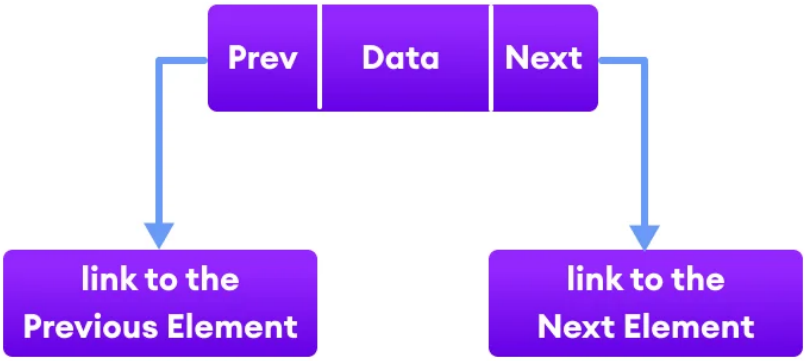
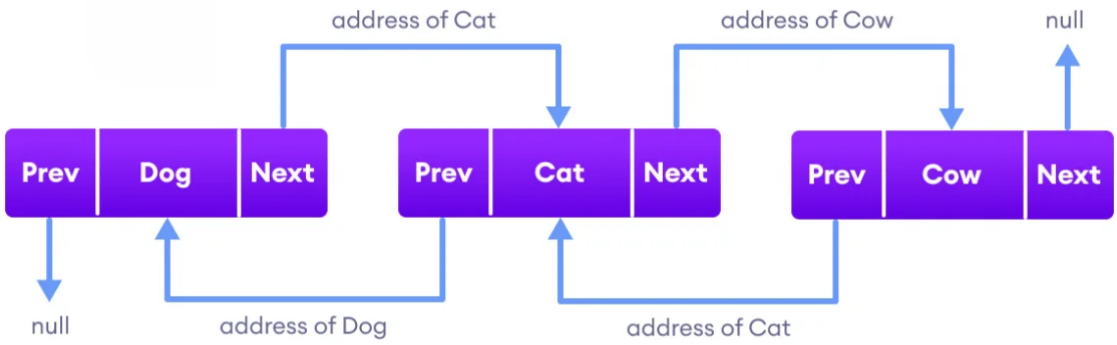
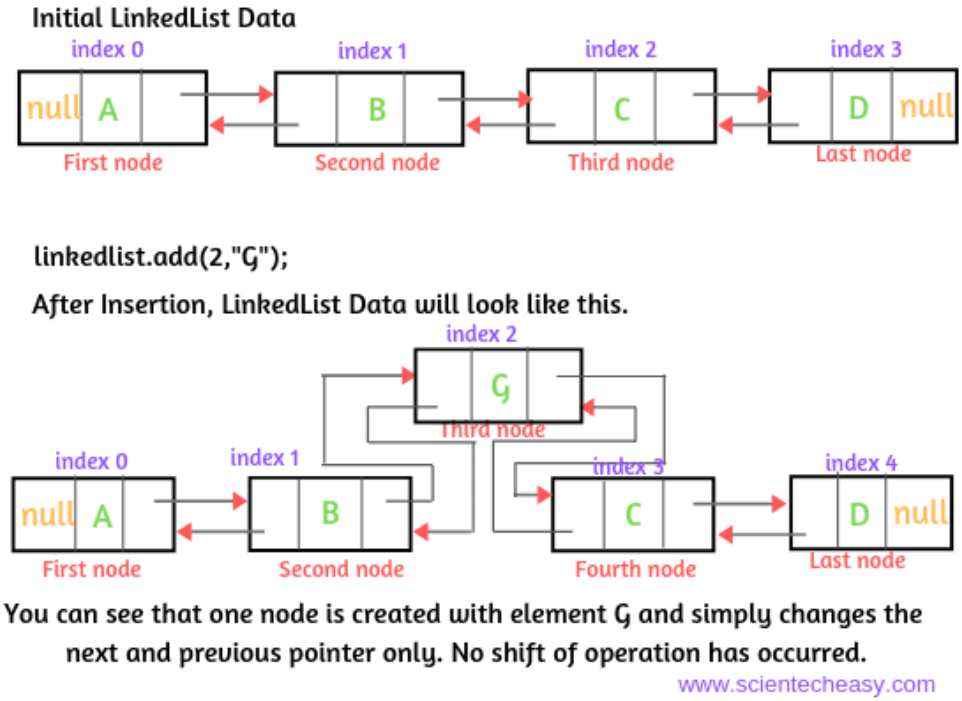
LinkedList
We can add element or delete any element in between in array without shifting

