**Programming Contest Preparation**

**Useful Libraries and Information**

USACO High School Programming Competition Online Tests:

http://train.usaco.org/usacogate

ACM: online-judge.uva.es

Java Collections

http://java.sun.com/docs/books/tutorial/collections/interfaces/index.html

A set can only contain one of a particular element. There is also a sortedSet (treeSet) that extends set and adds an ordering.

A list can contain more than one element and is sorted.

A map has keys and values, like a hash table. There is also a sortedMap (treeMap) that extends a map and adds an ordering.

Collections use Generics:

http://java.sun.com/docs/books/tutorial/java/generics/index.html

Set<String> s = new HashSet<String>();

Useful classes, taken from the AP CS AB handbook (http://www.collegeboard.com/prod\_downloads/ap/students/compsci/52435%20APCompSci-Locked.pdf)

“**interface java.util.List<E>**

• boolean add(E x)

• int size()

• E get(int index)

• E set(int index, E x)

// replaces the element at index with x

// returns the element formerly at the specified position

• void add(int index, E x)

// inserts x at position index, sliding elements

// at position index and higher to the right

// (adds 1 to their indices) and adjusts size

• E remove(int index)

// removes element from position index, sliding

// elements at position index  1 and higher to the

// left (subtracts 1 from their indices) and adjusts size

// returns the element formerly at the specified position

• Iterator<E> iterator()

• ListIterator<E> listIterator()

**class ja va.util.ArrayList<E>**

**implements java.util.List<E>**

**class ja va.util.LinkedList<E>**

**implements java.util.List<E>, java.util.Queue<E>**

• Methods in addition to the List methods:

• void addFirst(E x)

• void addLast(E x)

• E getFirst()

• E getLast()

• E removeFirst()

• E removeLast()

**interface java.util.Set<E>**

• boolean add(E x)

• boolean contains(Object x)

• boolean remove(Object x)

• int size()

• Iterator<E> iterator()

**class ja va.util.HashSet<E>**

**implements java.util.Set<E>**

**class ja va.util.TreeSet<E>**

**implements java.util.Set<E>**

**interface java.util.Map<K,V>**

• V put(K key, V value)

• V get(Object key)

• V remove(Object key)

• boolean containsKey(Object key)

• int size()

• Set<K> keySet()

**class ja va.util.HashMap<K,V>**

**implements java.util.Map<K,V>**

**class ja va.util.TreeMap<K,V>**

**implements java.util.Map<K,V>**

(Use treeMap if you want a sorted Map, HashMap if you want a fast Map. Same with Set.)

**interface java.util.Iterator<E>**

• boolean hasNext()

• E next()

• void remove()

**interface ja va.util.ListIterator<E>**

**extends java.util.Iterator<E>**

• Methods in addition to the Iterator methods

• void add(E x)

• void set(E x)

**class java.util.Stack<E>**

• E push(E x)

• E pop()

• E peek()

• boolean isEmpty()

**interface java.util.Queue<E>**

• boolean add(E x)

• E remove()

• E peek()

• boolean isEmpty()

**class java.util.PriorityQueue<E>**

• boolean add(E x)

• E remove()

• E peek()

• boolean isEmpty()

Java.util.Vector resizable linked list—size and capacity can vary

.add(index, obj)

.add(object)

.get(index i)

.remove(index i)

isEmpty()

remove(object)

clear()

ArrayLists are similar to Vectors.

