Java List ADT
Learning Objectives

- To be able to import the Java list package (the package contains the ArrayList and LinkedList classes)
- To be able to declare and create a List
- To be able to add, remove, or retrieve elements in a List
- To be able to check how many elements are stored inside a List
- To be able to replace an element in a List
- To be able to know when to use an ArrayList and when to use a LinkedList
- To be able to traverse a List using loops
- To be able to solve problems using List objects
Java Package

- A **package** is a collection of related classes
- Java provides several packages that we can use in our program
- Before using a package, it must be imported with the `import` keyword
- Import statements must be placed at the top (beginning) of the class
- The `ArrayList` and `LinkedList` classes are located inside the `java.util` package
- Before using a List, you must add the following statement at the top of your class:

```java
import java.util.*;
```
Java List Classes

List

ADT / Interface

ArrayList

LinkedList

Implementation classes
Modeling with List

- List objects are used to **store several elements of the same type**
- List objects are a type of data structure
- List objects model a list in real life: list of students, list of songs (playlist), etc.
- A List has a **variable length (size)**: it grows or shrinks automatically

- Each element in a List has a **position**:
  - The first element is at position 0
  - The last element is at the position (size of the arraylist) - 1
Declaring and Creating ArrayList Objects

- ArrayLists are objects
- To declare an ArrayList, you write:

  ```java
  List<TypeOfElements> arrayListName = new ArrayList<TypeOfElements>();
  ArrayList<TypeOfElements> arrayListName = new ArrayList<TypeOfElements>();
  ```

Example:

```java
ArrayList<Integer> myList = new ArrayList<Integer>();
```  
Will create an ArrayList of integers

```java
ArrayList<Student> studentsList = new ArrayList<Student>();
```  
Will create an ArrayList of Student
Declaring and Creating LinkedList Objects

- LinkedLists are objects
- To declare a LinkedList, you write
  
  ```java
  List<TypeOfElements> ListName = new LinkedList<TypeOfElements>();
  or
  LinkedList<TypeOfElements> linkedListName = new LinkedList<TypeOfElements>();
  ```

Example:

```java
LinkedList<Integer> myList = new LinkedList<Integer>();
```
Will create a LinkedList of integers

```java
LinkedList<Student> studentsList = new LinkedList<Student>();
```
Will create a LinkedList of Student
List Methods: size()

- The `size` method returns the number of elements inside an ArrayList.
- An empty ArrayList has a size of 0.

Example:

```java
ArrayList<Student> studentsList = new ArrayList<Student>();
or
LinkedList<Student> studentsList = new LinkedList<Student>());

System.out.println(studentsList.size()); ➜ 0
```
List Methods: `add(element)`

- The `add(element)` method adds the parameter element to the end of the list.
- Adding an element to the list will increase the size of the list by 1.
**List Methods: add(element)**

Example:

```java
ArrayList<String> studentsList = new ArrayList<String>();
System.out.println(studentsList.size()); // 0
studentsList.add("Serena"); ["Serena"]
System.out.println(studentsList.size()); // 1
studentsList.add("Travis"); ["Serena", "Travis"]
System.out.println(studentsList.size()); // 2
```

The elements must be of type **String**.
List Methods: `add(element)`

Example:
```java
LinkedList<String> myList = new LinkedList<String>();
myList.add("Serena");  // Serena is at position 0
myList.add("Travis");  // Travis is at position 1
myList.add("Tyrone");  // Tyrone is at position 2
```
List Methods: `add(element)`

Example:
```java
ArrayList<Student> studentsList = new ArrayList<Student>();

studentsList.add(new Student("Serena", "sophomore"));
// ["Serena","sophomore"]

studentsList.add(new Student("Travis", "freshman"));
// ["Serena","sophomore"], ["Travis", "freshman"]
```
List Methods: `add(index, element)`

- The `add(index, element)` method inserts the parameter element at the position `index`.
- The method first moves (to the right) all the element after position `index` (included) to make room.
List Methods: `add(index, element)`

Example:
```java
LinkedList<String> studentsList = new LinkedList<String>();
studentsList.add("Serena"); // ["Serena"]
studentsList.add(0, "Travis"); // ["Travis", "Serena"]
studentsList.add(1, "Eva"); // ["Travis", "Eva", "Serena"]
```
List Methods: `remove(index)`

