

**Mathematics for Computers**  
**(Discrete Mathematics)**

**Setty Operations – Set Theory**

**Project Report**

***By***

**Hesham Medhat & Merit Victor**

# **Set Theory Project Report**

**CS211 Course, Faculty of Engineering,  
Alexandria University.**

**Computer & Systems Engineering  
Department.**

**To Professor: Sahar M. Ghanem.**

**and Teaching Assistants:**

- **Eng/ Reham Osama.**

# Index

1. Introduction	4
2. Overview	5
3. Features	5
4. Packages	6
5. Data Structures	7
6. Functions	10
7. UML Diagram	15
8. Algorithms (Pseudocode)	16
9. Sample Runs	21
10. Assumptions	32

# Introduction

This project was assigned in the CS211: Mathematics for Computers course in the date: Friday, September 22<sup>nd</sup>, 2017.

Due on: Saturday, September 30<sup>th</sup>, 2017.

As the first project of the course, we were assigned this project as teams of two.

This delivery is by:

***Hesham Medhat Mahmoud Ahmed Abou-Mousa***

&

***Merit Victor Ageeb Saweeres Toussy***

The source of this project is available on github on this link:

[github.com/hesham-medhat/Setty\\_Operations/](https://github.com/hesham-medhat/Setty_Operations/)

## **Overview**

This program works to perform operations on sets of strings.

The program asks for the universe at first. Gets input sets, and performs operations on the on user's desire.

## **Features**

*In our implementation,* the program can:

- find the complement of a set.
- find the union of two sets.
- find the intersection of two sets.
- find the difference of two sets.
- display the output even if it's phi (empty set).
- display input sets and stores them for operations.
- Graphical user interface (GUI).

# **Packages**

## **A. Package: application:**

This package contains all the files related to the GUI using JavaFx.

- Controller class: Main.java.
- Twofxml files: application.fxml and layout.fxml
- Styling sheets: application.css.

## **B. Package: Sets:**

This package contains all the files related to the sets implementation and operations.

- Abstract class: Set.java.
- First child: Universe.java.
- Second child: Subset.java.

Their names are self-commenting. Go to “UML Diagram” section for visualization.

## **C. Package: LinkedLists:**

This package contains LinkedList implementation. Go to “Data Structures” section.

# Data Structures

As we have learned the course Data Structures-1 in the previous semester, we are using our own implementations of Data Structures in this project.

We are using Singly Linked Lists, of our own implementation.

How we used them is described thoroughly in the respective interfaces in the “Functions” section.

Here is the ILinkedList interface:

```
ILinkedList {  
    /**  
     * Inserts a specified element at the specified position in the  
     * list.  
     */  
    public void add(int index, Object element);  
  
    /** Inserts the specified element at the end of the list. */  
    public void add(Object element);  
}
```

/\*\* Returns the element at the specified position in this list.

\*/

public Object get(int index);

/\*\*

\* Replaces the element at the specified position in this list

\* with the specified element.

\*/

public void set(int index, Object element);

/\*\* Removes all of the elements from this list. \*/

public void clear();

/\*\* Returns true if this list contains no elements. \*/

public boolean isEmpty();

/\*\* Removes the element at the specified position in this list.

\*/

public void remove(int index);



```

    /** Returns the number of elements in this list. */
    public int size();

    /**
     * Returns a view of the portion of this list between the
     * specified
     * fromIndex and toIndex, inclusively.
     */
    public ILinkedList sublist(int fromIndex, int toIndex);

    /**
     * Returns true if this list contains an element with the same
     * value as the specified element.
     */
    public boolean contains(Object o);
}

```

# Functions

- The main idea for the functions is that each subset of the universe has a Boolean array of length equal to the length of the universe. Each element in this array represents a true/false value for whether the mirroring element in the universe exists in this subset or not.
- Set operations become easier this way where intersections are found by AND-ing these bits/values. Similarly union is found by OR-ing. Complements are found by negating.
- We have also implemented “difference” which is important in set operations.
- The complexity of these functions is all in Big-O-of (n). This gives them linear time performance.
- We shall mention the functions in the “Sets” package and that they do.

