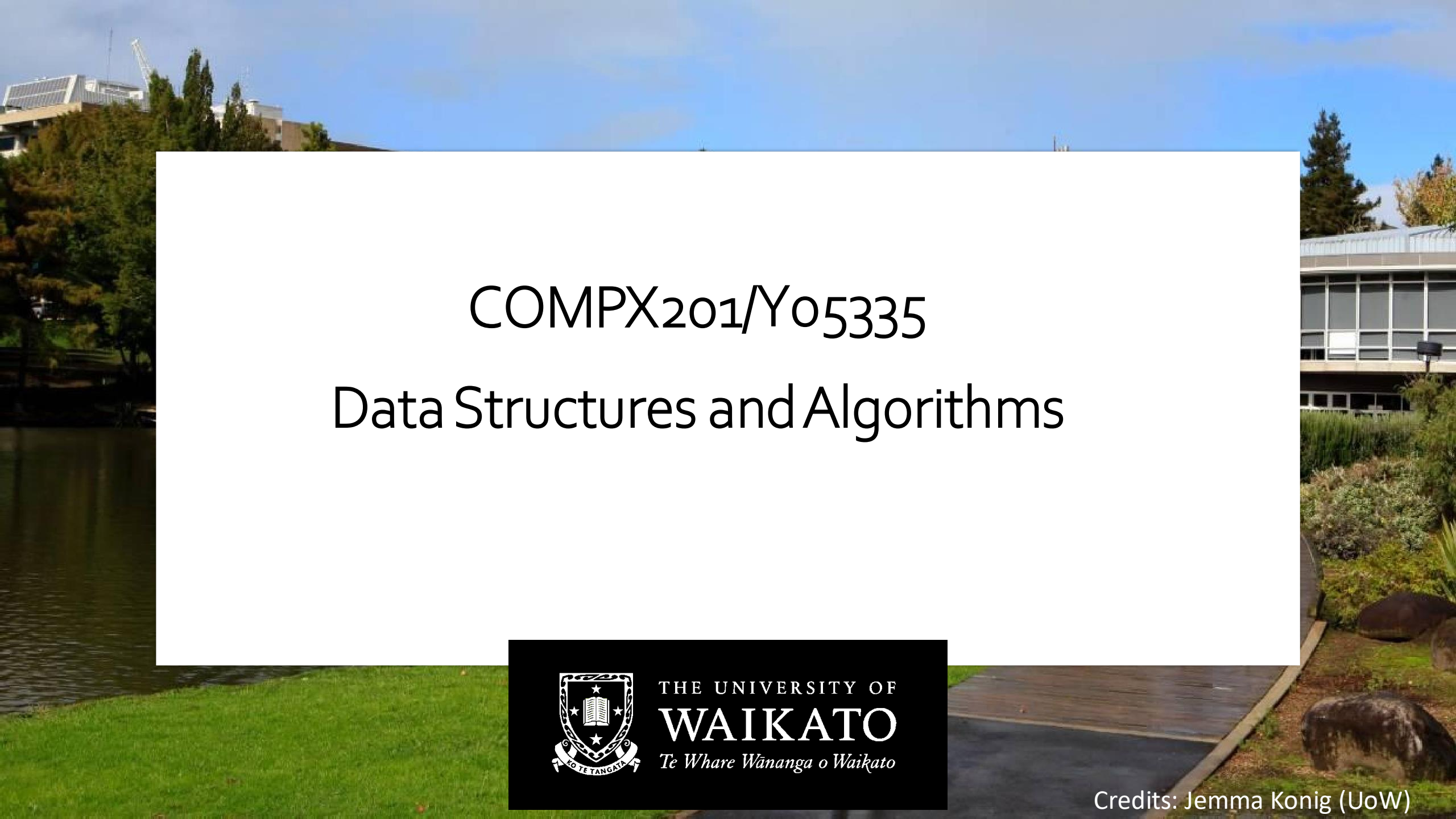


COMPX201/Y05335

Data Structures and Algorithms

THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

Credits: Jemma Konig (UoW)

Set Theory

COMPX201/Y05335

Overview

- Custom Data Structures
- Basic Set theory

Custom Data Structures

Custom Data Structures

- As we have seen, Java has a set of default data types and data structures
- e.g. Strings, Arrays etc.
- However, we may wish to implement our own data types/data structures.
- Throughout this course we will learn basic data structures (e.g. linked lists, sets, etc.)
- This will prepare you for writing custom data structures in the future.

Custom Data Structures

- Sets
- Linked lists
- Stacks
- Queues
- Hash tables
- Graphs

Basic Set Theory

Basic Set Theory: What is a set?

- A collection of unique elements of the same type (i.e. with no duplicates).

  $A = \{1, 2, 3, 4\}$

- We can say that 1 is an element of set A.

  $1 \in A$

- And that 5 is not an element of set A.

  $5 \notin A$

- There is a special set with no elements, this is called the empty set.

  $\{\} = \emptyset$

Basic Set Theory: Equality

- Two sets, A and B, are only equal if all the elements in A are also in B.

$A = \{1,2,3,4\}$ $B = \{4,5,6\}$ $C = \{1,2,3,4\}$

$A = C$

$A \neq B$

Basic Set Theory: Subset

- A set A is a subset of B if every element of A is an element of B.

  $A = \{1,2,3,4\}$ $\quad\quad$ $B = \{1,2,3,4,5\}$

  $A \subset B$

- Thus, $A=B$ if and only if $A \subset B$ and $B \subset A$.

