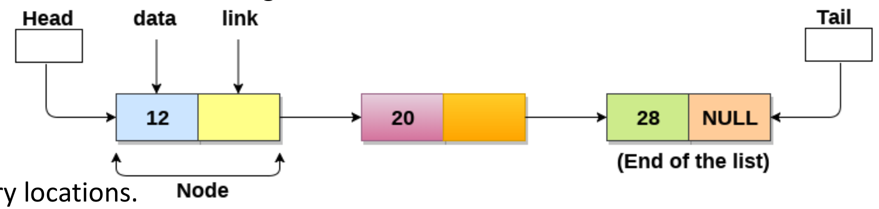
Linked List

1. Motivation

ArrayList is useful when adding or removing items to/from the **end of the list**.
Adding or removing items at the beginning or the middle of ArrayList takes long time, because the underlying array must be recreated in the memory in order to be **contiguous**. The alternative is to create a data structure that stores objects non-contiguously in memory and establishes links among them.

2. Linked List Structure

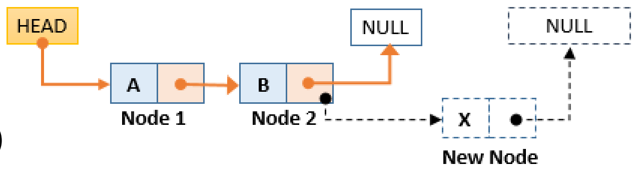


The elements of a linked list are called **nodes**.

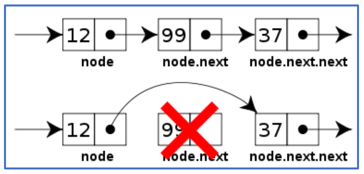
- The structure is linear.
- Nodes are stored in **non-contiguous** memory locations.
- Nodes are linked – each node refers to the next one.
- Nodes are comprised of two items – data and link/reference to the next node (**pointer**).
- The **entry point** into a liked list is called the **head** of the list. A pointer to the last node is called a **tail**.

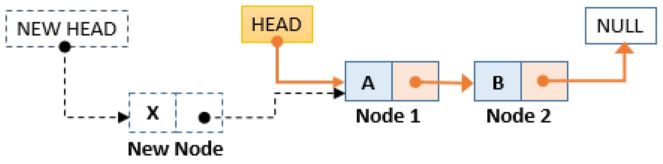
Notes: The last node has a reference to `null`.
The head is **not a node**, but reference to the first node! It contains only the address of the first node.
If the list is empty the head is a `null` reference.
Empty node **cannot** be present in a linked list.
The linked list size is limited only to the size of the memory. It is not defined at the time of declaration.

3. Main Operations on Linked List



- Add a node to the end (**not very useful!**)
- Add a node to the beginning
- Insert a node before/after another node (or at specific index)
- Remove a node by item value or by index:
- Get an item by index
- Find an index of an item
- Get the size of the list
- Clear the list





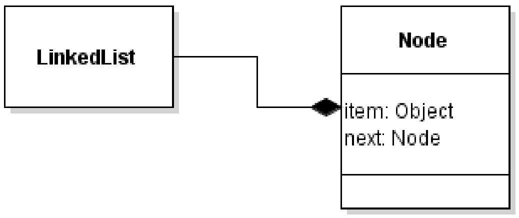
The most common operation on Linked List is traversing – visiting each node of the list once and using the link/reference to find the next node.

4. LinkedList in Java

Java has a `LinkedList` class in `java.util` package, as part of the Java Collections Framework.
The methods you've learned for ArrayList are **also applicable** to LinkedList – they both implement the `List` interface. In fact, `List` reference type is commonly used for both `ArrayList` and `LinkedList` instances:

```
List<Integer> arrayList = new ArrayList<Integer>();
List<Integer> linkedList = new LinkedList<Integer>();
```

5. Custom LinkedList Implementations



A custom LinkedList implementation consists of (at least) two classes:
Writing a custom implementation helps in better understanding how LinkedList works, where its efficiency is, and when and how to use it.
It is also a stepping stone for writing a custom data structure.