## Set:

- Set(**final** String[] setInput)

Main constructor when reading input.

**param** setInput the input array of strings.

- **Set**(**final**SinglyLinkedList list)

Constructor in case the list is ready.

**param** list : previously built set.

- **Set**(**final** Universe universeIn, **finalboolean**[] setBoolIn)

Constructor in case we know the boolean set.

**param** universeIn : universe of the set.

**param**setBoolIn : boolean array of existence of elements from universe.

- **boolean** isUnique(**final** Object element, **final** SinglyLinkedListsetSLL)

Auxiliary function used for detecting whether the input element is unique or a duplicate before adding it to the SLL.

**param** element : to be added to the list.

**param** setSLL : the list.

**returns** true if it is unique and false otherwise.

- **abstract** Set complement()

Finds the rest of the elements in the universe not existing in the set.

**returns**complement of a set Returns null if the output set is empty.

- **abstract** Set difference(Set other)

Finds the set difference with another set.

**param** other : input set.

**returns** the difference. Returns null if the output set is empty.

- **abstract** Set intersection(Set other)

Gets the intersection of this set and another.

**param** other : input set.

**returns** intersection set. Returns null if the output set is empty.

- **abstract** Set union(Set other)

Gets the union of this set and another.

**param** other : input set.

**returns** union set. Returns null if the output set is empty.

- **SinglyLinkedList** getSetList()

Getter for setList.

**returns** setList as SLL.

## **Universe:**

This class inherits from “Set” class and implements its abstract methods.

- Universe(**final**SinglyLinkedList list)

Constructor in case the list of elements is ready.

**param** list : previously built list of elements.

- Universe(**final** String[] setInput)

Constructor that passes the setInput as string array.

**param** setInput : in the form of a string array.

## **Subset:**

This class inherits from “Set” class and implements its abstract methods.

- Subset(**final**SinglyLinkedList list, **finalboolean**[] setBoolIn)

Constructor in case we already have the list built.

**param** list : of elements in the subset.

- Subset(**final** Universe universeIn, **finalboolean**[] setBoolIn)

Constructor in case the boolean array is ready.

**param** universeIn : universe

**param** setBoolIn : the readySetBool

- **Subset**(**final** Universe universeIn, **final** String[] setInput)

Constructor that calls the super "Set" constructor to build the SLL of set.

**param** universeIn : universe as object.

**param** setInput : set content input as string array.

- **boolean[]** getSetBool()

Getter for setBool.

**returns** setBool which acts as a bit map for the existence of the elements in this set in the universe that it belongs to.

