Java

Collections

Collections

- The java.util package contains one of the Java's most powerful subsystems - The Collections Framework
- Some Interfaces
  - Collection, List, Set, Queue, Deque etc.
- Some Classes
  - ArrayList, LinkedList, Vector, Stack etc.
- Very useful while working with a huge number of objects

Collection Interface

- It is the foundation upon which the Collection framework is built (***interface Collection<E>***)
- It must be implemented by any class that defines a collection
- Some functions

boolean add(E obj)	*boolean addAll(Collection c)*
void clear()	*boolean contains(Object obj)*
boolean isEmpty()	*int size()*
boolean remove(Object obj)	*boolean removeAll(Collection c)*

List Interface

- ***interface List<E>***
- Some functions

void add(int index, E obj)

boolean addAll(int index, Collection c)

E get(int index)

int indexOf(Object obj)

int lastIndexOf(Object obj)

E remove(int index)

Deque Interface

- ***interface Deque<E>***
- Some functions

void addFirst(E obj)	*void addLast(E obj)*
E getFirst()	*E getLast()*
E peekFirst()	*E peekLast()*
E pollFirst()	*E pollLast()*
E pop()	*void push(E obj)*
E removeFirst()	*E removeLast()*

ArrayList

- It extends the ***AbstractList*** class and implements the ***List*** Interface.
- It is a variable length array of object references and can dynamically increase or decrease in size
- Constructors
  - ArrayList()
  - ArrayList(Collection c)
  - ArrayList(int capacity)
- ***Example****: ArrayListDemo(1-5).java*

LinkedList

- It extends the ***AbstractSequentialList*** class and implements the ***List***, ***Deque*** and ***Queue*** Interface
- It provides a linked-list data structure
- Constructors
  - LinkedList()
  - LinkedList(Collection c)
- ***Example****: LinkedListDemo.java*

Arrays

- The ***Arrays*** class provides various methods that are useful when working with arrays
- Some methods such as binarySearch, copyOf, copyOfRange, fill, sort are there
- ***Example****: ArraysDemo.java*

Vector

- It extends the ***AbstractList*** class and implements the ***List*** Interface
- It implements a dynamic array
- Constructors
  - Vector()
  - Vector(int size)
  - Vector(int size, int incr)
  - Vector(Collection c)
- ***Example****: VectorDemo.java*

HashTable

- It stores key-value pairs
- Neither keys nor values can be null
- When using HashTable, you specify an object that is used as a key and the value you want linked to that key
- The key is then hashed and the resulting hash code is used as the index at which the value is stored within the table
- ***Example***: *HashTableDemo.java*

HashMap

- It also stores key-value pairs like ***HashTable***
- Differences:

	HashMap	HashTable
Synchronized	No	Yes
Thread-Safe	No	Yes
Keys and values	One null key, any null values	Not permit null keys and values
Performance	Fast	Slow in comparison
Superclass	AbstractMap	Dictionary

- Use ***ConcurrentHashMap*** for multi-threading
- ***Example****: HashMapDemo.java*

Custom Comparator

- Required to sort a collection/array of custom objects
- Must implement the ***Comparable*** interface
- Must implement the following method

public int compareTo(Object o) {

}

- ***Example****: ComparatorDemo.java*