JAVA ARRAY LISTS AND LINKED LISTS

DR. KRISHNENDU GUHA

TEACHING ASSISTANTS

ZARRAR HAIDER (123120583@UMAIL.UCC.IE)

RHEENA BLAS (120347046@UMAIL.UCC.IE)

ARRAY LIST

- This comes into context when we need to consider a **resizable array**

The ArrayList **class** is a resizable array, which can be found in the java.util package.

What is the difference between a built-in array and an Array List in Java?
The **size of an array cannot be modified** (if you want to add or remove elements to/from an array, you have to create a new one).
While **elements can be added and removed from an ArrayList** whenever you want.

Obviously, the **syntax is also slightly different**

Syntax of ArrayList:

```
import java.util.ArrayList; // import the ArrayList class
ArrayList<String> cars = new ArrayList<String>(); // Create an ArrayList object called cars that will store strings
```

For details: https://www.w3schools.com/java/java_arraylist.asp

ADDING ITEMS TO AN ARRAYLIST

To add elements to the ArrayList, use the add() method:

Example: Create a list of four cars

```
import java.util.ArrayList;
public class Main
{
        public static void main(String[] args)
        {
                ArrayList<String> cars = new ArrayList<String>();
                cars.add("Volvo");
                cars.add("BMW");
                cars.add("Ford");
                cars.add("Mazda");
                System.out.println(cars);
        }
}
```

```
[Volvo, BMW, Ford, Mazda]
```

NOTE: **Elements in an ArrayList are actually objects**.
In the examples above, we created elements (objects) of type "String".

ACCESS AN ITEM

To access an element in the ArrayList, use the **get() method** and refer to the **index number**:

Example Syntax

cars.get(0);

Example: Output the name of the first car in the list

```
import java.util.ArrayList;
public class Main
{
        public static void main(String[] args)
        {
                ArrayList<String> cars = new ArrayList<String>();
                cars.add("Volvo");
                cars.add("BMW");
                cars.add("Ford");
                cars.add("Mazda");
                System.out.println(cars.get(0));
        }
}
```

```
Volvo
```

NOTE: Array indexes start with 0: [0] is the first element. [1] is the second element, etc.

MODIFY/ CHANGE AN ITEM

To modify an element, use the **set() method** and refer to the **index number:**

Example Syntax

```
cars.set(0, "Opel");
```

Example: To change the name of the first car in the list from Volvo to Opel

```
import java.util.ArrayList;
public class Main
{
        public static void main(String[] args)
        {
                ArrayList<String> cars = new ArrayList<String>();
                cars.add("Volvo");
                cars.add("BMW");
                cars.add("Ford");
                cars.add("Mazda");
                cars.set(0, "Opel");
                System.out.println(cars.get(0));
        }
}
```

```
Opel
```

REMOVE AN ARRAY ELEMENT

To remove an element, use the **remove() method** and refer to the **index number**:

Example Syntax

```
cars.remove(0);
```

Example: To remove the first element in the cars

```
import java.util.ArrayList;
public class Main
{
        public static void main(String[] args)
        {
                ArrayList<String> cars = new ArrayList<String>();
                cars.add("Volvo");
                cars.add("BMW");
                cars.add("Ford");
                cars.add("Mazda");
                cars.remove(0);
                System.out.println(cars.get(0));
        }
}
```

```
BMW
```

ArrayList Size

To find out how many elements an ArrayList have, use the **size method**:

Example
cars.size();

```
import java.util.ArrayList;
public class Main
{
        public static void main(String[] args)
        {
                ArrayList<String> cars = new ArrayList<String>();
                cars.add("Volvo");
                cars.add("BMW");
                cars.add("Ford");
                cars.add("Mazda");
                System.out.println(cars.size());
        }
}
```

```
4
```

REMOVE ALL ELEMENTS OF ARRAY LIST

To remove all the elements in the ArrayList, use the **clear() method**:

Example Syntax

```
cars.clear();
```

```
import java.util.ArrayList;
public class Main
{
        public static void main(String[] args)
        {
                ArrayList<String> cars = new ArrayList<String>();
                cars.add("Volvo");
                cars.add("BMW");
                cars.add("Ford");
                cars.add("Mazda");
                System.out.println(cars.size());
                cars.clear();
                System.out.println(cars.size());
        }
}
```

```
4
0
```

