| Abstract<br>**HashNode** | Node |
| --- | --- |
| • Set Key Attribute<br>• Crafted to be utilized in a Hashed List<br>• Set next pointer | • Node |

| Abstract<br>**Node** | HashNode |
| --- | --- |
| • Set data attribute<br>• Set next pointer<br>• Crafted to be utilized in a Linked List | |

Abstract

# HashList

List

- Set head pointer
- Insert by key and position
- Determine the size
- Remove by Position
- Remove All Nodes
- Get Data by Position
- Get Key by Position
- Get Position of Item
- Link Series of Nodes

- List
- Efficiency
- HashNode

Abstract

## List

HashList

- Manipulate and Set: head and count
- Insert Data Anywhere
- Remove Data Anywhere
- Get Data by Position
- Set Data at any Location
- Overloaded <<,=, [] operators
- Link Series of Nodes
- Prevent Access to Nonexistent Memory

- Node
- Efficiency

## Input

- Perform data type conversions
- Prevent invalid arguments and invalid memory access
- Returning Appropriate data variables

Abstract

# HashTable

- Manipulate Dynamic Array of Hashed Lists
- Get Number of Nodes in the Table
- Generate a Hash Function Dependent on User key/value
- Rehash when Table becomes Sensitive to Multiple Collisions
- Implement Separate Chaining Collision Resolution Mechanism
- Retrieve Table Size
- Add by key/value
- Remove by key/value
- Calculate Load Factor
- Determine Size
- Retrieve all keys/data

- Node
- List
- HashNode
- HashList
- Efficiency
- Item