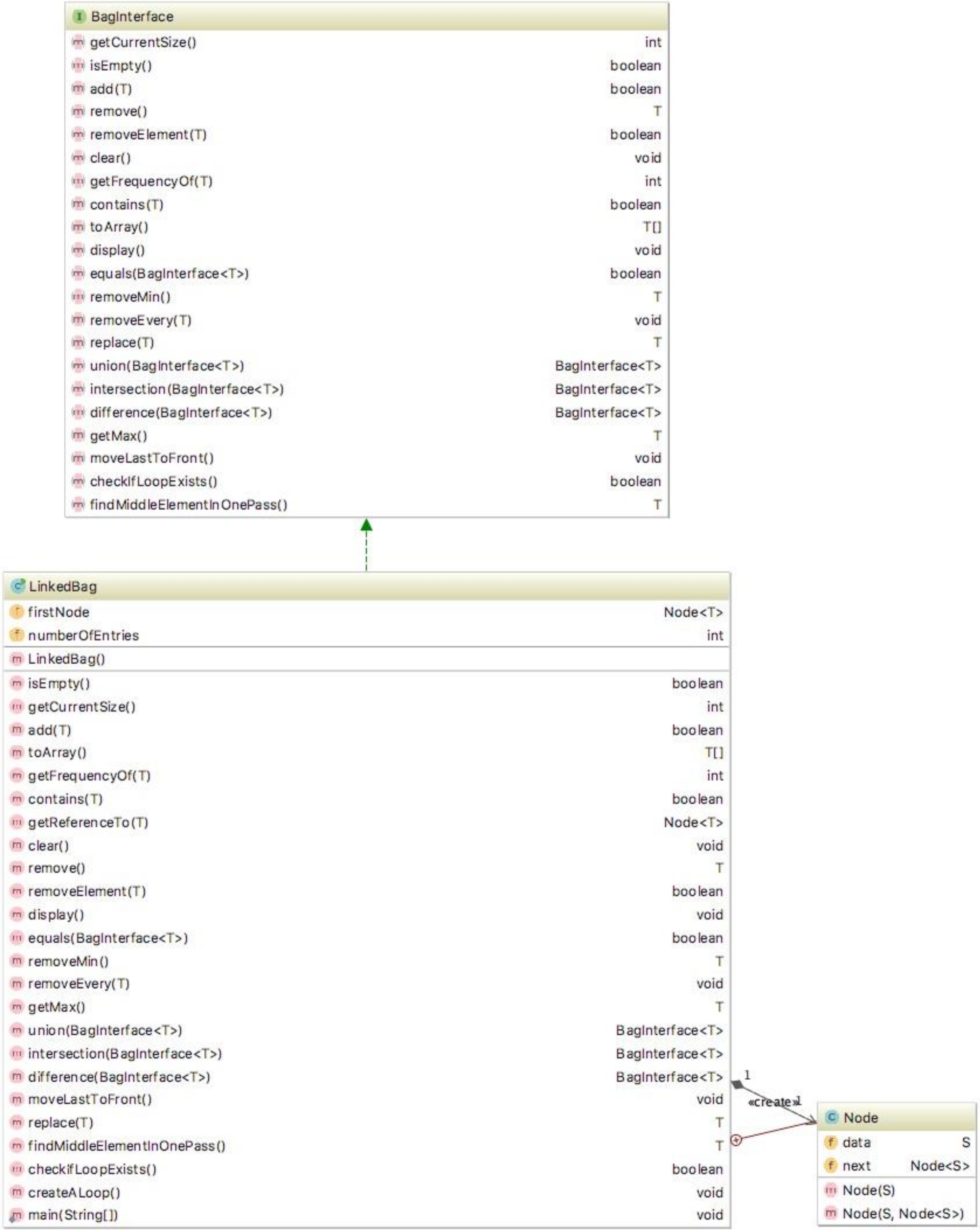
## Your Task:

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**:

   ```
   a.  public boolean equals(BagInterface<T> other);
   b.  public void removeEvery(T anEntry);
   c.  public T replace(T replacement);
   d.  public BagInterface <T> union(BagInterface <T> other);
   e.  public BagInterface <T> intersection(BagInterface <T> other);
   f.  public BagInterface <T> difference(BagInterface <T> other);
   g.  public T getMax();
   h.  public void moveLastToFront()
   i.  public T removeMin();
   j.  public boolean checkIfLoopExists();
   k.  public T findMiddleElementInOnePass();
   ```

