Comp151 Lab02

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

Project1

Define a class `ArraySetWithResizableArrayBag` that represents a set and implements the `SetInterface` (described in Segment 1.21 of chapter 1). Use the class `ResizableBag` in your implementation as defined in the UML diagram below. Test your class with the provided client `ArraySetWithResizableArrayBagClient` . Load to IDEA only the classes needed for this project from the provided `Lab02.zip` file. Make sure that the DEFAULT_CAPACITY in `ResizableArrayBag` is set to 3.

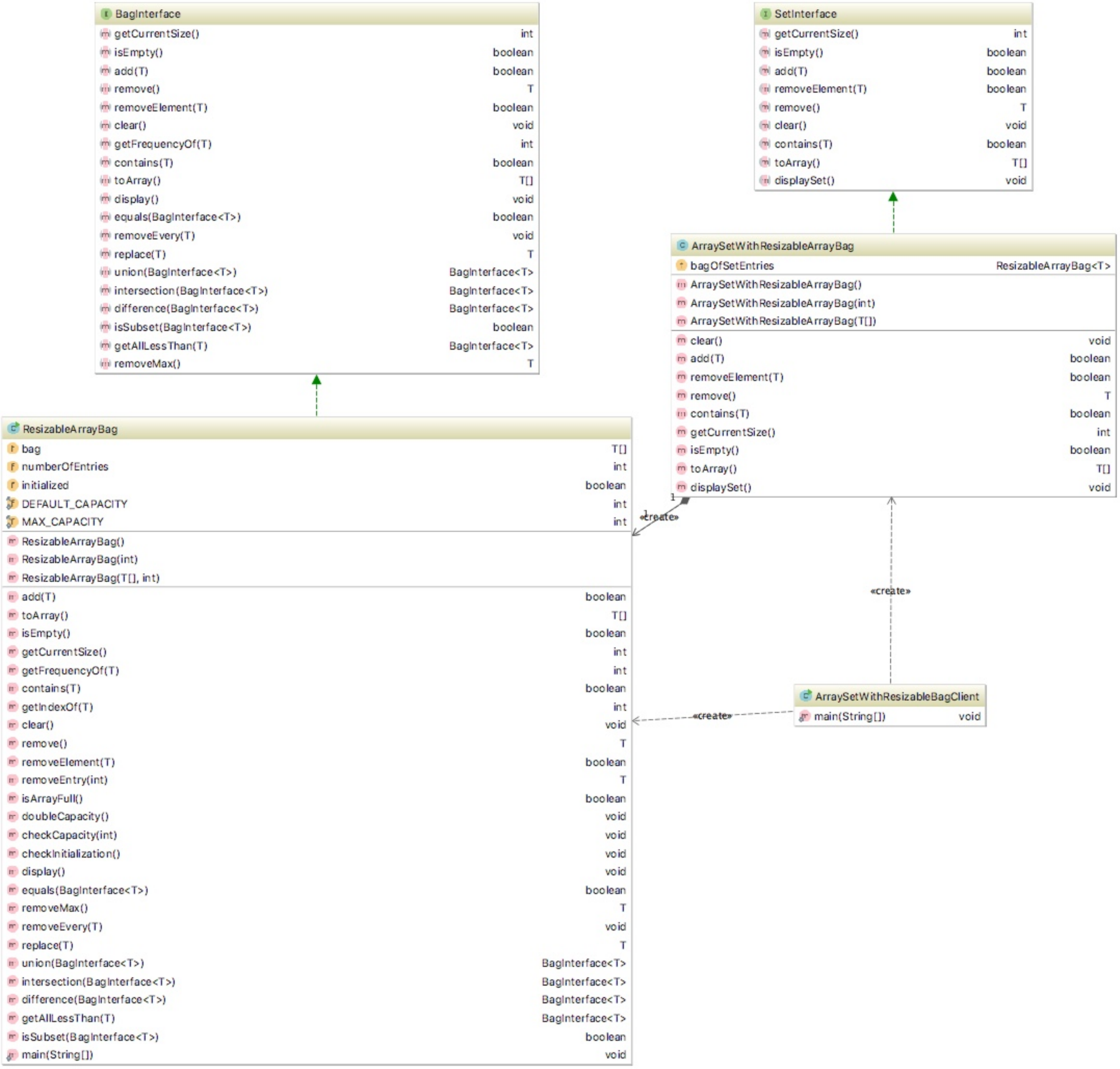
Most of the methods can be implemented by simply calling appropriate methods from the `ResizableArrayBag`, for example the `clear` method would be implemented as:

```
public void clear()
{
    this.bagOfSetEntries.clear();
}
```