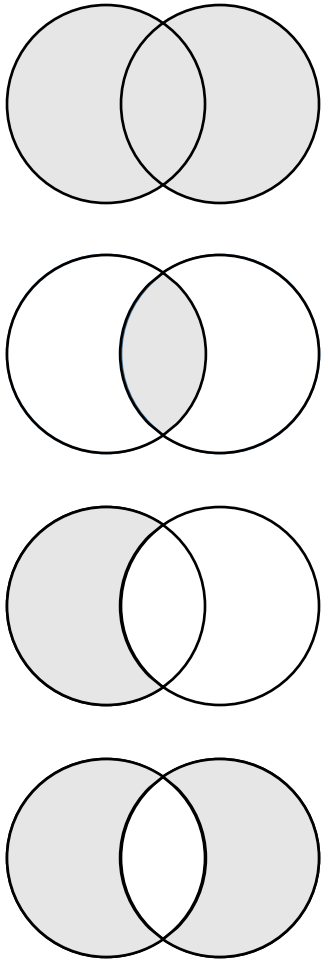
- The empty set is a subset of all sets.

  $\emptyset \subset A$

Basic Set Theory: Operations

Given two sets A and B, we can perform the following operations:

- `A ∪ B` : union
- `A ∩ B` : intersection
- `A \ B` : difference
- `A Δ B` : symmetric difference

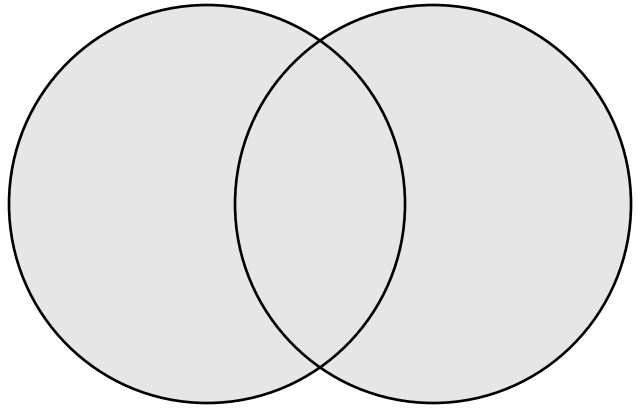


Basic Set Theory: Operations

Given two sets A and B, we can perform the following operations:

- `A ∪ B` : union

The *union* of A and B, includes all the elements of both A and B.

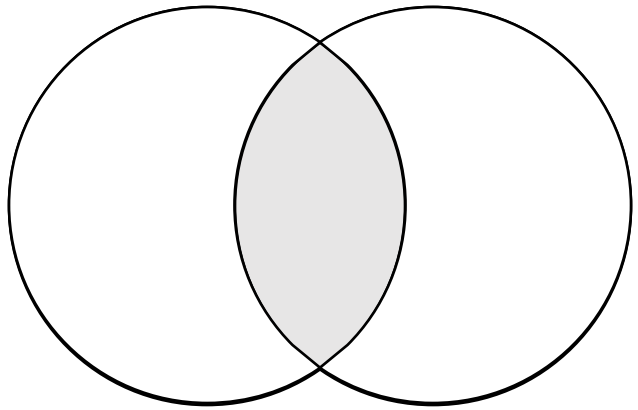


Basic Set Theory: Operations

Given two sets A and B, we can perform the following operations:

- `A ∩ B` : intersection

The *intersection* of A and B, includes all the elements common to A and B.

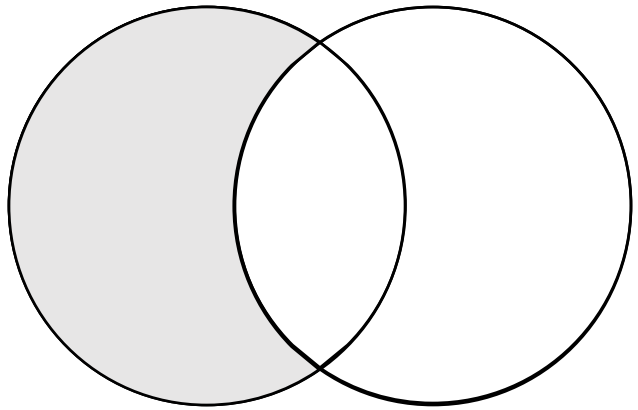


Basic Set Theory: Operations

Given two sets A and B, we can perform the following operations:

- `A \ B` : difference

The *difference* of A and B, includes all elements of A that are not elements of B

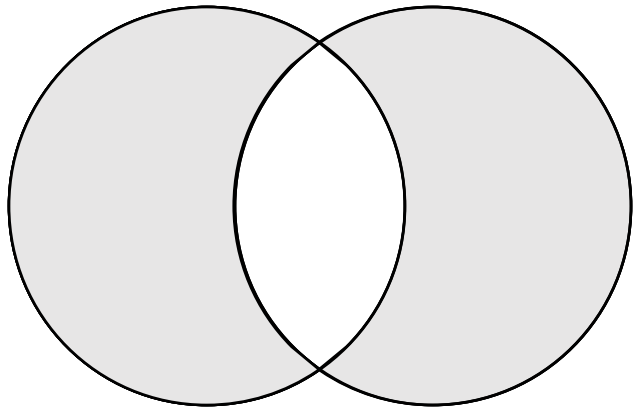


Basic Set Theory: Operations

Given two sets A and B, we can perform the following operations:

- `A Δ B` : symmetric difference

The *symmetric difference* of A and B, includes all elements of A that are not elements of B and all elements of B that are not in A.



