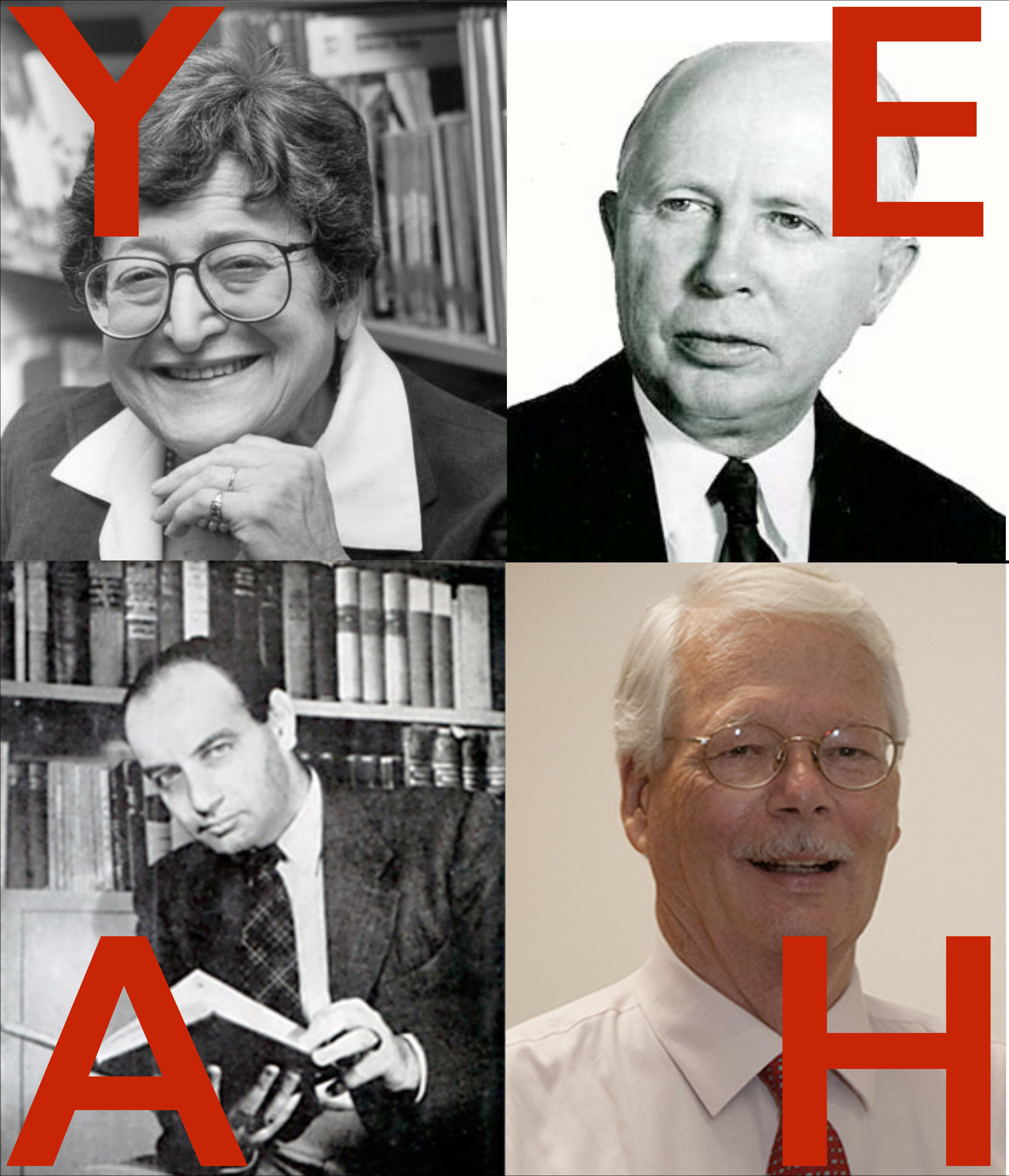


106A assignment review #4

13 Feb 2014
5:30p-6:30p

Miles Seiver

`char`

important!
here's how you convert
a char to a String

pdating a char

```
char ch = 'a';
Character.toUpperCase(ch);
println("" + ch);          //output: "a"
```

✗

```
char ch = 'b';
ch = Character.toUpperCase(ch);
println("" + ch);          //output: "B"
```