Ensure that your `add` method does not allow duplicates.

Note that the third constructor: `public ArraySetWithResizableArrayBag(T[] contents)` must call your `add` method to populate `this.bagOfSetEntries` with elements from the passed `contents` array (check for `null` elements!).

See sample run below.



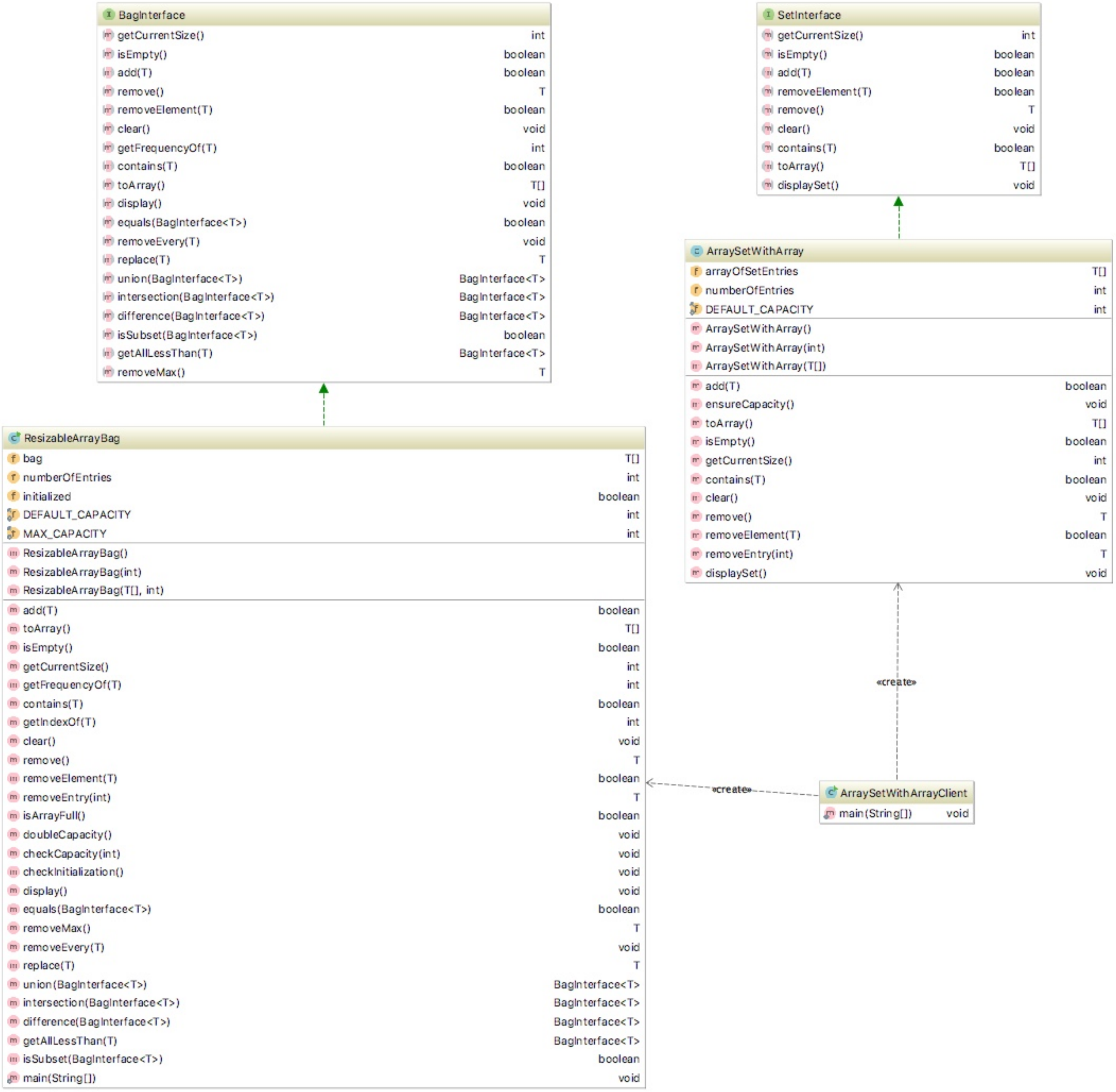
Sample Run:

```
There are 8 element(s): A B C D A C B B
Creating aSet with the secondary constructor - capacity of 8
Adding elements from aBag to aSet
The set contains 4 string(s): B C A D

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

Creating set1 with default constructor

set1 initially empty, capacity of 3:
The set is empty
```

```
Adding elements to set1

set1 after adding elements:
The set contains 3 string(s): A B C

set1 after adding and resizing:
The set contains 10 string(s): A B C V T U W X Y Z

Creating set2 with the content of contentsOfBag array
Adding more elements to set2
set2 after adding and resizing:
The set contains 8 string(s): A B C D E F G H

set1 contains A: true
set1 contains E: false
After removing B from set1:
The set contains 9 string(s): A Z C V T U W X Y
After removing Y from set1:
The set contains 8 string(s): A Z C V T U W X

Process finished with exit code 0
```