Basic Set Theory: Summary

Notations:

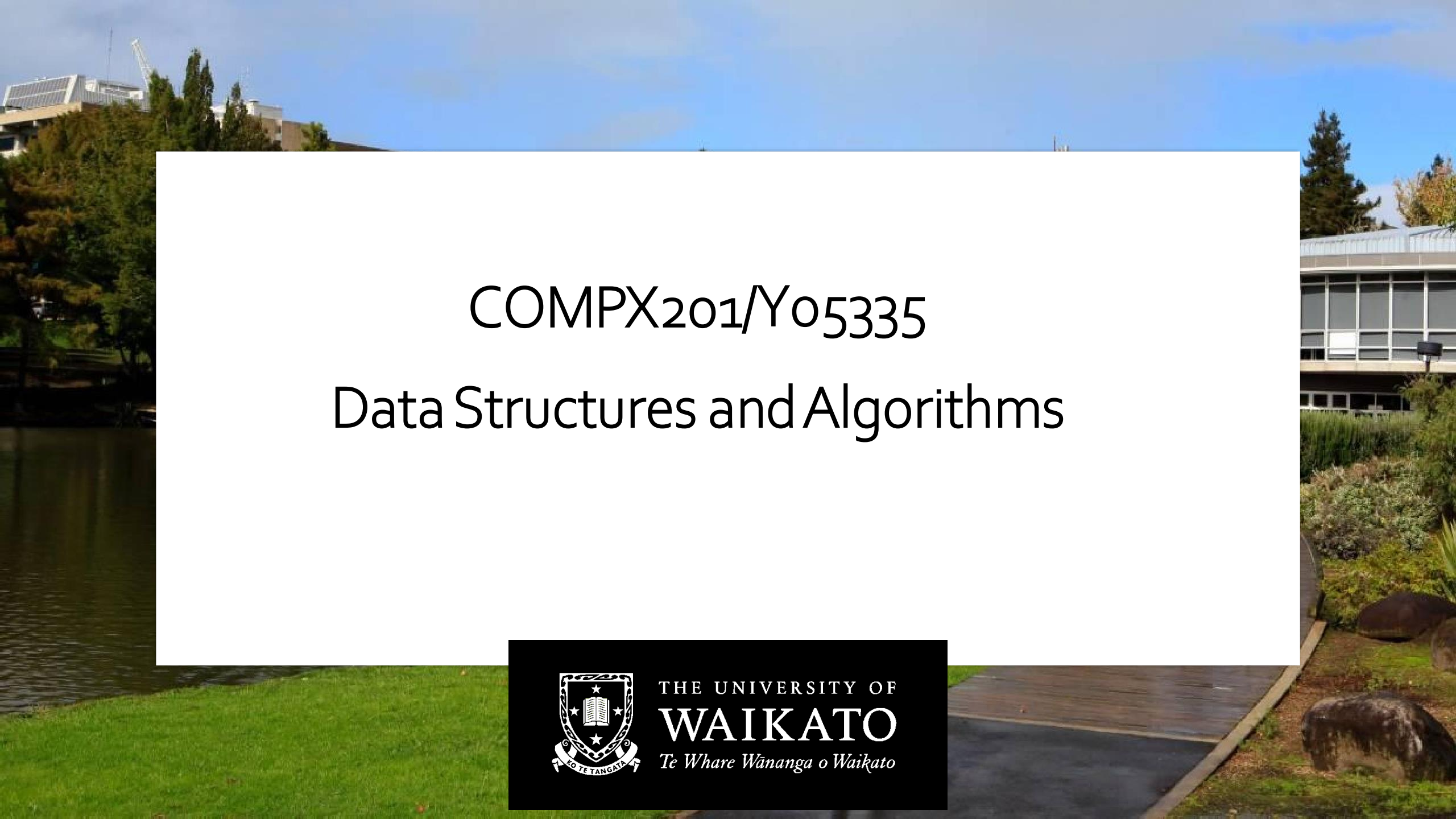
- $\in$: is element, X is an element of Set A
- $\notin$: is not element, X is not an element of Set A

- $\{\} = \emptyset$: empty set

- $=$: equals, Set A is equal to Set B
- $\neq$: not equal, Set A is not equal to Set B
- $\subset$: subset, set A is a subset of Set B

Operations:

- $A \cup B$: union
- $A \cap B$: intersection
- $A \setminus B$: difference
- $A \Delta B$: symmetric difference



COMPX201/Y05335

Data Structures and Algorithms

THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

Sets in Java
Part 1

COMPX201/Y05335

Overview

- Implementing a Set in Java
  - IsEmpty
  - hasElement
  - Add
  - Remove

Basic Set Theory: Summary

Notations:

- $\in$: is element, X is an element of Set A
- $\notin$: is not element, X is not an element of Set A

- $\{\} = \emptyset$: empty set

- $=$: equals, Set A is equal to Set B
- $\neq$: not equal, Set A is not equal to Set B
- $\subset$: subset, set A is a subset of Set B

Operations:

- $A \cup B$: union
- $A \cap B$: intersection
- $A \setminus B$: difference
- $A \Delta B$: symmetric difference

Implementing a Set in Java

We are going to create a custom data structure called “IntSet”

- Create data structure
- Print
- isEmpty
- hasElement
- Add
- Remove
- isSubset
- isEqual
- Union
- Intersection
- Difference
- Symmetric difference

Implementing a Set in Java

Creation and printing

Creating our custom datatype

- Use an array for our Set
- Give the set an initial size
- Keep track of the next index for adding elements

Print

- Create a ‘print’ method for testing our operations
- `“Arrays.toString(array)”`
- `import java.util.Arrays;`

Assumption: Valid numbers are positive integers greater than zero

Implementing a Set in Java

Okay, let's try it ...