Double-ended queue – Stacks combined with queues

class java.util.deque<E> extends queue - new with Java 6

Methods, from http://java.sun.com/javase/6/docs/api/java/util/Deque.html

(special value functions return null or false appropriately):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **First Element (Head)** | | **Last Element (Tail)** | |
|  | Throws exception | Special value | Throws exception | Special value |
| **Insert** | [addFirst(e)](http://java.sun.com/javase/6/docs/api/java/util/Deque.html#addFirst%28E%29) | [offerFirst(e)](http://java.sun.com/javase/6/docs/api/java/util/Deque.html#offerFirst%28E%29) | [addLast(e)](http://java.sun.com/javase/6/docs/api/java/util/Deque.html#addLast%28E%29) | [offerLast(e)](http://java.sun.com/javase/6/docs/api/java/util/Deque.html#offerLast%28E%29) |
| **Remove** | [removeFirst()](http://java.sun.com/javase/6/docs/api/java/util/Deque.html#removeFirst%28%29) | [pollFirst()](http://java.sun.com/javase/6/docs/api/java/util/Deque.html#pollFirst%28%29) | [removeLast()](http://java.sun.com/javase/6/docs/api/java/util/Deque.html#removeLast%28%29) | [pollLast()](http://java.sun.com/javase/6/docs/api/java/util/Deque.html#pollLast%28%29) |
| **Examine** | [getFirst()](http://java.sun.com/javase/6/docs/api/java/util/Deque.html#getFirst%28%29) | [peekFirst()](http://java.sun.com/javase/6/docs/api/java/util/Deque.html#peekFirst%28%29) | [getLast()](http://java.sun.com/javase/6/docs/api/java/util/Deque.html#getLast%28%29) | [peekLast()](http://java.sun.com/javase/6/docs/api/java/util/Deque.html#peekLast%28%29) |

Collections implementations and methods, from the Sun Java Collections Overview Site,

http://java.sun.com/javase/6/docs/technotes/guides/collections/index.html

:

|  | | **Implementations** | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| **Hash Table** | **Resizable Array** | **Balanced Tree** | **Linked List** | **Hash Table + Linked List** |
| **Interfaces** | **Set** | [HashSet](http://java.sun.com/javase/6/docs/api/java/util/HashSet.html) |  | [TreeSet](http://java.sun.com/javase/6/docs/api/java/util/TreeSet.html) |  | [LinkedHashSet](http://java.sun.com/javase/6/docs/api/java/util/LinkedHashSet.html) |
| **List** |  | [ArrayList](http://java.sun.com/javase/6/docs/api/java/util/ArrayList.html) |  | [LinkedList](http://java.sun.com/javase/6/docs/api/java/util/LinkedList.html) |  |
| **Deque** |  | [ArrayDeque](http://java.sun.com/javase/6/docs/api/java/util/ArrayDeque.html) |  | [LinkedList](http://java.sun.com/javase/6/docs/api/java/util/LinkedList.html) |  |
| **Map** | [HashMap](http://java.sun.com/javase/6/docs/api/java/util/HashMap.html) |  | [TreeMap](http://java.sun.com/javase/6/docs/api/java/util/TreeMap.html) |  | [LinkedHashMap](http://java.sun.com/javase/6/docs/api/java/util/LinkedHashMap.html) |