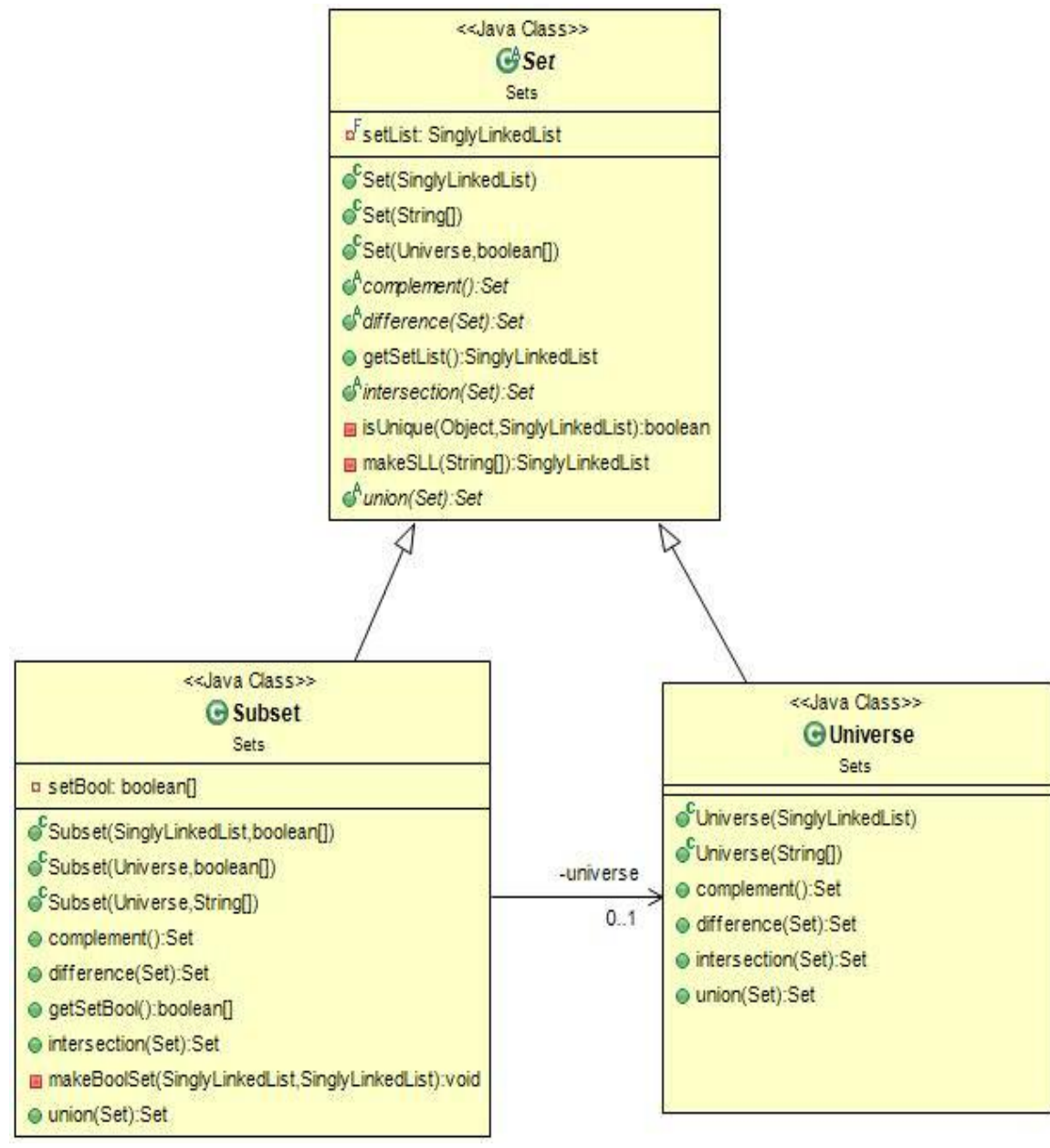
- **void** makeBoolSet(**final** SinglyLinkedList universe, **final** SinglyLinkedList set)

Constructs the setBool to be ready for operations.

**param** universe : in a SLL form.

**param** set : in a SLL form.

# UML Diagram



# Algorithms (Pseudocode)

- **Operations on Universe:**

- **Complement:**

- ```
complement() {  
    return null representing empty set (Phi).  
}
```

- **Union:**

- ```
union(final Set other) {  
    return new Universe object with the same  
    data.  
}
```

- **Intersection:**

- ```
intersection(final Set other) {  
    check whether the other set is Universe  
    if it's universe  
        return new Universe object with  
        the same data.  
    else if it's subset  
        return new Subset object with  
        the same data as "other".  
    End if.  
}
```

- **Difference:**

- ```
difference(final Set other) {  
    check whether the other set is Universe  
    if it's universe  
        return null representing empty  
        set (Phi).  
    else if it's subset  
        call the complement function of  
        the other set.  
    End if.}
```



- **Operations on Subset:**

- **Complement:**

```
complement() {  
    make new array of Booleans.  
    Declare Boolean "isUniverse" to make sure  
    this is not universe and initialize it to  
    true.  
    Get the head of the stored universe's list  
    of elements.  
  
    For I = 0 -> this Boolean array of length  
        If this element equals false  
            Then this isn't universe  
            Set "isUniverse" to false.  
            Set the element's index in the  
            new array of Booleans to true.  
        End if.  
        Get next node in the SLL.  
    End for loop.  
  
    Check value of "isUniverse"  
    If false  
        Return new object of subset with  
        the data stored in the array of  
        Booleans.  
    Else if true  
        Return null representing empty  
        set ( $\phi$ ).  
    End if  
}
```