Implementing a Set in Java

Checking if Set is empty

- Method: isEmpty()
- Loop through all elements
- If you find a non-empty element (i.e. x > 0), then Set is not empty

Implementing a Set in Java

Checking if element exists in the Set

- Method: hasElement()
- Loop through all elements
- If you find the element in the Set, then element exists

Implementing a Set in Java

Adding an element:

First time:

- Add number to first element in set
- Increment ‘index’

Next time:

- Add number to ‘index’ element in set
- Increment ‘index’

What if the set is full?

- Increase size of set
- Add number to ‘index’ element
- Increment ‘index’

What if the element already exists?

- Check if element exists
- If it does, don’t add it

Implementing a Set in Java

Adding an element:

~~First time:~~

- ~~Add number to first element in set~~
- ~~Increment 'index'~~

Adding an element:

- Add number to 'index' element in set
- Increment 'index'

What if the set is full?

- Increase size of set
- Add number to 'index' element
- Increment 'index'

What if the element already exists?

- Check if element exists
- If it does, don't add it

Implementing a Set in Java

Removing an element:

Pre-check

- Check if number is valid (BUT > 0)
- Check if number is in Set (i.e. hasElement)

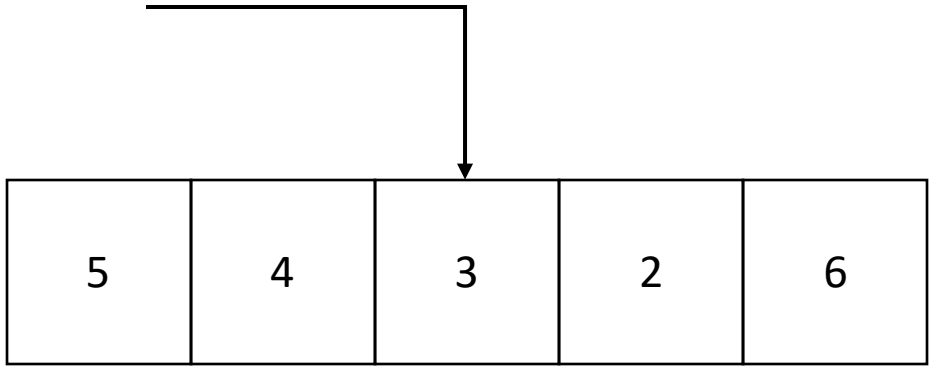
Remove element:

- Loop through list until you find element
- Replace it with zero

BUT, we need to clean up the empty element in the Set …

Implementing a Set in Java

Removing an element:

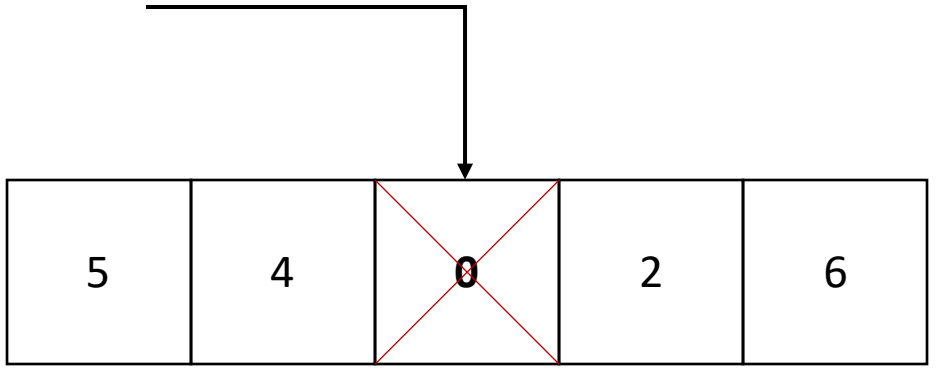


numSet.remove(3)

(1) Loop through list until you find element

Implementing a Set in Java

Removing an element:

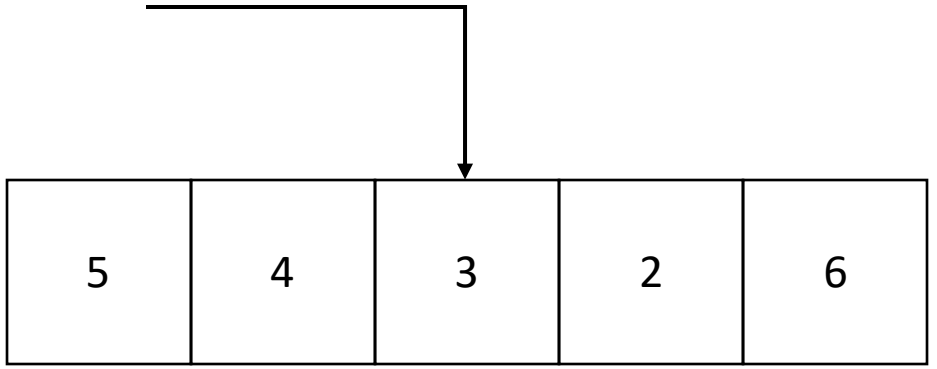


numSet.remove(3)

(1) Loop through list until you find element
(2) Replace it with zero

Implementing a Set in Java

Removing an element:

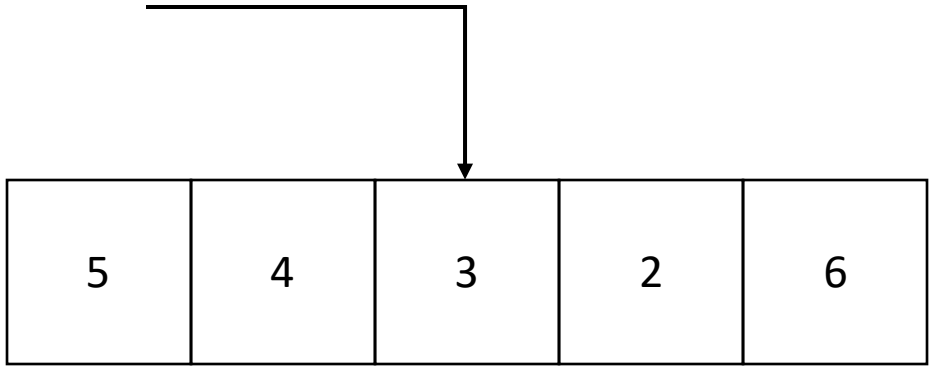


numSet.remove(3)

(1) Loop through list until you find element

Implementing a Set in Java

Removing an element:

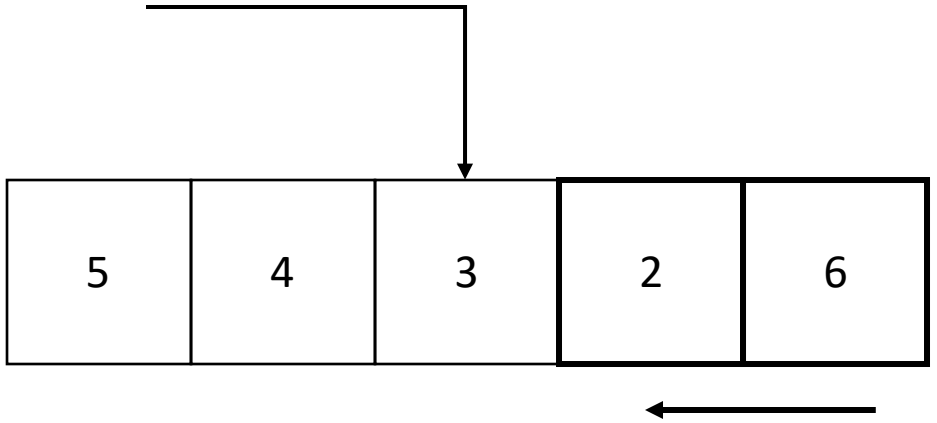


numSet.remove(3)

(1) Loop through list until you find element
(2) Keep track of it's index (index = 2)

Implementing a Set in Java

Removing an element:

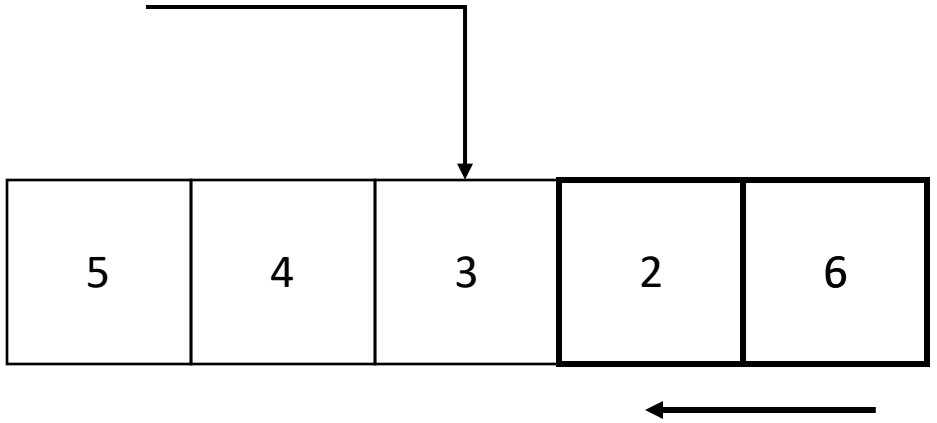


numSet.remove(3)

(1) Loop through list until you find element
(2) Keep track of it's index (index = 2)
(3) Copy the elements across from the right (i.e. index+1 onwards)

Implementing a Set in Java

Removing an element:

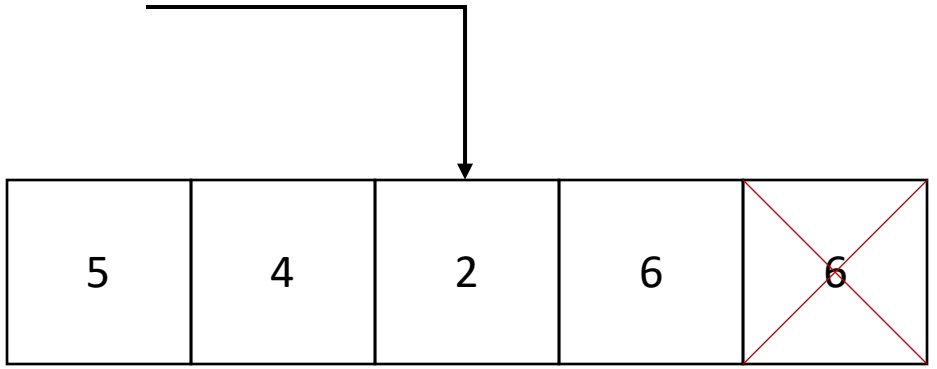


numSet.remove(3)

(1) Loop through list until you find element
(2) Keep track of it's index (index = 2)
(3) Copy the elements across from the right (i.e. index+1 onwards)

Implementing a Set in Java

Removing an element:



numSet.remove(3)

(1) Loop through list until you find element
(2) Keep track of its index (index = 2)
(3) Copy the elements from the right (i.e., index+1 onwards)
(4) Remove the last item

Implementing a Set in Java

Removing an element:

Pre-check

- Check if number is valid (its > 0)
- Check if number is in Set (i.e. hasElement)

Remove element:

- Loop through list until you find element
- Keep track of it's index
- Copy the elements across from the right
- Remove last item in Set

Implementing a Set in Java – IntSet.java

```
import java.util.Arrays;

public class IntSet{
    int size;
    int intArray[];

    public IntSet(int a){
        size = a;
        intArray = new int[size];
    }

    public void displayArray(){
        System.out.println("intSetA: " + Arrays.toString(intArray));
    }

    public boolean isEmpty(){
        for(int i = 0; i < intArray.length; i++){
            if(intArray[i] > 0){
                return false;
            }
        }
        return true;
    }
```

Implementing a Set in Java – IntSet.java

```
    public boolean hasItem(int t){
        for(int i = 0; i < intArray.length; i++){
            if(intArray[i] == t){
                return true;
            }
        }
        return false;
    }

    public void addItem(int a){
        if(isEmpty()){
            intArray[0] = a;
            return;
        }
        if(hasItem(a)){
            return;
        }
        for(int i = 0; i < intArray.length; i++){
            if(intArray[i] == 0){
                intArray[i] = a;
                return;
            }
        }
    }
    public void removeItem(int n){
        if(isEmpty()){
            return;
        }
        if(!hasItem(n)){
            return;
        }
        for(int i = 0; i < intArray.length; i++){
            if(intArray[i] == n){
                for(int j = i; j + 1 < intArray.length; j++){
                    intArray[j] = intArray[j+1];
                }
                intArray[intArray.length - 1] = 0;
                return;
            }
        }
    }
}
```

Implementing a Set in Java – Main.java

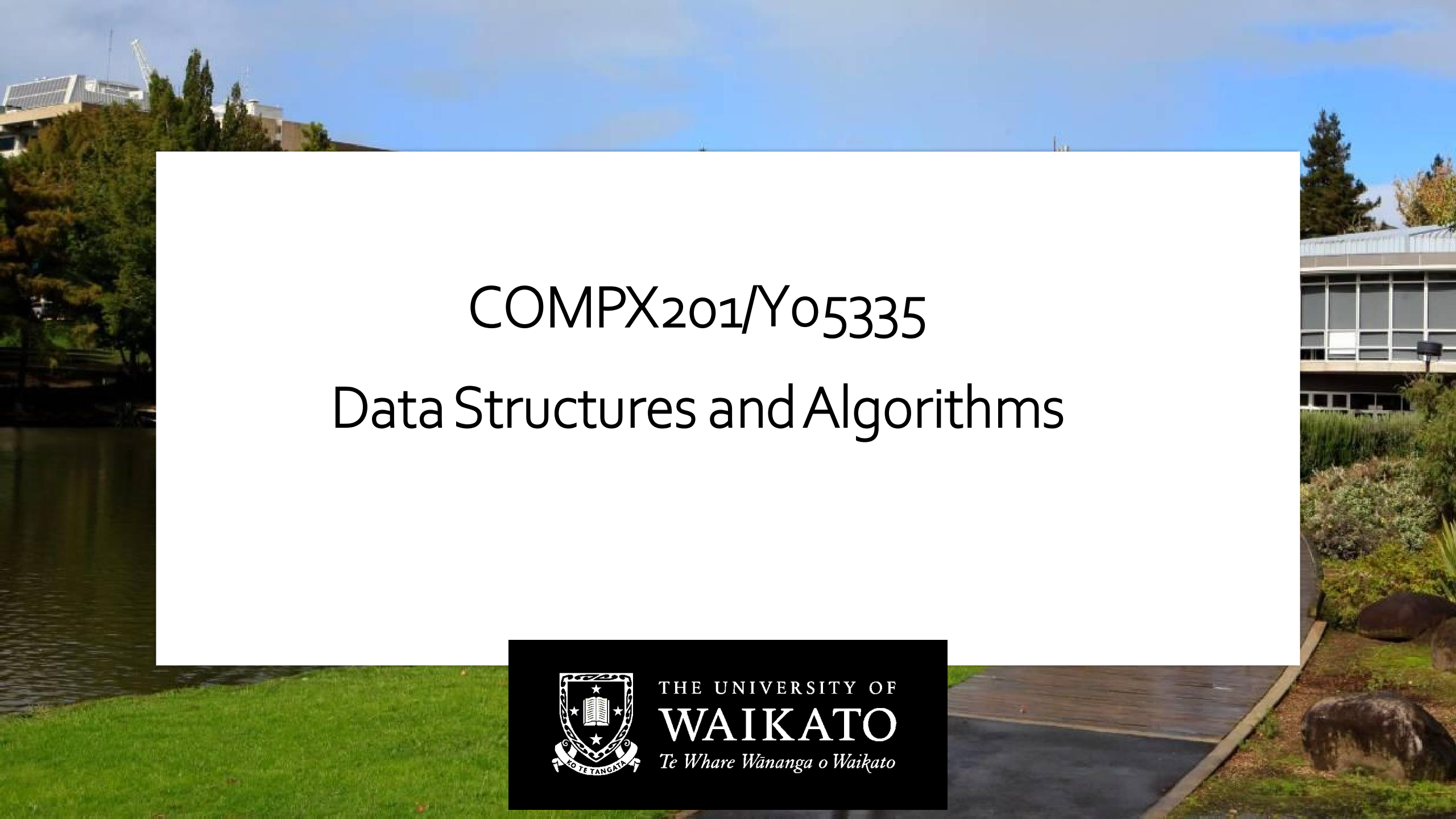
```
public class Main{
    public static void main(String [] args){
        IntSet intSetA = new IntSet(5);

        System.out.println("Printing an empty set...");
        intSetA.displayArray();
        System.out.println("Is the set empty?: " + intSetA.isEmpty());
        System.out.println("Does the set contain the integer 3?: " + intSetA.hasItem(3));

        intSetA.addItem(1);
        intSetA.addItem(2);
        intSetA.addItem(3);
        intSetA.addItem(4);
        intSetA.addItem(5);

        System.out.println();

        System.out.println("Printing a non-empty set...");
        intSetA.displayArray();
        System.out.println("Is the set empty?: " + intSetA.isEmpty());
        System.out.println("Does the set contain the integer 3?: " + intSetA.hasItem(3));
        intSetA.removeItem(2);
        System.out.print("Inspecting the set after removing 2: ");
        intSetA.displayArray();
    }
}
```