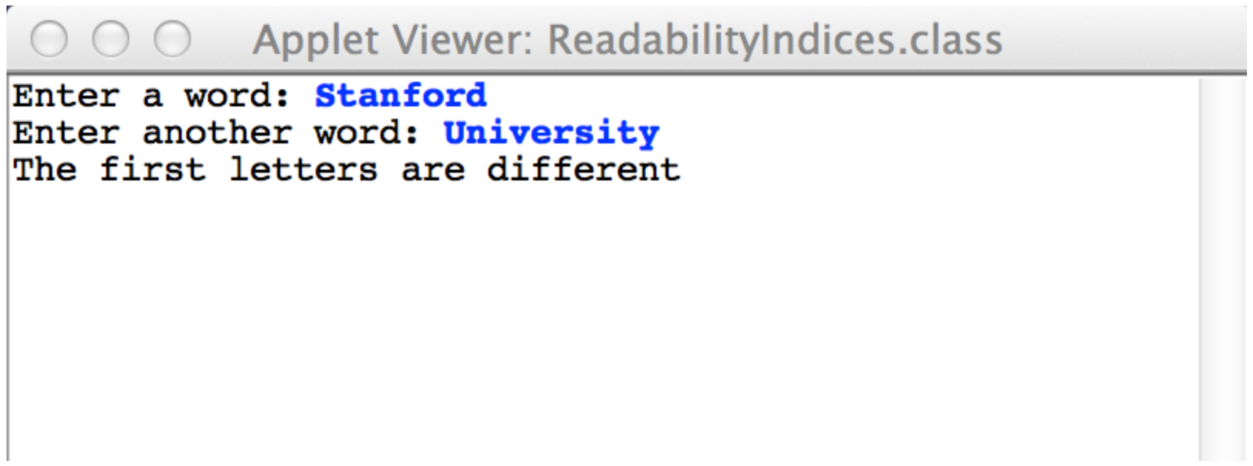
✓

Other char methods

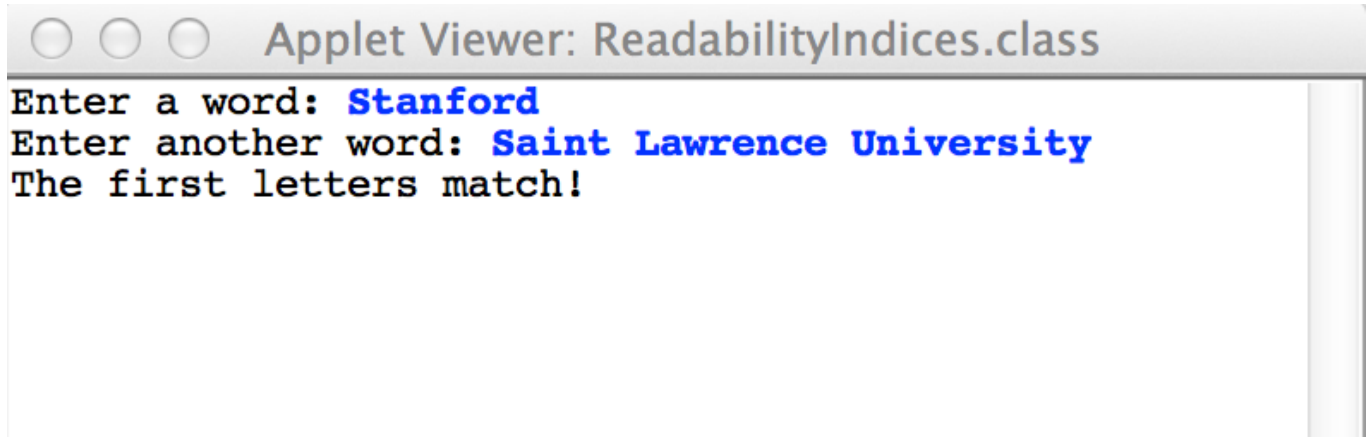
`static boolean isDigit(char ch)`	Determines if the specified character is a digit.
`static boolean isLetter(char ch)`	Determines if the specified character is a letter.
`static boolean isLetterOrDigit(char ch)`	Determines if the specified character is a letter or a digit.
`static boolean isLowerCase(char ch)`	Determines if the specified character is a lowercase letter.
`static boolean isUpperCase(char ch)`	Determines if the specified character is an uppercase letter.
`static boolean isWhitespace(char ch)`	Determines if the specified character is **whitespace** (spaces and tabs).
`static char toLowerCase(char ch)`	Converts **ch** to its lowercase equivalent, if any. If not, **ch** is returned unchanged.
`static char toUpperCase(char ch)`	Converts **ch** to its uppercase equivalent, if any. If not, **ch** is returned unchanged.