Best choice to add or remove element from middle
Worst choice for retrieving as retrieval operation start from first element
Underline data structure is linked list
Insertion order preserved
Duplicate element allowed
Heterogenous element allowed
Null insertion is possible
No random access interface
No default or initial capacity



Ajay Atul Manavi Priya ---→ Party





Program:

```
package CollectiionProg;

import java.util.LinkedList;

public class LinkedListProg {

                public static void main(String[] args) {

                 LinkedList a = new LinkedList();
                 LinkedList b = new LinkedList(a);

                 System.out.println(a.size());

                 a.add("Jon");
                 a.add("Sersi");
                 a.add("LittleFinger");
                 a.add("Tyrion");
                 a.add('+');
                 a.add(1000);
                 a.add(null);
                 System.out.println(a);
                 a.add("Jon");
                 a.add("Sersi");
                 a.add("LittleFinger");
                 a.add("Tyrion");
                 a.addFirst("Sansa");
                 a.addLast("Arya");
                 System.out.println(a);
```

```
        a.add(1, "Jaime");
        System.out.println(a);

        a.poll();
        System.out.println("Poll Operation = " + a);
        a.pollFirst();
        System.out.println(a);
        a.pollLast();
        System.out.println(a);
        a.remove();
        a.removeFirst();
        a.removeLast();
        System.out.println(a);
        a.removeFirstOccurrence("LittleFinger");
        System.out.println(a);

    }

}
```

Array List	**Linked List**
Best choice for frequent access	Best choice for insertion and deletion
Worst choice for insertion and deletion	Worst choice for frequent access
Underlying data structure is resizable and growable	Doubly linked list
Implement random access	Not

Set (I)

- It is sub interface of collection interface
- Does not allow duplicate values
- Insertion order not preserved
- Allow only 1 null value insertion
- Does not deal with array index operation
- It deal with hash table operation

1) Hash set

- It is implementation of set interface
- Only 1 null insertion is allowed
- Do not allow duplicate values
- Order of insertion => random
- Heterogenous
- **Storage type => hash table (key, values)**
- Data structure => hash table
- Searching easy so hashSet is best choice
- Default initial capacity 16
- Fill ration (load factor) 0.75
- It grows double when condition arrived (from 16 to 32)

We used hash table to remove duplicate values and if order of insertion is not maintained
111%10 = 1 2344%10 = 4
Hashset

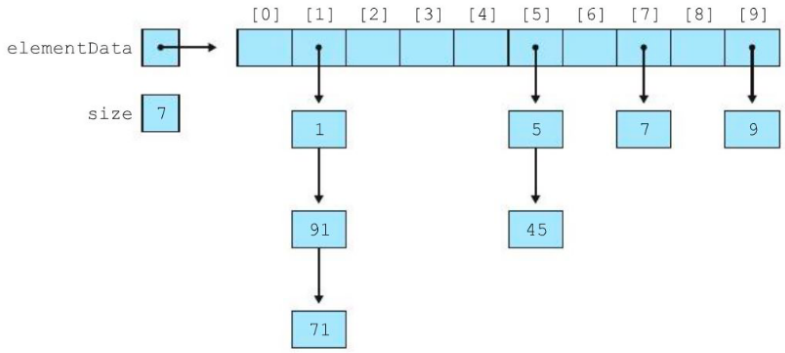


Figure 18.6 Hash collisions resolved by separate chaining