COMPX201/Y05335

Data Structures and Algorithms

THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

Sets in Java
Part 2

COMPX201/Y05335

Last time: Sets in Java Part 1

- Implementing a Set in Java
  - IsEmpty
  - hasElement
  - Add
  - Remove

Overview: Sets in Java Part 2

- Implementing a Set in Java
  - isSubset
  - isEqual
  - Union
  - Intersection
  - Difference
  - Symmetric difference

Basic Set Theory: Summary

Give it a try!

Notations:

- $=$: equals, Set A is equal to Set B
- $\neq$: not equal, Set A is not equal to Set B
- $\subset$: subset, set A is a subset of Set B

Operations:

- `A ∪ B` : union
- `A ∩ B` : intersection
- `A \ B` : difference
- `A Δ B` : symmetric difference

Implementing a Set in Java

Is Set A a subset of Set B? $A \subset B$

Give it a try!

Yes!

- If A is an empty Set
- Or if every element of A is an element of B

```
A = {}          B = anything
A = {1,2,3}     B = {1,2,3,4}
```

Implementing a Set in Java

Is Set A equal to Set B? `A=B`

Give it a try!

Yes!

- If both sets are empty `A = {}` `B = {}`
- Or if A and B are identical `A = {1,2,3}` `B = {1,2,3}`
- Or if A is a subset of B, and B is a subset of A `A = {1,2,3}` `B = {2,1,3}`

Implementing a Set in Java

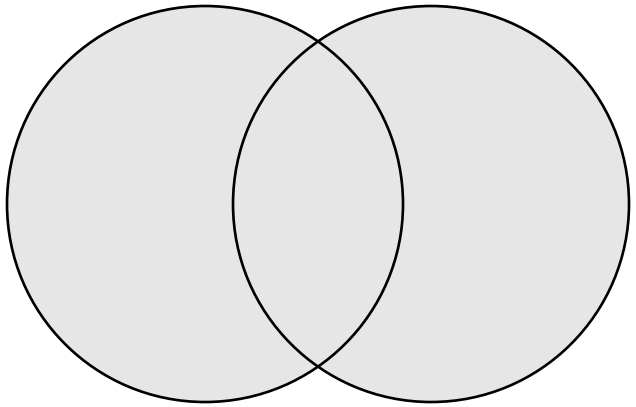
Union A ∪ B

Give it a try!

The *union* of A and B, includes all the elements of both A and B.

A = {1,2,3} B = {2,3,4}

A ∪ B = {1,2,3,4}



Implementing a Set in Java

Union A ∪ B

- If both sets are empty:

  Return an empty Set

- Or if A is empty:

  Return B

- Or if B is empty:

  Return A

- Otherwise, return the union of both

Give it a try!

```
A = {}   B = {}
A ∪ B = {}
```

```
A = {}          B = {2,3,4}
A ∪ B = {2,3,4}
```

```
A = {1,2,3}     B = {}
A ∪ B = {1,2,3}
```

```
A = {1,2,3}     B = {2,3,4}
A ∪ B = {1,2,3,4}
```

Implementing a Set in Java

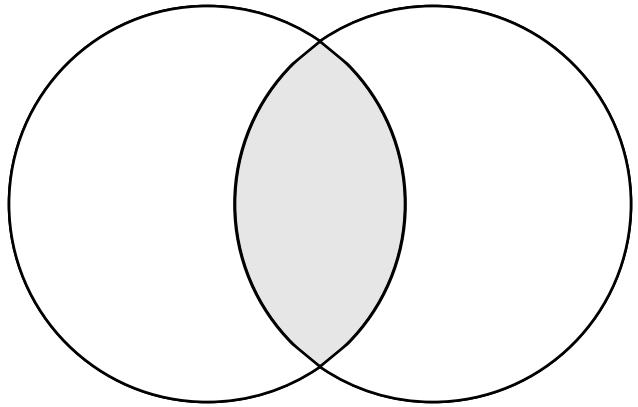
Intersection A ∩ B

The *intersection* of A and B, includes all the elements common to A and B.

```
A = {1,2,3}    B = {2,3,4}

A ∩ B = {2,3}
```

Give it a try!



Implementing a Set in Java

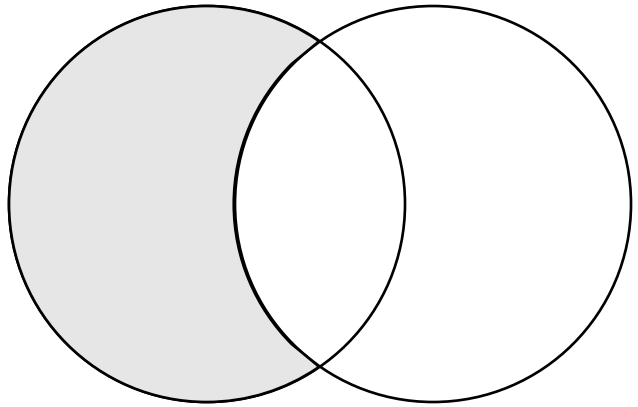
Difference `A \ B`

The *difference* of A and B, includes all elements of A that are not elements of B

`A = {1,2,3}` `B = {2,3,4}`

`A \ B = {1}`

Give it a try!