Loop Through an ArrayList

We can Loop through the elements of an ArrayList with a for loop, and use the size() method to specify how many times the loop should run:

```
import java.util.ArrayList;
public class Main
{
        public static void main(String[] args)
        {
                ArrayList<String> cars = new ArrayList<String>();
                cars.add("Volvo");
                cars.add("BMW");
                cars.add("Ford");
                cars.add("Mazda");

                for (int i = 0; i < cars.size(); i++)
                {
                        System.out.println(cars.get(i));
                }
        }
}
```

```
Volvo
BMW
Ford
Mazda
```

looping through an ArrayList with the **for-each** loop:

```
import java.util.ArrayList;

public class Main
{
        public static void main(String[] args)
        {
                ArrayList<String> cars = new ArrayList<String>();
                cars.add("Volvo");
                cars.add("BMW");
                cars.add("Ford");
                cars.add("Mazda");

                for (String i : cars)
                {
                        System.out.println(i);
                }
        }
}
```

```
Volvo
BMW
Ford
Mazda
```

Till now, we have **used Strings**

A **String in Java is an object (not a primitive type)**
To use other types, such as int, we use Integer.
For other primitive types, use: Boolean for boolean, Character for char, Double for double, etc:

Example
Create an ArrayList to store numbers (add elements of type Integer):

```
import java.util.ArrayList;
public class Main
{
        public static void main(String[] args)
        {
                ArrayList<Integer> myNumbers = new ArrayList<Integer>();
                myNumbers.add(1);
                myNumbers.add(2);
                myNumbers.add(3);
                myNumbers.add(4);

                for (int i : myNumbers)
                {
                        System.out.println(i);
                }
        }
}
```

```
1
2
3
4
```

SORTING AN ARRAY LIST

Another useful class in the java.util package is the **Collections class**, which include the **sort() method** for sorting lists alphabetically or numerically:

Sorting an arraylist of strings

```
import java.util.ArrayList;
import java.util.Collections; // Import the Collections class

public class Main
{
        public static void main(String[] args)
        {
                ArrayList<String> cars = new ArrayList<String>();
                cars.add("Volvo");
                cars.add("BMW");
                cars.add("Ford");
                cars.add("Mazda");

                Collections.sort(cars); // Sort cars
                for (String i : cars)
                {
                        System.out.println(i);
                }
        }
}
```

```
BMW
Ford
Mazda
Volvo
```

Sorting an **ArrayList of Integers**

```
import java.util.ArrayList;
import java.util.Collections; // Import the Collections class
public class Main
{
	public static void main(String[] args)
	{
		ArrayList<Integer> myNumbers = new ArrayList<Integer>();
		myNumbers.add(33);
		myNumbers.add(15);
		myNumbers.add(20);
		myNumbers.add(34);
		myNumbers.add(8);
		myNumbers.add(12);

		for (int i : myNumbers)
		{
			System.out.println(i);
		}


		Collections.sort(myNumbers); // Sort myNumbers

		for (int i : myNumbers)
		{
			System.out.println(i);
		}
	}
}
```

```
33
15
20
34
8
12
8
12
15
20
33
34
```

JAVA LINKED LISTS

- The LinkedList class is almost identical to the ArrayList

The LinkedList class is a **collection which can contain many objects of the same type**, just like the ArrayList.
The **LinkedList class has all of the same methods as the ArrayList class because they both implement the List interface**.

This means that you can **add items, change items, remove items and clear the list in the same way.**

However, while the ArrayList class and the LinkedList class can be used in the same way, **they are built very differently**.

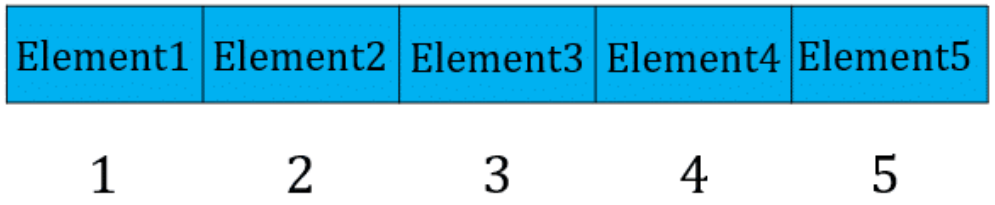
The ArrayList class **has a regular array inside it.**
When an element is added, it is **placed into the array**.
If the **array is not big enough, a new, larger array is created to replace the old one** and the old one is removed.

The LinkedList **stores its items in "containers**."
The **list has a link to the first container and each container has a link to the next container in the list**.
To **add an element to the list, the element is placed into a new container and that container is linked to one of the other containers** in the list.

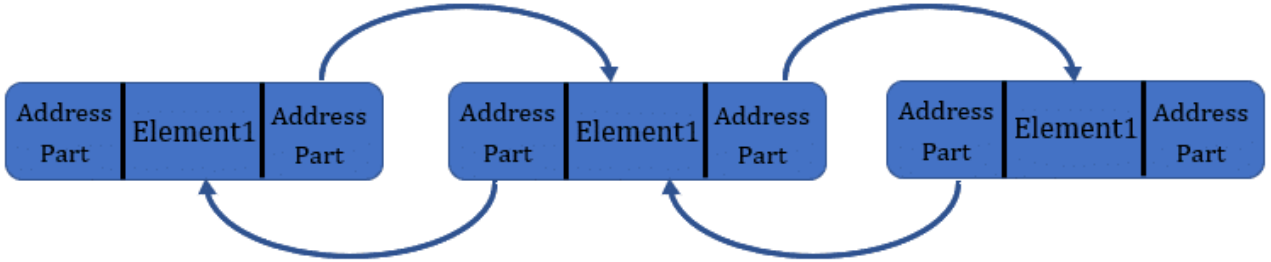
When To Use
Use an ArrayList for **storing and accessing data**, and LinkedList to **manipulate data**.

For details: https://www.w3schools.com/java/java_linkedlist.asp

ArrayList



LinkedList



IMPORTING THE LINKED LIST CLASS

```java
// Import the LinkedList class
import java.util.LinkedList;
public class Main
{
        public static void main(String[] args)
        {
                LinkedList<String> cars = new LinkedList<String>();
                cars.add("Volvo");
                cars.add("BMW");
                cars.add("Ford");
                cars.add("Mazda");
                System.out.println(cars);
        }
}
```

```
[Volvo, BMW, Ford, Mazda]
```

LINKED LIST METHODS

Method	**Description**
addFirst()	Adds an item to the beginning of the list.
addLast()	Add an item to the end of the list
removeFirst()	Remove an item from the beginning of the list.
removeLast()	Remove an item from the end of the list
getFirst()	Get the item at the beginning of the list
getLast()	Get the item at the end of the list

ADDING AN ITEM TO THE BEGINNING OF THE LIST

```
import java.util.LinkedList;
public class Main {
  public static void main(String[] args) {
    LinkedList<String> cars = new LinkedList<String>();
    cars.add("Volvo");
    cars.add("BMW");
    cars.add("Ford");
    System.out.println(cars);
    System.out.println();

    // Use addFirst() to add the item to the beginning
    cars.addFirst("Mazda");
    System.out.println(cars);
  }
}
```

```
[Volvo, BMW, Ford]

[Mazda, Volvo, BMW, Ford]
```

ADDING AN ITEM TO THE LAST OF THE LIST

```
import java.util.LinkedList;

public class Main {
  public static void main(String[] args) {
    LinkedList<String> cars = new LinkedList<String>();
    cars.add("Volvo");
    cars.add("BMW");
    cars.add("Ford");
    System.out.println(cars);
    System.out.println();
    // Use addLast() to add the item to the end
    cars.addLast("Mazda");
    System.out.println(cars);
  }
}
```

```
[Volvo, BMW, Ford]

[Volvo, BMW, Ford, Mazda]
```

REMOVING THE FIRST ITEM FROM A LIST

```
import java.util.LinkedList;


public class Main {
  public static void main(String[] args) {
    LinkedList<String> cars = new LinkedList<String>();
    cars.add("Volvo");
    cars.add("BMW");
    cars.add("Ford");
    cars.add("Mazda");
            System.out.println(cars);
        System.out.println();
    // Use removeFirst() remove the first item from the list
    cars.removeFirst();
    System.out.println(cars);
  }
}
```

```
[Volvo, BMW, Ford, Mazda]

[BMW, Ford, Mazda]
```

REMOVING THE LAST ITEM FROM THE LIST

```
import java.util.LinkedList;
public class Main {
  public static void main(String[] args) {
    LinkedList<String> cars = new LinkedList<String>();
    cars.add("Volvo");
    cars.add("BMW");
    cars.add("Ford");
    cars.add("Mazda");
          System.out.println(cars);
          System.out.println();
    // Use removeLast() remove the last item from the list
    cars.removeLast();
    System.out.println(cars);
  }
}
```

```
[Volvo, BMW, Ford, Mazda]

[Volvo, BMW, Ford]
```

OUTPUT THE FIRST ITEM OF A LIST

```
import java.util.LinkedList;

public class Main {
  public static void main(String[] args) {
    LinkedList<String> cars = new LinkedList<String>();
    cars.add("Volvo");
    cars.add("BMW");
    cars.add("Ford");
    cars.add("Mazda");

    System.out.println(cars);
    System.out.println();
    // Use getFirst() to display the first item in the list
    System.out.println(cars.getFirst());
  }
}
```

```
[Volvo, BMW, Ford, Mazda]

Volvo
```

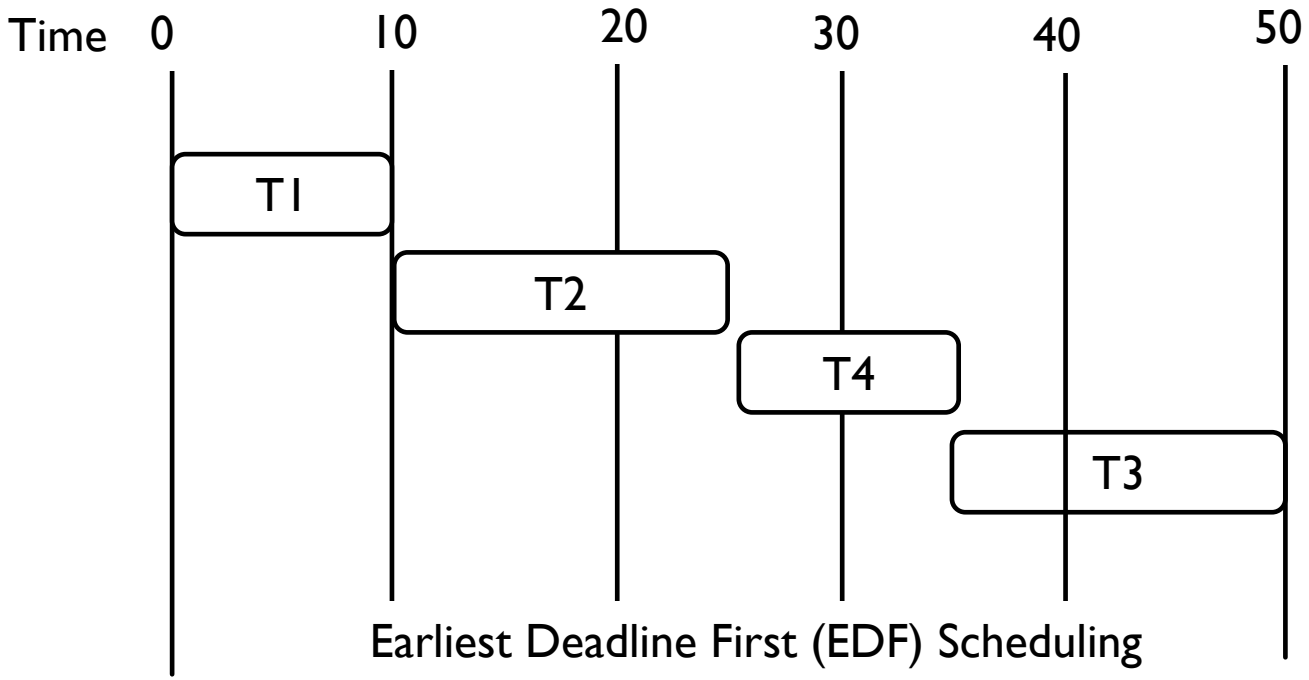
OUTPUT THE LAST ITEM OF A LIST

```
import java.util.LinkedList;

public class Main {
  public static void main(String[] args) {
    LinkedList<String> cars = new LinkedList<String>();
    cars.add("Volvo");
    cars.add("BMW");
    cars.add("Ford");
    cars.add("Mazda");
    System.out.println(cars);
    System.out.println();
    // Use getLast() to display the last item in the list
    System.out.println(cars.getLast());
  }
}
```

