Lists

* ArrayLists(Collection Data Type)
  + ArrayList<String> list = new ArrayList<String>();
    - How to make an ArrayList that can only hold Strings, however, you can also add other data types such as Integers
  + Able to use the Add method, Remove method, size() method, and isEmpty() method. (Because it’s a collection data type)

Adding and Removing

* Adding: <ArrayListName>.add(integer/string/other collections);
* Removing: <ArrayListName>.remove(Index);
* Set or change to: <ArrayListName>.set(index, new element) Changes the index to the new element

Iterating

* Use loops to loop through an array
  + Ex. ArrayList<Integer> list = new ArrayList<Integer>();

list.add(1); list.add(7); list.add(4); list.add(5);

list.add(2);

int[] arr = {1, 7, 4, 5, 2};

for(int i = 0; i < list.size(); i++){ // regular

System.out.println(list.get(i));

}

Dynamic Arrays

* Can do everything an Array can do but **reallocate storage to accommodate more elements as they are added.**
* *In array-based contest problems, we'll use one-, two-, and three-dimensional static arrays most of the time. However, we can also have static arrays of dynamic arrays, dynamic arrays of static arrays, and so on. Usually, the choice between a static array and a dynamic array is just personal preference.(Need help)*

Pairs and Tuples

* Stores points in a 2D plane using pairs
  + Making a Pair array
    - static class Pair<K, V> {

K first;

V second;

public Pair(K first\_value, V second\_value) {

first = first\_value;

second = second\_value;

}

}

* + - Pair<Integer, String> p = new Pair(5, "hello");

System.out.println(p.first + " " + p.second);

Hashmaps

* Making a Hashmap
  + HashMap<String, Integer> people = new HashMap<String, Integer>();
    - Uses key/value to store items
      * The first <String represents that the keys will be strings, and the next , integer> will represent the type of values that will go in the keys
* Adding values
  + Hashmap.put(<Key>, <value>);
* Access an item
  + Hashmap.get(<key>);
* Removing values
  + Hashmap.remove(<key>);
* Size
  + Hashmap.size();
* Looping through a hashmap
  + Use keyset() to loop through the keys and values() to loop through the values
  + Only works with for (String i : capitalCities.keySet()) {

System.out.println(i);

}

Hashsets

* Every item is unique
* Making a Hashset
  + HashSet<String> cars = new HashSet<String>();
    - Every string is unique
    - Ex. cars.add("BMW"); will add BMW to the

Memory

* Usaco’s memory limit is 256 MB
  + In bytes: 256 \* 10^6
  + Ex. an int value is limited to values under 64 \* 10^6
* Note that Program Overheads may reduce memory available