Implementing a Set in Java

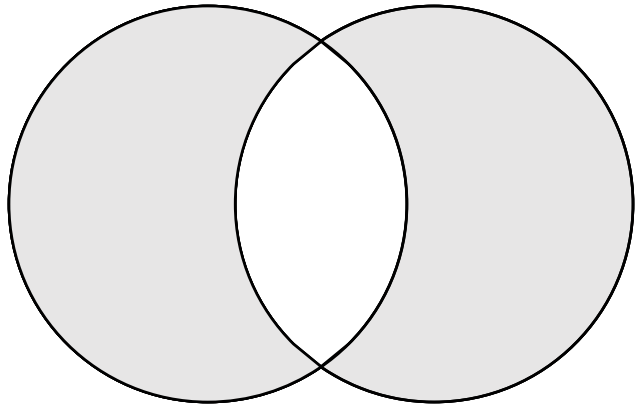
Symmetric difference $A \cup B$

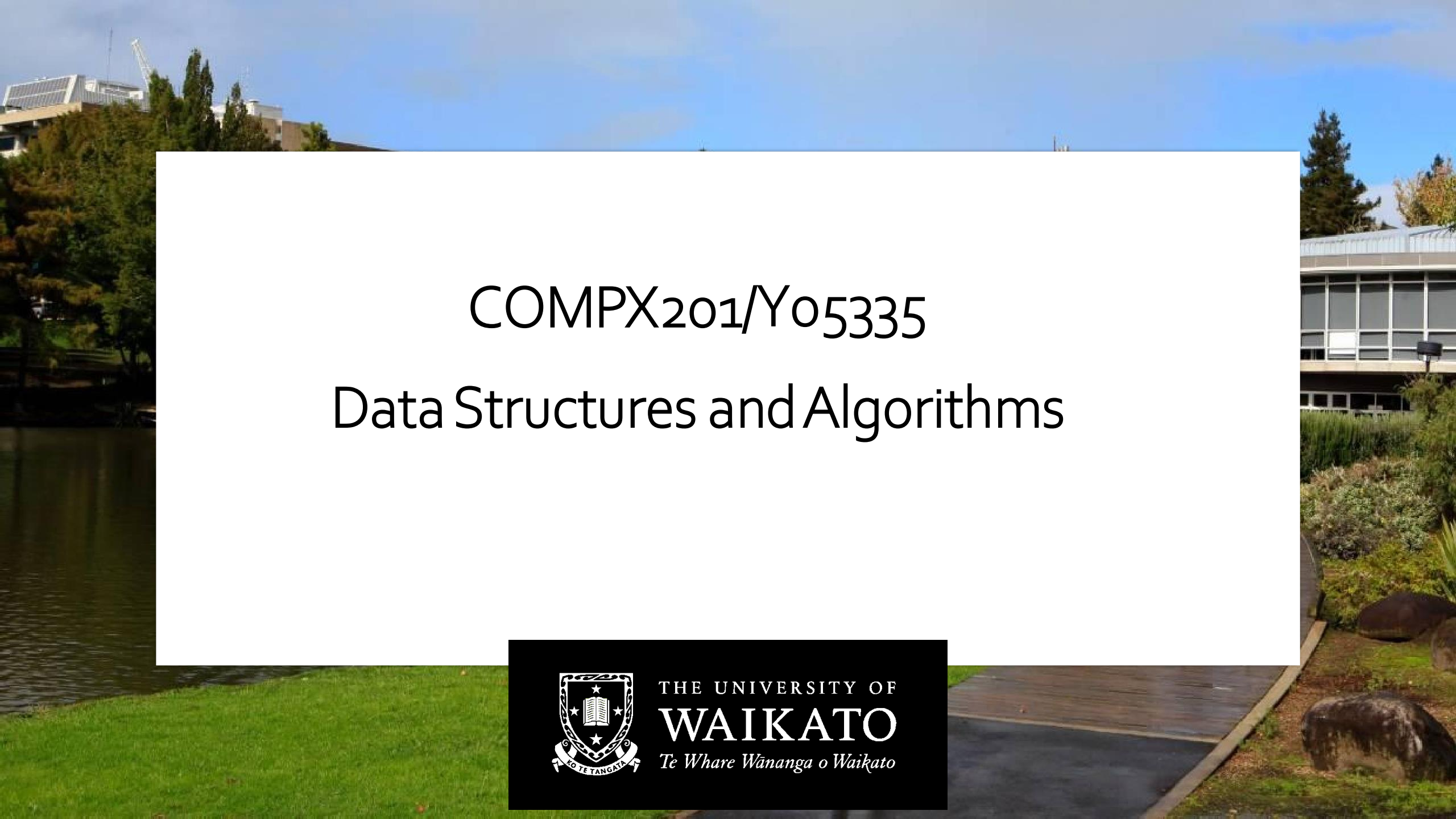
The *symmetric difference* of A and B includes all elements of A that are not elements of B, and all elements of B that are not in A.

$A = \{1,2,3\}$ $B = \{2,3,4\}$

$A \cup B = \{1,4\}$

Give it a try!





COMPX201/Y05335

Data Structures and Algorithms

THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato