Import java.util.\*;

<https://www.hackerrank.com/> for practice

## COLLECTION Types Implementation Summary;

* For the Set interface, HashSet is the most commonly used implementation.
  + It is a Collection that, unlike List, does not allow duplicates.
  + HashSet allows at most one null element.
  + HashSet is faster than other implementations of Set, TreeSet and LinkedHashSet.

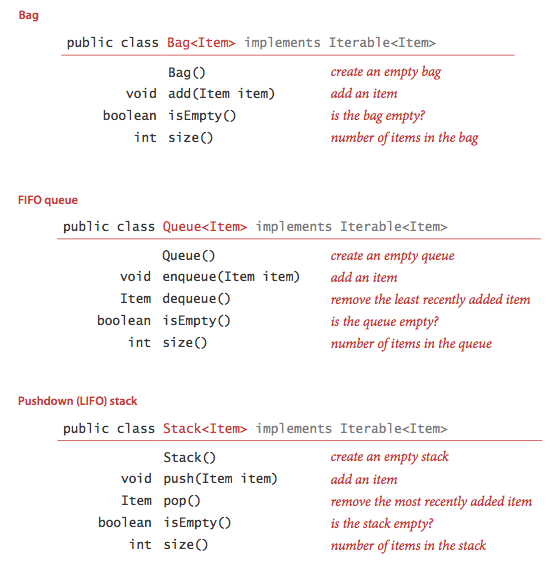
Set<Integer> set = new HashSet<Integer>();

* For the List interface, ArrayList is the most commonly used implementation.

List<MyObject> list = new ArrayList<MyObject>();

* For the Map interface, HashMap is the most commonly used implementation.

Map m1 = new HashMap();

peak/top returns first element without removing it.

## Reading Inputs

**Reading from file**

|  |  |
| --- | --- |
| import java.io.File;  import java.io.BufferedReader;  import java.io.FileReader;  import java.io.FileNotFoundException;  import java.io.IOException; | File file = new File("grid.txt");  BufferedReader reader = new BufferedReader(new FileReader(file));  String tempLine = reader.readLine(); |

**Reading from Console**

|  |  |
| --- | --- |
| import java.util.Scanner; | System.out.println("Enter your username: ");  Scanner scanner = new Scanner(System.in);  String username = scanner.nextLine();  System.out.println("Your username is " + username); |
| Import java util.regex.Matcher;  Import java util.regex.Pattern.  Splitting strings | String[] parts = string.split(Pattern.quote("."));  ([^-]+)-([^-]+) // Each part consists of characters other than - |

## SORTS

<http://www.java2novice.com/java-sorting-algorithms/>

|  |  |
| --- | --- |
| Bubble sort | Worst and average О(n2) |
| Selection sort | Θ(n2) |
| Insertion sort | О(n2) |
| Quick sort | Average Θ(n log(n)) worst Θ(n2) |
| Merge sort | O(n\*log(n)) |

## SEARCHES

|  |  |
| --- | --- |
| Binary Search (Divide and Conquer) |  |
| Binary Search (Recursive) |  |

## GRAPHS

Topological Sort

BFS for finding friendship levels

DFS

## Methods

|  |  |  |
| --- | --- | --- |
| Char | C = "abcdefghijklmn".toCharArray(); | Converts to char arrary |
| all | Arrays.toString(array); | Prints out array with ease |
| Maps | Map.get(“key”);  Maps.put(“key”, data);  **Maps.**[**size**](http://docs.oracle.com/javase/7/docs/api/java/util/Map.html#size())() | Gets element  Puts data in map. |
| Import java.util.LinkedList<E> | add(E e)  [**clone**](http://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html#clone())()  [**element**](http://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html#element())()  [**remove**](http://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html#remove())() | Appends the specified element to the end of this list.  Retrieves, but does not remove, the head (first element) of this list.  Retrieves and removes the head (first element) of this list. |
| Queue | Queue queue = new LinkedList();  queue.add("Java");  queue.remove("Java");  queue.poll();  queue.peek(); | Add to the back, remove from the front  Poll: retrieves and removes the head of this queue  .peek() just returns the current element |
| Stack | Stack st = new Stack() |  |

## Readers

*BufferedReader* is the main tool for inputting external data.

* *InputStreamReader* for reading from console.
* *FileReader* for reading from file!

# MVC

Architectural pattern for implementing user interface.

**Model**: Handles data, logic and rules of the application.