- **Union:**

```
union(final Set other) {  
    if this other is universe  
        return it.  
    Else  
        Make new array of Booleans with same  
        size as this subset.  
        Declare Boolean "isEmpty" and  
        initialize it to true.  
  
    For I = 0 -> the Boolean array length  
        If OR-ing both elements of this'  
        Boolean array and the other's  
        Boolean array results true  
            Set "isEmpty" to false.  
            Set the element's index to  
            true in the new array of  
            Booleans.  
        Else  
            Set the element's index to  
            false in the new array of  
            Booleans.  
        End if  
    End for loop.  
  
    Check value of "isEmpty"  
    If false  
        Return new subset object  
        with the data stored in the  
        new array of Booleans.  
    Else  
        Return null representing  
        empty set ( $\phi$ ).  
    End if  
End if  
}
```

- **Intersection:**

```
intersection(final Set other) {  
    if this other is universe  
        return this subset.  
    Else  
        Make new array of Booleans with same  
        size as this subset.  
        Declare Boolean "intersected" and  
        initialize it to false.  
  
        For I = 0 -> the Boolean array length  
            If AND-ing both elements of this'  
            Boolean array and the other's  
            Boolean array results true  
                Set "intersected" to true.  
                Set the element's index to  
                true in the new array of  
                Booleans.  
            Else  
                Set the element's index to  
                false in the new array of  
                Booleans.  
            End if  
        End for loop.  
  
        Check value of "intersected"  
        If true  
            Return new subset object  
            with the data stored in the  
            new array of Booleans.  
        Else  
            Return null representing  
            empty set ( $\phi$ ).  
        End if  
    End if  
}
```

- **Difference:**

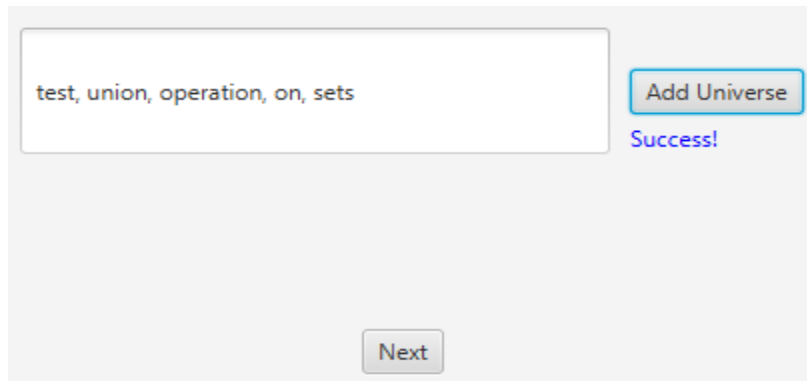
```
difference(final Set other) {
    if this other is universe OR it equals our
    subset
        return null representing empty set
        (phi).
    Else
        Make new array of Booleans with same
        size as this subset.
        Declare Boolean "isEmpty" and
        initialize it to true.

        For I = 0 -> the Boolean array length
            If this set's element exist and
            the other's doesn't
                Set "isEmpty" to false.
                Set the element's index to
                true in the new array of
                Booleans.
            Else
                Set the element's index to
                false in the new array of
                Booleans.
            End if
        End for loop.

        Check value of "intersected"
        If false
            Return new subset object
            with the data stored in the
            new array of Booleans.
        Else
            Return null representing
            empty set (phi).
        End if
    End if
}
```