Project2

Load to the IDEA the remaining classes from the provided `Lab02.zip` file. Repeat the previous project inside the `ArraySetWithArray` class, but this time use a dynamically resizable array instead of the `ResizableArrayBag` as defined in the UML diagram below. Test your class with the provided client `ArraySetWithArrayClient` , See sample run below.



Sample Run:

```
There are 8 element(s): A B C D A C B B
Creating aSet with the secondary constructor - capacity of 8
Adding elements from aBag to aSet
The set contains 4 string(s): B C A D

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

Creating set1 with default constructor

set1 initially empty, capacity of 3:
The set is empty

set1 after adding elements:
The set contains 3 string(s): A B C

Adding elements to set1
-->Resizing this.arrayOfSetEntries from 3 to 6
-->Resizing this.arrayOfSetEntries from 6 to 12

set1 after adding and resizing:
The set contains 10 string(s): A B C V T U W X Y Z

Creating set2 with the content of contentsOfBag array
Adding more elements to set2

set2 after adding and resizing:
The set contains 8 string(s): A B C D E F G H

set1 contains A: true
set1 contains E: false
After removing B from set1:
The set contains 9 string(s): A Z C V T U W X Y
After removing Y from set1:
The set contains 8 string(s): A Z C V T U W X

Process finished with exit code 0
```

Project3

Consider a bag as a way to organize data. A grocery bag, for example, contains items in no particular order. Some of them might be duplicate items. The `ADT bag`, like a grocery bag, is perhaps the simplest of data organizations. It holds objects but does not arrange or organize them further.

The attached `BagIterface.java` contains `java interface` that defines all the operations that can be performed on `ADT bag`. Please note that the interface has been expanded from its definition in Chapter 1; javadoc comments describe operation's purpose, parameters and return values.
In addition to these basic operations, the following are included:

- **union** - combines the contents of two bags into a third bag (see Exercise 5 in Chapter 1)
- **intersection** - creates a third bag of only those items that occur in both two bags (see Exercise 6 in Chapter 1)
- **difference** - 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)
- **getAllLessThan** - creates a bag of all the items that are smaller than the given item
- **removeMax** – removes and returns the largest element from the bag
- **isSubset** - returns true if all the elements of the given bag are also included in the other bag
- **equals** - returns true if the contents of two bags are the same. Note that two equal bags contain the same number of entries, each entry occurs in each bag in the same position in the "collection of objects"

