Documentation for new data structures:

Yifeng Liang

June 8, 2015

1. **Stack**

   A stack is a simple last-in-first-out collection of objects.

```
type Stack<T>= object & type{
   size -> Number
   // returns the number of elements in the stack

   push -> Done
   //adds and element to the top

   Pop -> T
   //removes the last element that was added to the top
}
```

2. Bi-directionaldictionary

   A Bi-directional dictionary is a dictionary that preserves the uniqueness of its values as well as that of its keys.

```
type BiDictionary<K,V>=Dictionary<K,V> & type{
      reversed -> BiDictionary<V,K>
      //retruns the inverse view of the dictionary, which maps each values to its associated key

      at(key:K)put(value:V) -> BiDictionary<K,V>
      //If value already exists, return. Otherwise, puts value at key

      at(key:K)forcePut(value:V) -> BiDictionary<V,K>
      //puts value at key. If value already exists, remove the existing value.

      at(value:V)putReversed(key:K)-> BiDictionary<V,K>
      //If key already exists, return. Otherwise, puts key at value

      at(value:V)forcePutReversed(key:K) -> BiDictionary<V,K>
      //puts key at value. If    key already exists, remove the the key
}
```

3. TreeSet

   A treeSet is an ordered set implemented by an AVL tree. It's iterator method returns an inorder iterator which allows user to iterate the elements in the sorted

order.

```
type TreeSet<T>=Set<T> & type{
    method iterator -> Iterator<T>
    //returns an in-order iterator
}
```

4. ImmutableSet
   A sorted immutable set using BST to store the data.

```
type ImmutableSet<T> = Set<T> & type {
    hash -> Number
    //returns the custom hash code of the BST tree.
}
```