* Collections [**sort(List)**](http://java.sun.com/javase/6/docs/api/java/util/Collections.html#sort%28java.util.List%29) - Sorts a list using a merge sort algorithm, which provides average-case performance comparable to a high-quality quicksort, guaranteed O(n\*log n) performance (unlike quicksort), and stability (unlike quicksort). (A stable sort is one that does not reorder equal elements.)
* sort(List, comparator)
* [**binarySearch(List, Object)**](http://java.sun.com/javase/6/docs/api/java/util/Collections.html#binarySearch%28java.util.List,%20T%29) - Searches for an element in an ordered list using the binary search algorithm.
* [**reverse(List)**](http://java.sun.com/javase/6/docs/api/java/util/Collections.html#reverse%28java.util.List%29) - Reverses the order of the elements in the a list.
* [**shuffle(List)**](http://java.sun.com/javase/6/docs/api/java/util/Collections.html#shuffle%28java.util.List%29) - Randomly permutes the elements in a list.
* [**fill(List, Object)**](http://java.sun.com/javase/6/docs/api/java/util/Collections.html#fill%28java.util.List,%20T%29) - Overwrites every element in a list with the specified value.
* [**copy(List dest, List src)**](http://java.sun.com/javase/6/docs/api/java/util/Collections.html#copy%28java.util.List,%20java.util.List%29) - Copies the source list into the destination list.
* [**min(Collection)**](http://java.sun.com/javase/6/docs/api/java/util/Collections.html#min%28java.util.Collection%29) - Returns the minimum element in a collection.
* [**max(Collection)**](http://java.sun.com/javase/6/docs/api/java/util/Collections.html#max%28java.util.Collection%29) - Returns the maximum element in a collection.
* [**rotate(List list, int distance)**](http://java.sun.com/javase/6/docs/api/java/util/Collections.html#rotate%28java.util.List,%20int%29) - Rotates all of the elements in the list by the specified distance.
* [**replaceAll(List list, Object oldVal, Object newVal)**](http://java.sun.com/javase/6/docs/api/java/util/Collections.html#replaceAll%28java.util.List,%20T,%20T%29) - Replaces all occurrences of one specified value with another.
* [**indexOfSubList(List source, List target)**](http://java.sun.com/javase/6/docs/api/java/util/Collections.html#indexOfSubList%28java.util.List,%20java.util.List%29) - Returns the index of the first sublist of source that is equal to target.
* [**lastIndexOfSubList(List source, List target)**](http://java.sun.com/javase/6/docs/api/java/util/Collections.html#lastIndexOfSubList%28java.util.List,%20java.util.List%29) - Returns the index of the last sublist of source that is equal to target.
* [**swap(List, int, int)**](http://java.sun.com/javase/6/docs/api/java/util/Collections.html#swap%28java.util.List,%20int,%20int%29) - Swaps the elements at the specified positions in the specified list.
* [**frequency(Collection, Object)**](http://java.sun.com/javase/6/docs/api/java/util/Collections.html#frequency%28java.util.Collection,%20java.lang.Object%29) - Counts the number of times the specified element occurs in the specified collection.
* [**disjoint(Collection, Collection)**](http://java.sun.com/javase/6/docs/api/java/util/Collections.html#disjoint%28java.util.Collection,%20java.util.Collection%29) - Determines whether two collections are disjoint, in other words, whether they contain no elements in common.
* [**addAll(Collection<? super T>, T...)**](http://java.sun.com/javase/6/docs/api/java/util/Collections.html#addAll%28java.util.Collection,%20T...%29) - Adds all of the elements in the specified array to the specified collection.
* [**newSetFromMap(Map)**](http://java.sun.com/javase/6/docs/api/java/util/Collections.html#newSetFromMap%28java.util.Map%29) - Creates a general purpose Set implementation from a general purpose Map implementation.
* [**asLifoQueue(Deque)**](http://java.sun.com/javase/6/docs/api/java/util/Collections.html#asLifoQueue%28java.util.Deque%29) - Returns a view of a Deque as a Last-in-first-out (Lifo) Queue.

Range-view operations on a sorted set

dictionary.subSet("f", "g").clear();

.subSet(“f”,”g”).size()

dictionary.headSet(object n) up to n, exclusive

dictionary.tailSet(Object n)

.first() last()

For each-loops java.sun.com/j2se/1.5.0/docs/guide/language/foreach.html

Instead of

void cancelAll(Collection<TimerTask> c) {

for (Iterator<TimerTask> i = c.iterator(); i.hasNext(); )

i.next().cancel();

}

we can use

void cancelAll(Collection<TimerTask> c) {

for (TimerTask t : c)

t.cancel();

}

**// a is an array of integers**

for (int i : a)

result += i;

Does not work for collections if you’re removing elements

Example with a map:

for (KeyType key : m.keySet())

System.out.println(key);

and with an iterator:

// Filter a map based on some property of its keys.

for (Iterator<Type> it = m.keySet().iterator(); it.hasNext(); )

if (it.next().isBogus())

it.remove();

Sorting an array (code taken from http://www.exampledepot.com/egs/java.util/pkg.html , Java Developer’s Almanac)

Arrays.sort(strArray, String.CASE\_INSENSITIVE\_ORDER);

// [a, C, z]

// Reverse-order sort

Arrays.sort(strArray, Collections.reverseOrder());

Convert an array to a Collection

// Fixed-size list

List list = Arrays.asList(*array*);

// Growable list

list = new LinkedList(Arrays.asList(*array*));

// Duplicate elements are discarded

Set set = new HashSet(Arrays.asList(*array*));

Converting a Collection to an Array

.toArray() returns an array of Objects. If you want an array of a particular kind, you have to cast it as follows.