Your Task:

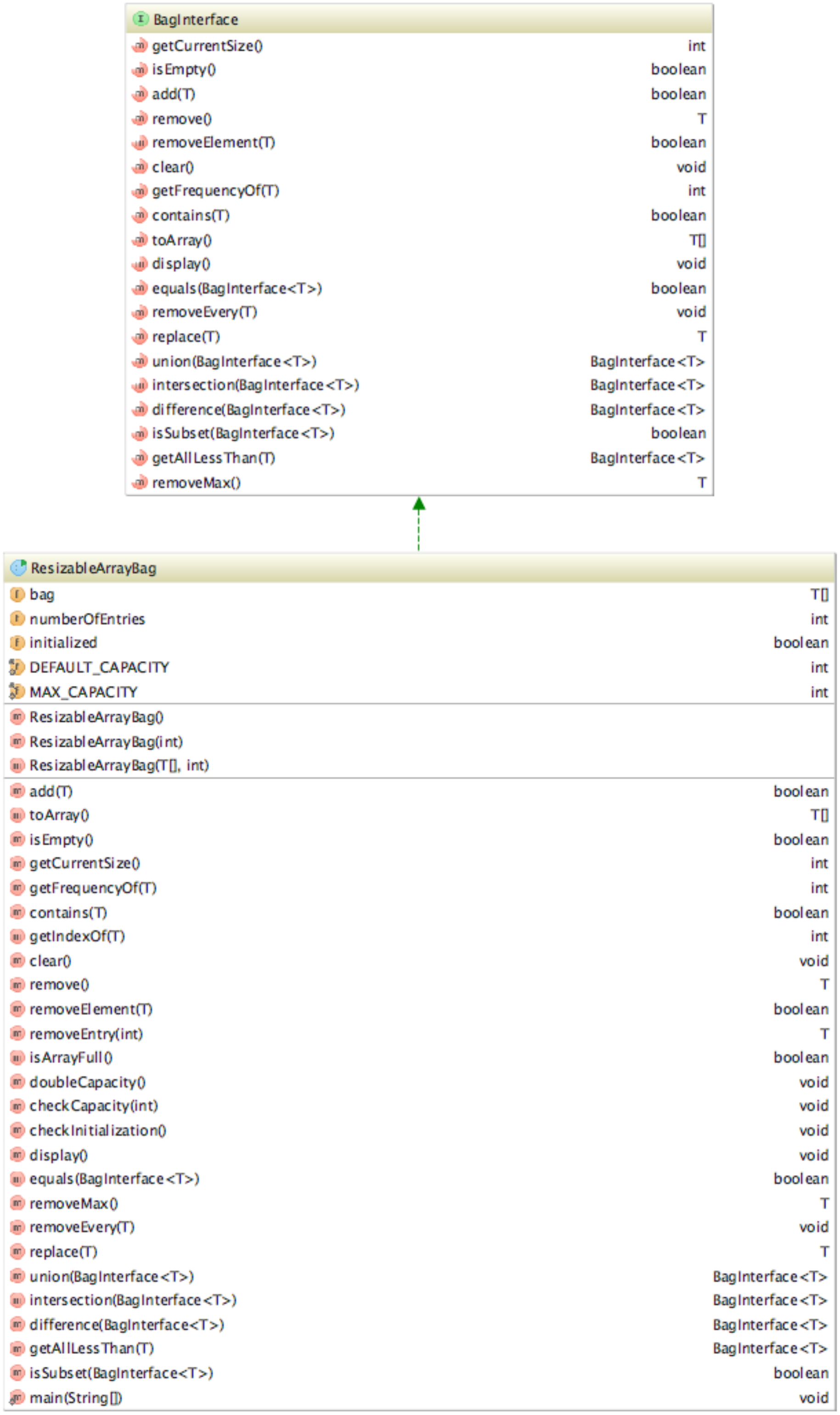
1. The class `ResizableArrayBag` implements the interface `BagInterface` where the bag is represented as an array that can expand dynamically as necessary. Make sure that the DEFAULT_CAPACITY in `ResizableArrayBag` is set to 25.
2. Analyze provided interface including javadoc comments that describe the 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 `ResizableArrayBag` class. Note that the `main` is included in this class as well.
4. Implement the remaining methods that are "stubs" at this moment:

```
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 boolean isSubset(BagInterface <T> other);
h.      public BagInterface <T> getAllLessThan(T anEntry);
i.      public T removeMax();
```

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

NOTE: Efficiency of your algorithms will also be graded. Even though your method produces correct results it is possible that some points may be deducted if the code is not efficient.

