# COMP249 Tutorial 9: Linked Data Structures and Collections

Ella Noyes

August 7, 2024

## 1 Linked Data Structures

A linked data structure contains a set of nodes, each containing data and one or more references to other nodes in the structure.

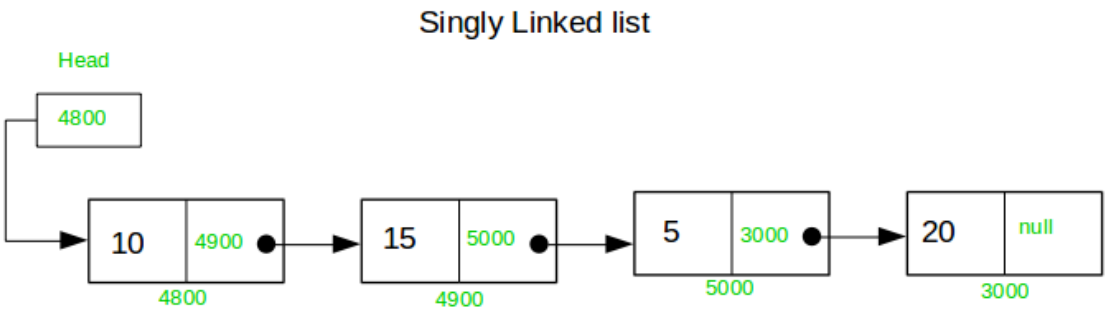


Figure 1: Figure is from: geeksforgeeks

Here, the "head" node contains a reference to the node at address 4800. That next node contains some data (10), and a reference to the next node at address 4900. That next node contains data (15), and a reference to the next node at 5000, etc. This is a singly linked-list: each node contains only one reference, which points to the next node in the list.

A doubly-linked list contains two references: one to the next node in the list, and another to the previous node in the list.
Circular linked-lists are another type of linked list, a

Linked-lists can be used like arrays, but have their own advantages:

- Dynamic size: unlike an array, which uses contiguous blocks of memory and is therefore fixed in size, adding and removing nodes to a linked-list is straightforward
- Nodes can be efficiently inserted or deleted
- Efficient memory use

## 2 Java Collections

"The collections framework is a unified architecture for representing and manipulating collections, enabling them to be manipulated independently of the details of their representation" - from Oracle's Java Documentation.

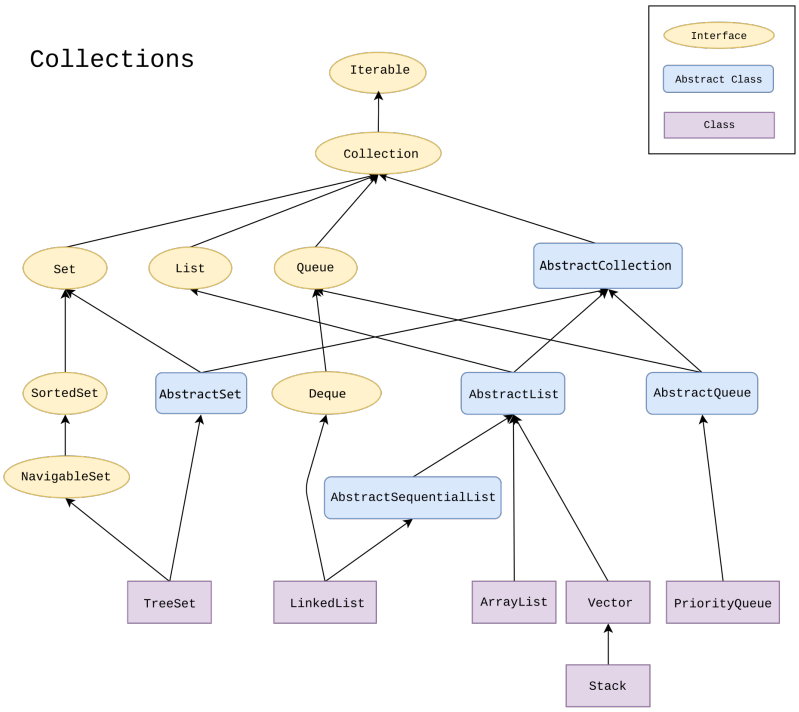


Figure 2: Figure is from: https://en.wikipedia.org/wiki/Java_collections_framework

Java provides a robust framework for handling collections of objects through the Java Collections Framework. This framework includes various interfaces and classes that help in storing and manipulating groups of objects. The primary interfaces are Collection, List, Set, and Queue, and the framework is designed to be generic, meaning you can work with any type of object.

Collections are used to store, retrieve, manipulate, and communicate aggregate data. The Java Collections Framework provides:

- Lists: Ordered collections that allow duplicates. Example: ArrayList, LinkedList.
- Sets: Collections that do not allow duplicates. Example: HashSet, TreeSet.
- Queues: Collections designed for holding elements prior to processing. Example: LinkedList (also implements Queue), PriorityQueue.
- Maps: Collections that store key-value pairs. Example: HashMap, TreeMap.

As shown in the hierarchy, all of these Interfaces implement Collection, which implements Iterable. Implementing iterable means that you can iterate through your collection using a for-each loop (like you might do with an array!).

## 2.1 Working with Lists

Lists are ordered collections that allow duplicate elements. This interface supports inserting an element anywhere in the list.

We'll look at a simple system that manages grocery orders using a LinkedList to implement a queue. But first, let's look at the LinkedList class:

LinkedList is a generic Java class with methods for performing various List operations.
Now let's look at the grocery order manager code example.

## 2.2 Working with Queues

Queues are collections that hold elements for processing; they sometimes follow the First-In-First-Out (FIFO) principle. They are ideal for managing tasks or processes that need to be handled in a specific order.
In this next example, we'll use Java's LinkedList class to handle a First-In-First-Out (FIFO) queue of tasks.

# Programming Problem: Music Queue

You are tasked with creating a system to manage a music queue. This queue will store a sequence of music tracks (songs) for playback in a specific order. The system should support the following functionalities:

1. **Add a Song:** Add a single song to the end of the play queue.
2. **Play a Song Now:** Play a song now and clear rest of the queue.
3. **Print the Queue:** Print the current state of the music queue using an iterator.

## Classes

- **Song Class:** Represents a song with a title and artist.
- **MusicQueue Class:** Manages a queue of songs using a linked list.

## Details

- The `Song` class should have the following attributes:
  - `title` (String): The title of the song.
  - `artist` (String): The artist of the song.
- The `MusicQueue` class should provide the following methods:
  - `playNext(Song song)`: Adds a single song to the end of the queue.
  - `playSong()`: Removes and plays the song at the front of the queue
  - `playNow()`: Plays a song now and clears the rest of the queue
  - `printQueue()`: Prints the list of songs currently in the queue.

Test your program's functionality. Use your printQueue() method between operations to verify that the program's logic performs correctly.

**Optional:** If you have the time to extend this problem, then create another class, Album, containing a LinkedList of songs. Implement methods in the MusicQueue class to add an entire album to the queue, or to play an entire album now (i.e., clear the existing queue and add all of the songs of the album to the empty queue).