Exam One Cheat Sheet
by Brian Lin

Comparisons

Array[]	vs.	ArrayList<String>
new String[n]		new ArrayList<String>()
.length		.size()
[k]		.get(k)
Arrays.sort(w)		Collections.sort(ww)

Comparable	vs.	Comparator
compareTo method		compare method
· "natural" order		· mult. sorting methods
· ⊖ if this < o2		· 2 arguments
· 0 if this == o2		
· ⊕ if this > o2		

Conversions

String → String[]
```
s.split("\\s+");
```

String[] to Set/List
```
List<String> myList = Arrays.asList(sA);
```
(for a set)

List/ArrayList → Set
```
Set<String> set = new TreeSet<String>(list);
```

Set/List → String[]
```
String[] sA = mySet.toArray(new String[mySet.size()];
```
(myList) (myList)

Coll → String
```
String out = new String();
for (String elem : input)
    out = out + " " + elem;
return out.trim();
```

String → Syllables
```
pb SC] stoS(s s){
    SB b = n SB(s);
    f(i=0 → b.toS().l()-1){
        if (isVowel(b.toS().c@(i))...
        && !isVowel(b.toS().c@(i+1)))
            b.insert(i+1, space);}
    r b.toS().split("\\s");}
```

Scanner

```
Scanner in = new Scanner(myString);
String whole = in.useDelimiter("\\Z").next();
in.close(); // after you're done w/ the Scanner
List<Set<String> myList = new ArrayList<String>();
while (in.hasNext()) {
    String[] words = in.next+Line.split("\\s+");
    Set<String> mySet = new TreeSet<String>
      ... (Arrays.asList(words));
    for (String elem : words) {
        mySet.add(elem);
    }
    myList.add(mySet);
}
return myList;
```

Scaling Hash

```
pb int hashCode(){
    int h=0;
    h+= myState.hC() + myRegion.hC() *3;
    for (k=9, λ=9; k<myMembers.l(); k++, λ*=3){
        h+= myMembers.hC() * λ;
    return h;
}
```

Forbes' Pro Stratz

Reverse Sort →
```
Arrays.sort(a, Collections.reverseOrder())
Collections.reverse(arrList);
```
```
TreeSet<String>((s1,s2) → -s1.compareTo(s2));
```

Methods

String
charAt(i)
compareTo(S)
concat(s)
contains(s)
equals(o)
hashCode()
indexOf(c)
length()
replace(c1, c2)
split(regex)
substring(i1,i2)
trim()
valueOf(c)

Map
entrySet()
containsKey/Value
entrySet()
get(key)
keySet()
put(k,v)
remove(k)
size()
values()

List
add(e)
addAll(coll)
contains(o)
get(i)
indexOf(o)
remove(o)
retainAll()
size()
sort(?)
toString()

SB
append(?)
charAt(i)
delete(i1, i2)
insert(i, ?)
length()
toString()
reverse()

Arrays
asList()
sort()
toString()

Integer
intValue()
equals(o)
compareTo(I)
valueOf(?)

Big O

N=100 → t = .5 ms

N=500 → t = ? using nlogn

.5 = k(100 log₂ 100)

? = k(500 log₂ 500)

Bullshit Harry Potter

```
Class 100: First Mid Last: bs@duke.edu
⋮
pb AL<S> mailsTargetCourse(AL<S> all){
    pb AL<S> result = n AL<S>();
    M<S,AL<S>> mp = n HM<S, AL<S>>();

    // make map
    int max=0;
    S maxKey= "";
    f(S elem : all){
        S[] items = elem.split(": ");
        S course = items[0];
        S email = items[1]
        AL<S> emails = mp.get(course);
        if (emails == null) {
            emails = new AL<S>();
            mp.put(course, emails);
        }
        emails.add(email);
        if (emails.size()>max){
            max = emails.size();
            maxKey = course;
        }}
        r coll.sort(mp.get(maxKey));
```

CloseToOrigin

```
pb S Point closestToOrigin(int[] xV, int[] yV){
    d min = Double.MAX_VALUE;
    Point o = n Point(0,0);
    Point c = null;
    f(k → xV.l() && yV.l(); k++)
      Point curr = n Point(xV[k], yV[k]);
      if (o.distFrom(curr) < min) {
          c = curr;
          min = o.distFrom(curr);
      }
    }
    return c;
}
```