// Create an array containing the elements in a list

Object[] objectArray = *list*.toArray();

*MyClass*[] array = (*MyClass*[])*list*.toArray(new *MyClass*[*list*.size()]);

// Create an array containing the elements in a set

objectArray = *set*.toArray();

array = (*MyClass*[])*set*.toArray(new *MyClass*[*set*.size()]);

// Create an array containing the keys in a map

objectArray = *map*.keySet().toArray();

array = (*MyClass*[])*map*.keySet().toArray(new *MyClass*[*set*.size()]);

// Create an array containing the values in a map

objectArray = *map*.values().toArray();

array = (*MyClass*[])*map*.values().toArray(new *MyClass*[*set*.size()]);

Subarrays with Java 6

int[] newArray = Arrays.copyOf(a, newLength);

public static int[] **copyOfRange**(int[] original,

int from,

int to)

Math.Random()

Random.nextInt(int) returns [0,int)

float [**nextFloat**](http://java.sun.com/j2se/1.4.2/docs/api/java/util/Random.html#nextFloat%28%29)() [0.0,1.0]

**Strings, Characters, and I/O**

Use the Scanner class for reading input.

.nextInt() nextLine() hasNext() nextFloat();

new Scanner(new File(“prob.in”));

String substring(int from, int to)

// returns the substring beginning at from

// and ending at to-1

String charAt(char)

StringBuffer buffer = new StringBuffer(128);

New String(StringBuffer);

yourStringBuffer.append(string);

String.toLowerCase() toUpperCase() equals()

Can use ‘C’ –‘A’ + ‘a’ to convert to lower case—or Character.toLowerCase

Can also use character codes as above to index an array of characters instead of ints—be creative!

.startsWith(string) trim()

.indexOf(char) .indexOf(string)

NumberFormat—can use for output

NumberFormat nf = new NumberFormat();

nf.setMinimumFractionDigits( 3 );

nf.setMaximumFractionDigits( 3 );

nf.setMinimumIntegerDigits( 1 );

nf.setMaximumIntegerDigits( 10 );

string = nf.format(int);

Easy toString: concatenate starting with a string,

Or an empty string: “” + yourint – no toString needed

Of course you’re familiar with Integer.parseInt(String) and related classes.

Character.isDigit() .isLetterOrDigit() .isLetter() isSpace()

String [] = yourString.split() instead of tokenizer – uses regular expressions

You can use .split() on the nextLine() from a Scanner.

Split(“\\s”) splits with spaces as separators

\s stands for spaces, \\ is so the backslash is recognized

\S all but whitespace

java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html

[abc] only characters a,b,c

^ negation

[a-zA-Z] inclusive range union

ACM/Programming Contest Judge Feedback

Accepted (AC)

Presentation Error (PE) – The I/O is off. They’re picky with this—be careful.

Accepted (PE) – Accepted with presentation error, like extra blank at end of line

Wrong Answer(WA) – Be sure to look at boundary cases and ensure you’re solving the right problem. You won’t be able to know what the judge’s input is to test.

Compile Error (CE)

Runtime Error (RE)

Time Limit Exceeded (TL) – You probably won’t have to worry about this for the contest. Make sure you’re outputting to System.out.

Memory Limit Exceed (ML)

Output Limit Exceeded (OL) – For this and the two above, watch out for infinite loops.

Restricted Function (RF) – In the real contest, some functions may be restricted, like BigInteger. Check with the judges for details.

Submission Error (SE) – You gave an incorrect user ID or problem number—the automated system we use takes care of this.

You can have more than one problem with your program and only have one error message reported to you—there’s a hierarchy. You’ll get a PE before a WA, for example.