- The `remove(index)` method removes the element at position `index` from the List.
- The method moves (to the left) all the elements after position `index` to fill up the gap.
- The method decreases the size of the list by 1.
List Methods: \texttt{remove(index)}

Example:
\begin{verbatim}
ArrayList<String> studentsList = new ArrayList<String>();

studentsList.add("Serena");
studentsList.add("Travis");
studentsList.add("Eva");

studentsList.remove(0);  \[\text{"Travis", "Eva"}\]

studentsList.remove(1); \[\text{"Travis"}\]
\end{verbatim}
List Methods: remove(index)

Example:
```java
LinkedList<String> studentsList = new LinkedList<String>();
studentsList.add("Serena");
studentsList.add("Travis");
studentsList.add("Eva");

studentsList.remove(1); \[\{\text{"Serena"}, \text{"Eva"}\}\]
```

Removes Travis. Eva is now at position 1

```java
studentsList.remove(0); \[\{\text{"Eva"}\}\]
```

Removes Serena. Eva is now at position 0
List Methods: get(index)

- The **get(index)** returns the element at position `index` in the ArrayList.
- The method can be called after adding elements to the ArrayList.
- `index` is a number between 0 and the **size of the List -1**.
List Methods: `get(index)`

Example:
```java
ArrayList<String> studentsList = new ArrayList<String>();

studentsList.add("Serena");
studentsList.add("Travis");
studentsList.add("Eva");

studentsList.get(0); ➔ Serena

studentsList.get(1); ➔ Travis

studentsList.get(2); ➔ Eva
```
List Methods: `set(index, element)`

- The `set(index, element)` replaces the value at position index with element
- The method can be called after adding elements to the List
- `index` is a number between 0 and the size of the List - 1
List Methods: `set(index, element)`

Example:
```java
ArrayList<String> studentsList = new ArrayList<String>();

studentsList.add("Serena");
studentsList.add("Travis");
studentsList.add("Eva");

studentsList.set(0, "Williams");
studentsList.set(1, "Scott");
```

Output:
```
[“Serena”, “Travis”, “Eva”]
[“Williams”, “Travis”, “Eva”]
[“Williams”, “Scott”, “Eva”]
```
ArrayList vs LinkedList

● Use an ArrayList when:
  ○ You know how many items will be stored
  ○ The items won’t change in order
  ○ The items won’t change in number

● Use an LinkedList when:
  ○ You don’t know how many items will be stored
  ○ You want to add many items when the program is running
  ○ You want to remove many items when the program is running
## Array vs List

<table>
<thead>
<tr>
<th>Operation</th>
<th>Array</th>
<th>ArrayList / LinkedList</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length/size</td>
<td>array.length;</td>
<td>aList.size();</td>
</tr>
<tr>
<td>Access element given the index</td>
<td>value = array[index];</td>
<td>value = aList.get(index);</td>
</tr>
<tr>
<td>Modify element given index</td>
<td>array[index] = value;</td>
<td>aList.set(index, value);</td>
</tr>
</tbody>
</table>
List iteration with the Enhanced For-loop

Example:
```java
ArrayList<String> studentsList = new ArrayList<String>();
studentsList.add("Serena");
studentsList.add("Travis");
studentsList.add("Eva");

for(String name : studentsList){
    System.out.println(name);
}
```

Will print:
"Serena"
"Travis"
"Eva"
List iteration with the For-loop

- Same as with arrays

Example:

```java
LinkedList<String> studentsList = new LinkedList<String>();
studentsList.add(“Serena”);
studentsList.add(“Travis”);
studentsList.add(“Eva”);

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

Will print:
“Serena”
“Travis”
“Eva”
List iteration with the `while`-loop

- Same as with arrays

Example:
```java
ArrayList<String> studentsList = new ArrayList<String>();
studentsList.add("Serena");
studentsList.add("Travis");
studentsList.add("Eva");

index = 0;
while(index < studentsList.size()){
    System.out.println(studentsList.get(index));
    index++;
}
```

Will print:
- "Serena"
- "Travis"
- "Eva"