Comparing chars

- Let's write some code to:
  - prompt the user for two words
  - print out "they match" if the first letters of the two words are the same and print out "they don't match" otherwise



```
Enter a word: Stanford
Enter another word: University
The first letters are different
```



```
Enter a word: Stanford
Enter another word: Saint Lawrence University
The first letters match!
```

Version 1

```
String first = readLine("Enter a word: ");
String second = readLine("Enter another word: ");

if (first.charAt(0) == second.charAt(0))
  println("The first letters match!");
else
  println("The first letters are different");
```

Now let's make the code case-insensitive.

Version 2

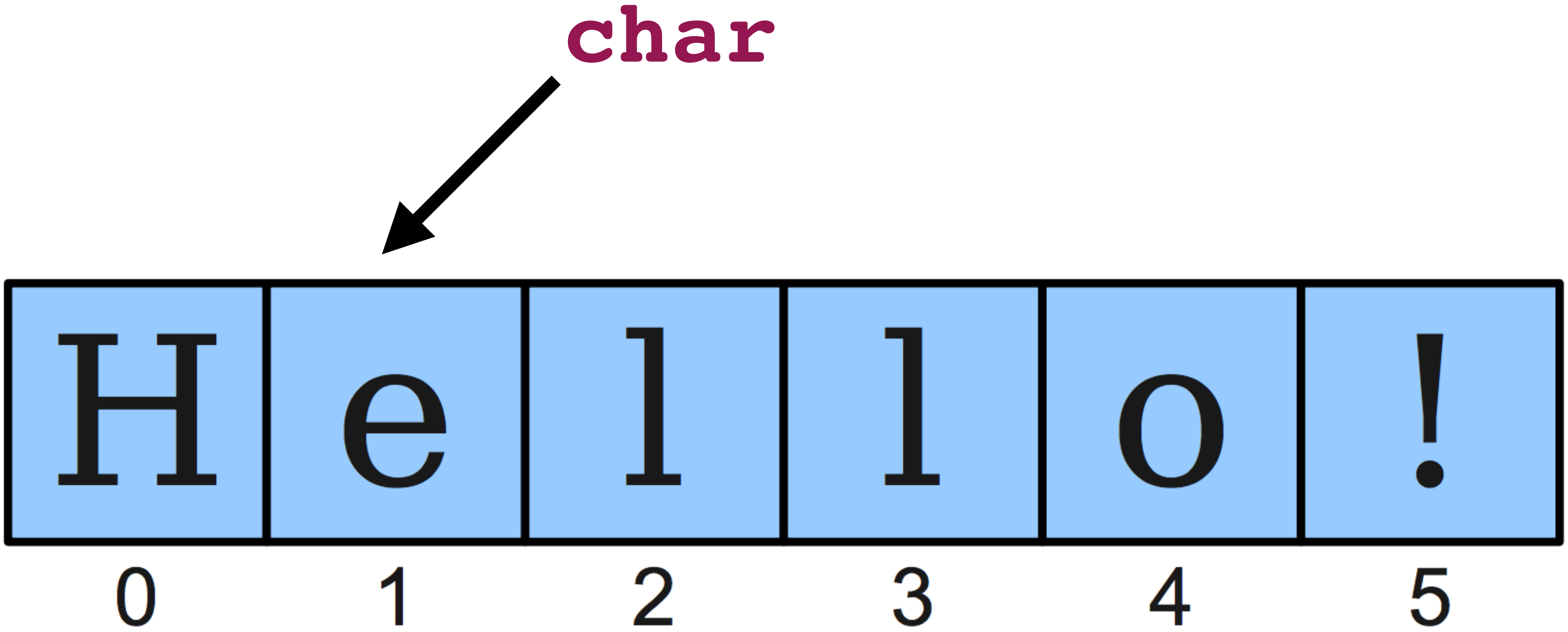
```
String first = readLine("Enter a word: ");
String second = readLine("Enter another word: ");

if (Character.toLowerCase(first.charAt(0)) ==
    Character.toLowerCase(second.charAt(0)))
  println("The first letters match!");
else
  println("The first letters are different");
```

There's still an edge-case bug. How would we fix it?