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
removeEvery(T) void
replace(T) T
union(BagInterface<T>) BagInterface<T>
intersection(BagInterface<T>) BagInterface<T>
difference(BagInterface<T>) BagInterface<T>
isSubset(BagInterface<T>) boolean
getAllLessThan(T) BagInterface<T>
removeMax() T
ResizableArrayBag
bag T[]
numberOfEntries int
initialized boolean
DEFAULT_CAPACITY int
MAX_CAPACITY int
ResizableArrayBag()
ResizableArrayBag(int)
ResizableArrayBag(T[], int)
add(T) boolean
toArray() T[]
isEmpty() boolean
getCurrentSize() int
getFrequencyOf(T) int
contains(T) boolean
getIndexOf(T) int
clear() void
remove() T
removeElement(T) boolean
removeEntry(int) T
isArrayFull() boolean
doubleCapacity() void
checkCapacity(int) void
checkInitialization() void
display() void
equals(BagInterface<T>) boolean
removeMax() T
removeEvery(T) void
replace(T) T
union(BagInterface<T>) BagInterface<T>
intersection(BagInterface<T>) BagInterface<T>
difference(BagInterface<T>) BagInterface<T>
getAllLessThan(T) BagInterface<T>
isSubset(BagInterface<T>) boolean
main(String[]) void

Sample Run:

```
RUNNING TEST CASES
Created bag1:
There are 5 element(s): C A A A X
Created bag2:
The bag is empty
After removing the last element X from bag1, it contains:
There are 4 element(s): C A A A

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

***Testing union, intersection, difference, removeMax, getAllLessThan and isSubset methods***
bag1:
There are 5 element(s): C A A X A
bag2:
There are 7 element(s): A B B A C C D

***Testing union method***
The union of bag1 and bag2 is:
There are 12 element(s): C A A X A A B B A C C D

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

***Testing intersection method***
The intersection of bag1 and bag2 is:
There are 3 element(s): C A A

***Testing difference method***
The difference of bag1 and bag2 is:
There are 2 element(s): A X
The difference of bag2 and bag1 is:
There are 4 element(s): C B B D

***Testing getAllLessThan method***
The following entries in bag1 are smaller than "Z":
There are 5 element(s): C A A X A
The following entries in bag2 are smaller than "C":
There are 4 element(s): A B B A

***Testing isSubset method***
Is bag1 a subset of bag1 ? --> YES
Is bag1 a subset of bag2 ? --> NO
Is an empty bag a subset of bag2 ? --> YES
Is bag2 a subset of an empty bag ? --> NO
Created bag3:
There are 3 element(s): A B C
Created bag4:
There are 3 element(s): B C A
Is bag3 a subset of bag4 ? --> YES
Is bag3 a subset of bag4 after adding "Z" to it ? --> YES
Is bag4 a subset of bag3 ? --> NO
Adding  "Z" to bag 3 twice
bag3:
There are 5 element(s): A B C Z Z
bag4:
There are 4 element(s): B C A Z
Is bag3 a subset of bag4 ? --> NO

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

***Testing removeEvery method***
bag1:
There are 7 element(s): A A B X A C X
Removing all "Z"
After removing all "Z" bag1 contains:
There are 7 element(s): A A B X A C X
Removing all "A"
After removing all "A" bag1 contains:
There are 4 element(s): X C B X
Removing all "X"
After removing all "X" bag1 contains:
There are 2 element(s): B C

Process finished with exit code 0
```