5. Make sure that the output is correct (see Sample Run below).

## UML Diagram:


BagInterface
getCurrentSize() int
isEmpty() boolean
add(T) boolean
remove() T
removeElement(T) boolean
clear() void
getFrequencyOf(T) int
contains(T) boolean
toArray() T[]
display() void
equals(BagInterface<T>) boolean
removeMin() T
removeEvery(T) void
replace(T) T
union(BagInterface<T>) BagInterface<T>
intersection(BagInterface<T>) BagInterface<T>
difference(BagInterface<T>) BagInterface<T>
getMax() T
moveLastToFront() void
checkIfLoopExists() boolean
findMiddleElementInOnePass() T
LinkedBag
firstNode Node<T>
numberOfEntries int
LinkedBag()
isEmpty() boolean
getCurrentSize() int
add(T) boolean
toArray() T[]
getFrequencyOf(T) int
contains(T) boolean
getReferenceTo(T) Node<T>
clear() void
remove() T
removeElement(T) boolean
display() void
equals(BagInterface<T>) boolean
removeMin() T
removeEvery(T) void
getMax() T
union(BagInterface<T>) BagInterface<T>
intersection(BagInterface<T>) BagInterface<T>
difference(BagInterface<T>) BagInterface<T>
moveLastToFront() void
replace(T) T
findMiddleElementInOnePass() T
checkIfLoopExists() boolean
createALoop() void
main(String[]) void
1
«create»
1
Node
data S
next Node<S>
Node(S)
Node(S, Node<S>)

## Sample Run:

```
RUNNING TEST CASES

***Testing display method***
bag1 is
There are 5 element(s): B C A B A
bag2 is
The bag is empty.
After removing the first element B from bag1, it contains
There are 4 element(s): C A B A

***Testing equals method***
Are bag1 and bag2 equal? --> NO
Are bag2 and bag1 equal? --> NO
bag2:
There are 5 element(s): X B A C A
Are bag1 and bag2 equal? --> NO
Removed X from bag2.
There are 4 element(s): B A C A
Are bag1 and bag2 equal now? --> NO
Created bagCopyOfBag1:
There are 4 element(s): C A B A
Are bag1 and bagCopyOfBag1 equal? --> YES

***Testing getMax method***
bag1: There are 7 element(s): A C A X B A A
bag2: There are 7 element(s): D C C A B B A
The largest item in bag1 is: X
The largest item in bag2 is: D

***Testing union, removeMin, intersection, difference and subset methods***
bag1: There are 7 element(s): A C A X B A A
bag2: There are 7 element(s): D C C A B B A

***Testing union method***
The union of bag1 and bag2 is
There are 14 element(s): A B B A C C D A A B X A C A

***Testing removeMin method***
Removed the smallest element "A" from the union bag; the current content is:
There are 13 element(s): B B A C C D A A B X A C A
Removed the smallest element "A" from the union bag; the current content is:
There are 12 element(s): B B C C D A A B X A C A
Removed the smallest element "A" from the union bag; the current content is:
There are 11 element(s): B C C D B A B X A C A
Clearing the bag
The bag is empty and removeMin returned null - CORRECT

***Testing intersection method***
The intersection of bag1 and bag2 is
There are 4 element(s): B A C A

***Testing difference method***
The difference of bag1 and bag2 is
There are 3 element(s): X A A
The difference of bag2 and bag1 is
There are 3 element(s): B C D

***Testing replace method***
Bag1 contains:
There are 7 element(s): A C A X B A A
Replaced "A" with "X"
Now bag1 contains:
There are 7 element(s): X C A X B A A

***Testing removeEvery method***
Bag1 contains:
There are 7 element(s): X C A X B A A
Removing all "Z"
After removing all "Z" bag1 contains:
There are 7 element(s): X C A X B A A
Removing all "X"
After removing all "X" bag1 contains:
There are 5 element(s): C A B A A
After adding two "A" bag1 contains:
There are 7 element(s): A A C A B A A
Removing all "A"
After removing all "A" bag1 contains:
There are 2 element(s): C B
```

```
Removing all "B"
After removing all "B" bag1 contains:
There are 1 element(s): C

*** TESTING moveLastToFront ***
List before:
There are 3 element(s): A B C
List after:
There are 3 element(s): C A B

Calling moveLastToFront three times
List before:
There are 3 element(s): A C B
List after:
There are 3 element(s): A C B

Calling moveLastToFront on a list of length 0
List before:
The bag is empty.
List after:
The bag is empty.

Calling moveLastToFront on a list of length 1
List before:
There are 1 element(s): B
List after:
There are 1 element(s): B

Calling moveLastToFront on a list of length 2
List before:
There are 2 element(s): B A
List after:
There are 2 element(s): A B


*** TESTING findMiddleElementInOnePass ***
The bag is empty.
middle: null
There are 1 element(s): A
middle: A
There are 2 element(s): B A
middle: B
There are 6 element(s): F E D C B A
middle: D
There are 7 element(s): G F E D C B A
middle: D

*** TESTING checkIfLoopExists ***
testBag does not have a loop - CORRECT
bagWithLoop does have a loop - CORRECT

Process finished with exit code 0
```