MemberCheck

2 param union — .addAll (a)
.addAll (b)

2 param inter. — .addAll(a)
.retainAll(b)

List<Set> Union

```
Iterator<Set<String>> myIt = sets.iterator();
while(myIt.hasNext())
    results.addAll(myIt.next());
```

SesameNames

in: String[] = {"VALUE key1 key2"...}

out: String[] = {"key ==> VALUE"...}

1) Create a map, "myMap"
2) Iterate over map:

```
for (Map.Entry<String, List<String>> entry : myMap.entrySet())
    mySet.add(entry.getKey() + "_==>_" + ...
    entry.getValue.toString().replace("[","").replace("]",""));

return mySet.toArray(new String[myMap.values().size()]);
```

ClassScores

↳ Find mode of int[]

- create int[] to count
- populate + find max
- create output []

WordNgram

```
public class WordNgram implements Comparable<WordNgram> {
    public string[] words;
    public int myLength;
    public WordNgram(String[] list, int start, int n) {
        myWords = new String[n];
        System.arraycopy(list, start, myWords, 0, n);
        myLength = myWords.length;

    public String lastWord() {
        return myWords[myLength-1]; }

    public int compareTo(WordNgram wg) {
        return Arrays.toString(myWords).compareTo(Arrays ...
          .toString(wg.myWords)); }
    public boolean equals(Object o) {
        if (this==o) // point to same obj. in memory
            return true;
        if (o==null || getClass()!=o.getClass())
            return false;
        WordNgram wg = (WordNgram) o;
        for (int k=0; k<myWords.length; k++) {
            if (!myWords[k].equals(wg.myWords[k]))
                return false;
        }
        return true;
    }
    public int hashCode() {
        int sum = 0;
        for (int k=0; k<myWords.length; k++)
            sum = 100 * sum + myWords[k].hashCode();
        return sum; }
```

CountUp

{1, 1,2, 1,2,3...} (n=3)

```
public static int[] countUp(int n) {
    int[] result = new int[n*(n+1)/2];
    for (int index=0, i=1; i<n; i++)
        for (int j=1; j<=i; j++)
            result[index++] = j;
    return result;
}
```

Fuckin' John

```
public int nameSum(String s){
    int sum = 0;
    for (int k=0; k<s.length(); k++)
        sum += s.charAt(k) - 64;
    return sum; }
```

SortByFreqs

```
public class SortByFreqs {
    private TreeMap<String, Integer> myWC;
    class FreqCompare implements Comparator<String>{
        public int compare(String o1, String o2)
            return myWC.get(o2)-myWC.get(o1); }
    public String[] sort(String[] data) {
        Δ put stuff in map
        String[] out = myWC.keySet()...
            .toArray(new String[myWC.size()]);
        Arrays.sort(out, new FreqCompare());
        return out; }
```

Thesaurus

```
boolean refresh = true;
int refreshCount = 0;

for (int i=0; i<myArray.size() + refreshCount; i++){
    for (int j=0; j<myArray.size(); j++){
        refreshCount = 0; // Avoid out of bounds error
        if(refresh){ i=0; j=0; }  // Restart from beg.
        refresh = false
        if (i!=j){
        ...Set1 = myArray.get(i);
        ...Set2 = myArray.get(j);
            if(numInCommon(set1, set2)>1){
                refresh = true;
                refreshCount ++;
            ...UnionSet = Union(set1, set2);
                myArray... .add(unionSet);
                           .remove(set1);
                           .remove(set2);
```

ClientsList

```
public String nameBuilder(String s) {
    if (s.contains(", ")) return ...
    printFirst(s)+" "+printLast(s)).trim();
    else return s.trim(); }
```

Markov Generation

```
for (int i=0; i<maxWords; i++) {
    List<WordNGram> myList = myMap.get(seed); //Start point
    WordNGram myK = myList.get(myRandom.nextInt(myList.size());
    if(myK==null) // E-o-f
        return build.toString().trim();
    String myWord = myK.lastWord();
    build.append(" ");  //add a space
    build.append(myWord); //add a word
    seed = myK; // New seed is chosen WordNgram
}
return build.toString().trim();
```

Markov Mapping

```
if(!myMap.contains(thisK)) {
    ArrayList<String> myAL = new ArrayList<String>();
    myAL.add(nextK);
    myMap.put(thisK, myAL);
}
else myMap.get(thisK).add(nextK);
```