**Data Structures and ADTs**

- *Abstract Data Types (ADTs)* are user defined data types. An ADT has 2 parts:
  1. A name or type, specifying a set of data (e.g. **Dictionary**).
  2. Descriptions of all the operations (or methods) that do things with that type (e.g. **find**, **insert**, **remove**)

  The descriptions indicate *what* the operations do, not *how* they do it.

- A *data structure* is a systematic way of organizing and accessing data from a computer (e.g. array, linked list).
  Data structures are used to implement ADTs.
**ADT Dictionary**

- **find (key):** returns a record with the given key, or null if no record has the given key
- **insert(key, data):** inserts a new record with given key and data
  - **ERROR** if the dictionary already contains a record with the given key
- **remove(key):** removes the record with the given key
  - **ERROR** if there is no record with the given key
Java Interface for ADT Dictionary

public interface Dictionary {

    public Object find(Object key);

    public void insert(Object key, Object data)
        throws DuplicatedKeyException;

    public void remove(Object key)
        throws NoKeyException;

}
A Java Implementation for ADT Dictionary

```java
public class LinkedListDictionary implements Dictionary {
    protected int size;
    protected DNode head, tail;
    public LinkedListDictionary() {
        size = 0;
        head = new DNode(null, null, null);
        tail = new DNode(null, null, null);
        head.setNext(tail);
    }
    public Object find(Object key) {
        if (size == 0) return null;
        else {
            :
        }
    }
    public void insert (Object key, Object data) throws DuplicatedKeyException {
        :
    }
}
```
Abstract Data Types

- Preferred way of designing and implementing data structures.
- Uses 2 general principles: information hiding and re-usability.
- Information hiding:
  - User data structure should not need to know details of its implementation.
  - We should be able to change implementation without affecting applications that use it.
  - Therefore, implementation information should be hidden.
Re-usability

- Re-usability:
  - If data structure is useful for one application, it is probably useful for another.
  - Therefore, we should design it to be as re-usable as possible.
The Position ADT

*Position* is an ADT with just one operation:

*element()*: Returns the data item stored at this position.