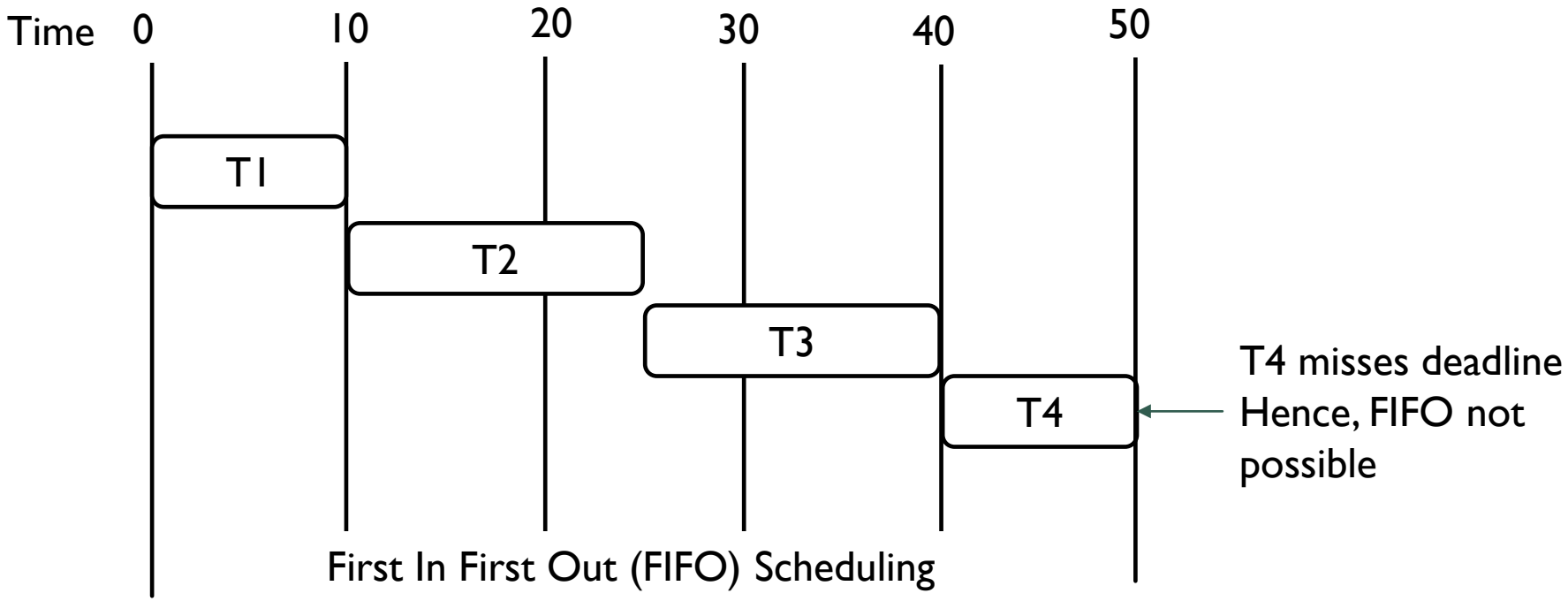
```
[Volvo, BMW, Ford, Mazda]

Mazda
```


Any Questions


Thank You!



Earliest Deadline First (EDF) Scheduling



First In First Out (FIFO) Scheduling