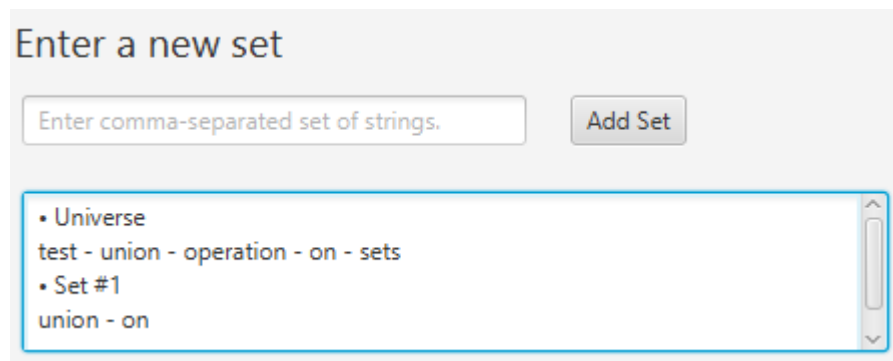
# Sample Runs

## Test Union operation:



A screenshot of a web application interface. At the top, there is a text input field containing the string "test, union, operation, on, sets". To the right of this field is a button labeled "Add Universe". Below the button, the word "Success!" is displayed in blue text. At the bottom center of the interface is a button labeled "Next".

- User inserts new universe and Next button is enabled.



A screenshot of a web application interface titled "Enter a new set". It features a text input field with the placeholder text "Enter comma-separated set of strings." and an "Add Set" button. Below the input field is a scrollable list box containing two items: "• Universe" followed by "test - union - operation - on - sets", and "• Set #1" followed by "union - on".

- User inserts first set.

• Universe  
 test - union - operation - on - sets  
 • Set #1  
 union - on  
 Set #2

First Set: 1  
 Complement

Second Set: 0  
 Complement

Union of first set and second set:  
 test - union - operation - on - sets

select 0 for universe

■ Union of subset and universe.

test - union - operation - on - sets  
 • Set #1  
 union - on  
 • Set #2  
 on - test

First Set: 1  
 Complement

Second Set: 2  
 Complement

Union of first set and second set:  
 test - union - on

select 0 for universe

■ Union of two subsets.

• Universe  
test - union - operation - on - sets

• Set #1  
union - on

First Set: 0    Second Set: 0

Union of first set and second set:  
test - union - operation - on - sets

select 0 for universe

■ Union of two identical universes.

## Test Complement operation:

• Universe  
test - union - operation - on - sets

• Set #1  
union - on

First Set: 1    Second Set: 2

First set's complement:  
test - operation - sets

■ Complement of subset.

• Universe  
test - union - operation - on - sets  
• Set #1  
union - on  
Set #2

First Set: 0    U    -    ∩    Second Set: 2

Complement    Complement

First set's complement:  
Phi - Empty set.

select 0 for universe

■ Complement of Universe.

Set #2  
on - test  
• Set #3  
test - union - operation - on - sets

First Set: 3    U    -    ∩    Second Set: 0

Complement    Complement

First set's complement:  
Phi - Empty set.

select 0 for universe

■ Complement of subset includes all elements in the universe.



## Test Intersection operation:

• Universe  
test - union - operation - on - sets  
• Set #1  
union - on  
Set #2

First Set: 1  
Second Set: 0

Buttons: U, -,  $\cap$  (selected), Complement

Intersection of first set and second set:  
union - on

select 0 for universe

■ Intersection of subset and universe.

• Set #2  
on - test  
• Set #3  
sets - test - union  
Set #4

First Set: 2  
Second Set: 3

Buttons: U, -,  $\cap$  (selected), Complement

Intersection of first set and second set:  
test

select 0 for universe

■ Intersection of two subsets.

on - test  
 • Set #3  
 sets - test - union  
 • Set #4  
 operation - on

First Set: 3  
 Complement

Second Set: 4  
 Complement

Intersection of first set and second set:  
 Phi - Empty set

■ Intersection of non-intersected subsets.

First Set: 0  
 Complement

Second Set: 0  
 Complement

Intersection of first set and second set:  
 test - union - operation - on - sets

select 0 for universe

■ Intersection of identical two universes.

## Test Difference operation:

First Set: 0  
Second Set: 0

Buttons: U, -, ∩

Buttons: Complement

Difference of first set from second set:  
Phi - Empty set.

select 0 for universe

- Difference between two identical universes.

