Notes from the AverageFinder.java program

- `readFile()` shows an example of throwing an exception
  - defer handling the exception to the constructor, which calls `readFile()`
- a boolean variable is used to indicate when we've read the first line of the file
  - the first line is different from the rest, so we have to treat it differently
  - the first line contains the dimensions for the array

split()

- From the String class
- “Splits” a string into pieces
- Pieces are marked by a specified character (pass in as input parameter)
- Returns an array of Strings

split() example

```java
String[] parts;
String line1 = "this is my line";
String line2 = "this,is,my,line";
parts = line1.split("\s");
parts = line2.split(",");
```
split() example (cont.)

- In each case, after calling `split()`, `parts[]` contains the same items:
  - `parts[0] = "this"
  - `parts[1] = "is"
  - `parts[2] = "my"
  - `parts[3] = "line"
- `line1.split("\s")` splits on any space character
- `line2.split("," )` splits on commas

Regular expressions

- Used when we want to specify a number of possible characters of a particular type
  - digits, letters, words, spaces
- Most common examples:
  - `\s`: any space character (space[s], tab, etc.)
  - `\d`: any digit (0-9)
  - `\w`: any combination of letters and numbers, no spaces
  - `[a-zA-Z]`: any letter

Regular expressions in Java

- In Java, the “\” character denotes the start of an “escape sequence” in a string
  - examples: “\n” (carriage return), “\t” (tab)
- Therefore, if we want to specify a regular expression in a string, we have to “escape” the escape sequence indicator
  - in other words, “\s” --> “\s” and “\d” --> “\d”

2-dimensional arrays

- position is denoted by a row and a column
- example: `scores[3][4]` means “the value in the array named score that’s in the fourth row and the fifth column.” (remember: array indices are zero-based!)
- for loops are often used in dealing with arrays
Adding an element to an array

- if no space left in array,
  - create a new array (size = size of old array + 1)
  - copy each element of the old array into the new array
  - add the new element
- else,
  - add the new element to the end of the array

Adding a new element to the middle of an array

- if no space left in array,
  - create a new array (size = size of old array + 1)
  - copy each element of the old array into the new array, up until the space where the new element would go
  - place new element into the array at the designated spot
  - continue copying the rest of the elements
- else, (see next page)

Adding a new element to the middle of an array (cont.)

- locate the spot in the array to place the new element
- for each element from the last element to the element after this spot,
  - move the element down by one spot
- insert the new element into the now-empty slot

Deleting an element from the middle of an array

- locate the element to be deleted in the array
- delete the element
- for each element from this point until the end of the array,
  - move the element up by one spot
ArrayList

- Java's built-in array class
- Handles these tasks automatically:
  - allocating space for an array
  - adding and deleting items from an array (any location)
- Only handles items of type Object!
  - grandparent class to all other reference data types
  - have to “cast” items as we pull them out of the array
  - primitive data types must be treated as reference data type equivalents (i.e., int --> Integer object)
- Use the other method of declaring and using arrays in this class