

# Linked Lists

---

A byte size lesson in Java programming.

# Before we begin make sure you know...

Keyword	Definition
Abstraction	The process of simplifying reality as much as possible.
Class	The template to encapsulate data and behaviour of an object.
Contract	A promise to behave in a certain way.
Inheritance	The 'is a' relationship between two classes.
Instance	A concrete object of a particular class.
Interface	The 'behaves like' relationship between two classes. This comes with a 'promise'.
Object	A complex data type that has unique identity, state and behaviour.

The keywords in this table will be underlined throughout the slides.



# Linked List is a Collection

---

- Java provides a **Collection** class which is meant to define how a **dynamic set** of objects should behave.
- Any class that is a **Collection** through inheritance means that the purpose of the class is to efficiently **maintain a dynamic set** of similar objects.
- The **Collection** class in Java is an interface so it is an abstraction of a set and we cannot create an instance of **Collection** directly but we create instances of classes that implement the contract.
- We can create an instance of a **LinkedList** which is a type of **Collection** to efficiently maintain a dynamic set of objects **in a certain way**.

# How does a Collection behave?

---

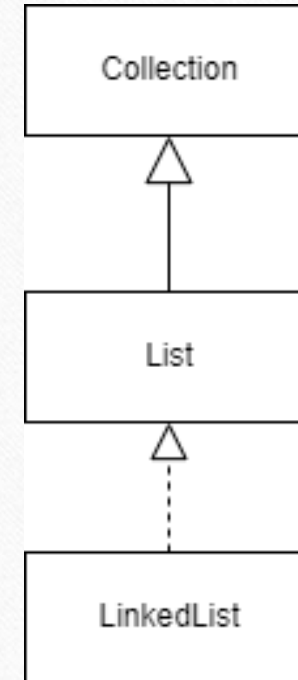
- An object that promises to behave like a **Collection** must implement the following methods:
- `boolean add(Object obj);` // this is for adding a node
- `Iterator<Object> iterator();` // this is to be able to traverse nodes one by one
- `boolean isEmpty();` // whether the collection has nodes at all

**Link 4.2.2** Standard operations of a collection – When writing pseudocode and it involves a **collection** the above operations can be assumed. Since the above methods are specific to Java, IB suggests more explicit methods: `addItem`, `getNext`, `hasNext` and `isEmpty`.

## LinkedList 'acts like' List which 'is a' Collection

- Java provides behavioural contracts for various types of **Collection**.
- An example is the **List**.
- Therefore to be more precise:

**LinkedList** implements **List**  
extends **Collection** { } \*

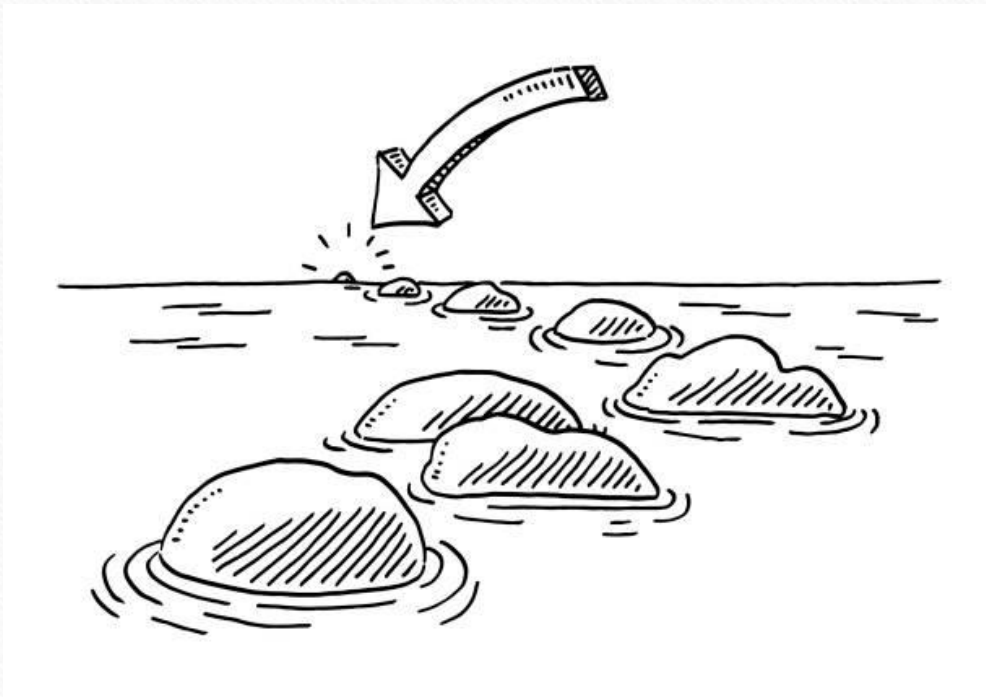


```
LinkedList<Integer> numbersList = new LinkedList<Integer>();
```

\* Unfortunately the actual structure in Java is more complicated than depicted here. The above is a stripped down version.



# What does it take to be a List?



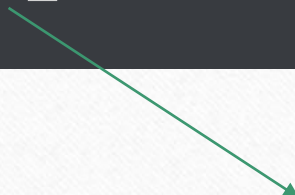
- A **sequence** of nodes that is always traversed from the first one, known as **head**.
- Each node contains the **data object** and a **reference to the next one in sequence**.
- The **last node** in the sequence has data and a **null pointer reference**.
- Any particular node can be retrieved, added and removed.
- Java enforces other rules for lists to make it more useable.

# Create a LinkedList

---

- Here is how we can create a **LinkedList** in Java:

```
LinkedList<insert_type_here> linkedList = new LinkedList<>();
```



We don't have to commit to the size of an entire list, but we have to give a clear indication of size for a **single node**.

# Let us test your understanding

```
LinkedList<> names = new LinkedList<>();
```

```
LinkedList<> numbers = new LinkedList<>();
```

```
LinkedList<> prices = new LinkedList<>();
```

```
LinkedList<> players = new LinkedList<>();
```

```
LinkedList<> students = new LinkedList<>();
```



# Standard operations on a LinkedList

---

```
LinkedList<String> animals = new LinkedList<>();

// do we have a farm?
boolean isFarm = animals.isEmpty();

// adding nodes
animals.add("cow");
animals.add("chicken");
animals.add("pig");

// how about now?
isFarm = animals.isEmpty();
```

# Standard operations on a LinkedList

```
if (isFarm) {  
    // get this object to be able to traverse nodes  
    Iterator<String> iterator = animals.iterator();  
  
    // typical looping through all nodes  
    while(iterator.hasNext()) {  
        System.out.println("Old Mc Donald had a farm ee-ay-ee-ay-oh");  
        System.out.println("And on this farm he had a ");  
  
        System.out.println(animals.next());  
  
        System.out.println("Ee-ay-ee-ay-oh");  
    }  
}
```