First Set: 0  
Second Set: 2

Buttons: U, -, ∩

Buttons: Complement

Difference of first set from second set:  
union - operation - sets

select 0 for universe

- Difference of universe from subset.

• Universe  
 0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10  
 • Set #1  
 2 - 3 - 4 - 6 - 7

First Set: 1  
 Complement

U   -   ∩

Second Set: 0  
 Complement

Difference of first set from second set:  
 Phi - Empty set.

select 0 for universe

■ Difference of subset from universe.

First Set: 1  
 Complement

U   -   ∩

Second Set: 1  
 Complement

Difference of first set from second set:  
 Phi - Empty set.

■ Difference between two identical universes.

Set #1  
2 - 3 - 4 - 6 - 7  
• Set #2  
2 - 4

First Set  
2  
Complement

$\cup$   $-$   $\cap$

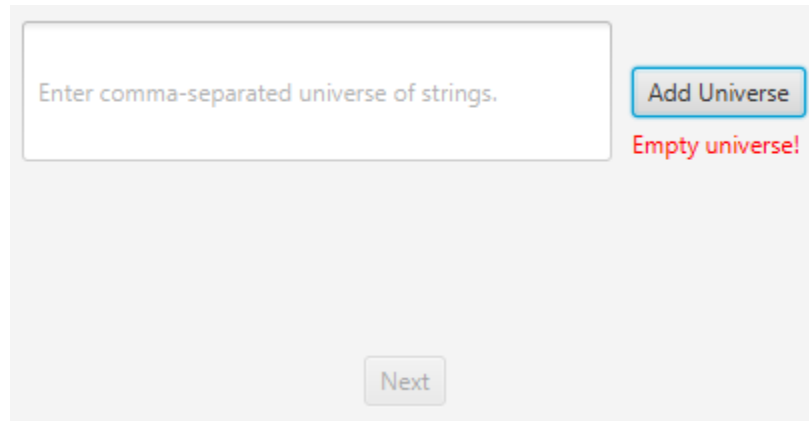
Second Set  
1  
Complement

Difference of first set from second set:  
Phi - Empty set.

select 0 for universe

- Difference of subset from another subset which is subset of it too.

## Special Cases:



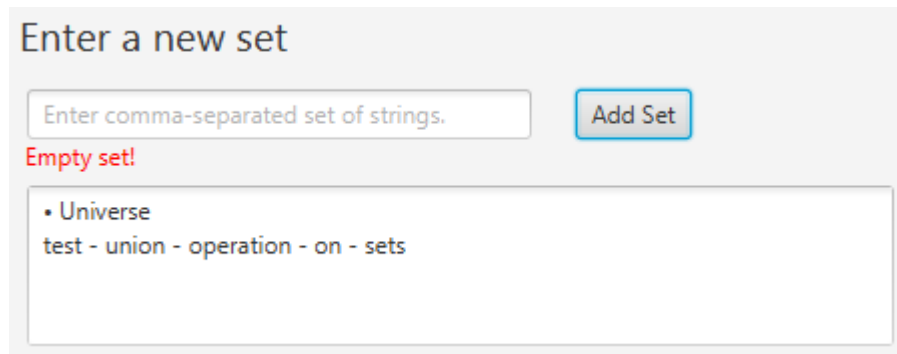
Enter comma-separated universe of strings.

Add Universe

Empty universe!

Next

- *Empty universe.*



Enter a new set

Enter comma-separated set of strings.

Add Set

Empty set!

- Universe

test - union - operation - on - sets

- **Empty set.**

### Enter a new set

Add Set

An element in the set doesn't exist in the universe!

- Set #1
  - union - on
- Set #2
  - on - test

- **Subset outside universe.**

Enter a new set

- Set #1
  - union - on
- Set #2
  - on - test

- **Remove duplicates and whitespaces.**

2.....4

Add Set

• Set #1  
2 - 3 - 4 - 6 - 7

• Set #2  
2 - 4

- Remove empty elements from sets.

# Assumptions

- The user inserts all sets in one line.
- Elements of the set are comma-separated strings.
- The user is allowed to insert only one universe.

*Thank you.*