CS201 Discussion 3

FUN WITH SCANNERS
Before we begin...

Questions about n-Body?
A quick review of StdDraw

Methods in StdDraw you will use:

.setXscale(a, b) (Also setYscale(a, b)): Sets the left boundary to x-position a and the right boundary to x-position b.

.show(n): Waits n milliseconds, then shows the current display
  ◦ Useful for smooth animation independent of computational time.
  ◦ Default is n = 0.

.picture(x, y, filename): Displays the image named filename centered at (x, y) on the display.
General outline for drawing in nBody

Call .setXscale() and .setYscale()

For each time step:
  .picture() every object
  .show(n) to show the new display

(Obviously, there are calculations to be made between these steps)
Warmup: ClassScores

Review the following problem:

Problem Statement

A teacher has just finished grading the test papers for his class. To get an idea of how difficult the test was, he would now like to determine the most common score on the test. In statistics, this is called the "mode" of a set of data points. For instance, if the scores were {65, 70, 88, 70}, then the mode would be 70, since it appears twice while all others appear once.

Sometimes, in the case of a tie, the mode will be more than one number. For instance, if the scores were {88, 70, 65, 70, 88}, then the mode would be {70, 88}, since they both appear most frequently.

You are given an int[] scores. You are to return an int[] representing the mode of the set of scores. In the case of more than one number, they should be returned in increasing order.

More or less, all the problem is asking is for you to find the modes of an array and return them as a sorted array
Things to think about

How can we find the mode(s) of an array in two steps?

We don’t necessarily know what the size of our output is going to be when calculating it. What data structure can we store our results in before returning the final array to get around this problem?

How could we use Arrays.sort() or Collections.sort() to make this problem easier?

Given the original integer array, how can we find the max frequency of any element?
Things to think about

How can we find the mode(s) of an array in two steps?

1. Find the max frequency of any element
2. Find all elements with that frequency

We don’t necessarily know what the size of our output is going to be when calculating it. What data structure can we store our results in before returning the final array to get around this problem?

An ArrayList – ArrayLists have variable size and can be made into arrays easily.

How could we use Arrays.sort() or Collections.sort() to make this problem easier?

Obviously, we can use it to sort our final output. We could also sort the input to make finding the mode easier.

Given the original integer array, how can we find the max frequency of any element?

We’ll get to this on the next slide...
Finding the max frequency

Given the original integer array, how can we find the max frequency of any element?

There are multiple ways to approach this:

◦ One way is, for each element, count how many duplicate exists in the remainder of the array. If this frequency is larger than any past frequency, store it.

◦ Another way is to sort the list using Arrays.sort(). This will put all elements of the array of the same type next to each other. Then, you just have to find the largest sequence of elements.

Is either approach better than the other? For APTs, does it matter?

Once you know the max frequency of any element, finding the elements with that frequency is pretty similar – the approach will be rather similar to finding the max frequency elements, except rather than storing the frequency, you can just add the high-frequency element to a List.
Other comments

Once you have your output as a List, all you need to do is convert it to array form and sort it.

Unfortunately, int is a primitive type, so to make the List into an array, you’ll have to copy the output element by element.

For non-primitive types like Strings, you can easily convert Lists to arrays using the following snippet. This may be useful for future APTs 😊

```java
ArrayList<String> list = new ArrayList<String>();
//add elements to list here
String[] array = list.toArray(new String[list.size()]);
```

Lastly, there is a data structure you will soon learn about in lecture called Map, which makes this problem much easier. If you feel confident in your ability to self-teach, feel free to try using that instead...
Today’s main topic

Scanners! Useful for file reading

(Thankfully, not this kind:)
Intro to Scanners

Scanners – useful for parsing text input piece-by-piece

Important things to know about scanners:
  ◦ Constructors
  ◦ Reading Tokens
  ◦ Delimiters
Why use Scanners?
The easiest way to read from external text files in Java
Why use files?
Scanner Constructors

Scanner comes with a ton of constructors for your use.

The two you are most likely to use in this class:

◦ `new Scanner(new File(sourcefile))`
  ◦ Opens a text file. `sourcefile` is the path to the file you want to read.

◦ `new Scanner(String sourcestring)`
  ◦ Reads from a String `sourcestring`.

◦ Why might we use this instead of just a regular File source?
Reading Tokens

Scanner comes with many methods for reading the next token. The most common:

- `.next()` – Reads the next unread token and returns it as a String
- `.hasNext()` – Returns true if there is a next token (i.e. if `.next()` would not throw an error)
- `.nextInt()`, `.nextDouble()`... - Reads the next unread token and returns it as a primitive type if possible (if not, throws an error)
- `.hasNext___()`... - Returns true if `.next___()` would not throw an error.
Delimiters

What is a delimiter?

◦ Spaces are to words in English as delimiters are to tokens in Scanners
Useful delimiters

"\s+" – Generic whitespace (use to read word-by-word in files with tabs and new lines)

"\\Z" – End-of-file (use to read entire files at once)
Example 1: Changing the delimiter

Without running it, what is the output of the following snippet:

```java
Scanner input = new Scanner("banana man");
input.useDelimiter(delimiter);
while(input.hasNext()){  
    System.out.println(input.next());
}
```

If delimiter is "a"? "n"? "\Z"? "\s+"?
Example 2: ClassScores

Suppose you had the ClassScores APT done. You could test it using the following main method:

```java
public static void main(String[] args){
    ClassScores tester = new ClassScores();
    int[] input = {1, 2, 2, 3, 3, 3};
    System.out.println(tester.findMode(input));
}
```
Using Scanner to test

Rewrite the main method from the previous slide to initialize the variable input from a text file called input.txt with one element of the input on each line.

- Make sure you can initialize correctly regardless of the length of input.txt.
- When you finish ClassScores later, try testing this.
Example 3: Using .next() properly

Look at the NumberPrinter class, specifically the method printSmallNumbers. The class is supposed to read in a file containing only numbers, and print numbers which are at most 5.

Try running the class – does the class work as described? Why or why not? With a partner, rewrite the method to fix this.
Example 4: Filtering

Occasionally, we may want to extract only certain parts of a Scanner out.

Look at the source file input2.txt – it has comments in it. Try running NumberPrinter and observe what happens.

With a partner, implement a fix so that NumberPrinter prints the small numbers in input2.txt correctly.
Example 5: Filtering comments

Look now at input3.txt. It also has comments, but now these comments include numbers.

Try running your NumberPrinter using input3.txt and observe what happens.

Assuming all comments will start with (* and end with *), with a partner modify printSmallNumbers to ignore these comments.