(You will encounter this same bug in the assignment.
Make sure you handle it!)

String



string`.charAt(`***index***`)`

Comparing Strings

```
String s1 = "racecar";
String s2 = reverseString(s1);
if (s1.equals(s2)) {
    /* … s1 and s2 are equal … */
}
```

✓

~~`if (s1 == s2) {…`~~

Updating a String

```
String str = "hello";
str.toUpperCase();
println(str);            //output: "hello"
```

✗

```
String str = "hello";
str = str.toUpperCase();
println(str);            //output: "HELLO"
```

✓

Other `String` methods

`int length()`	Returns the length of the string
`char charAt(int index)`	Returns the character at the specified index. Note: Strings indexed starting at 0.
`String substring(int p1, int p2)`	Returns the substring beginning at **p1** and extending up to but not including **p2**
`String substring(int p1)`	Returns substring beginning at **p1** and extending through end of string.
`boolean equals(String s2)`	Returns true if string **s2** is equal to the receiver string. This is case sensitive.
`int compareTo(String s2)`	Returns integer whose sign indicates how strings compare in lexicographic order
`int indexOf(char ch)` *or* **`int indexOf(String s)`**	Returns index of first occurrence of the character or the string, or -1 if not found
`String toLowerCase()` *or* **`String toUpperCase()`**	Returns a lowercase or uppercase version of the receiver string

Remember this?

```
String first = readLine("Enter a word: ");
String second = readLine("Enter another word: ");

if (Character.toLowerCase(first.charAt(0)) ==
    Character.toLowerCase(second.charAt(0)))
  println("The first letters match!");
else
  println("The first letters are different");
```

How could we accomplish case-insensitivity using `String` methods instead?

Version 2.5

```
String first =
  readLine("Enter a word: ").toLowerCase();
String second =
  readLine("Enter another word: ").toLowerCase();

if (first.charAt(0) == second.charAt(0))
  println("The first letters match!");
else
  println("The first letters are different");
```

(There's still an edge-case bug, though.)

ugly way to work with `Strings`

(often more complex to use than other approaches,
but use ‘it if you think it’s the best way to get the job done!)

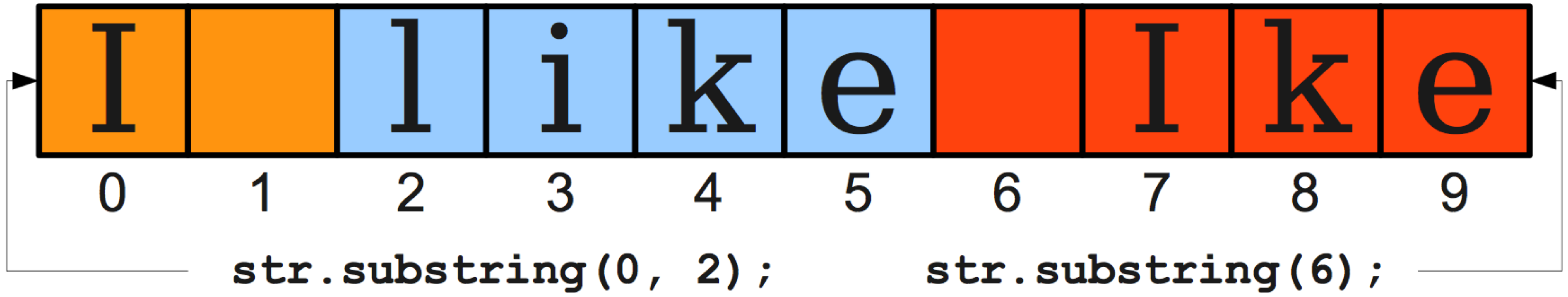
Obtaining Substrings

- To get all of the characters in the range [start, stop), use

string **.substring(*****start*****,** ***stop*****)**

- To get all of the characters from some specified point forward, use

string **.substring(*****start*****)**



nice way to work with Strings

How to approach a `String` problem

- Start with nothing and build up the result
- Iterate left to right or right to left?
- Use `Character` methods at each position to build the new String
- Use `StringTokenizer` or `.substring` only if necessary
    - Avoid them for this assignment

the assignment

Readability Indices

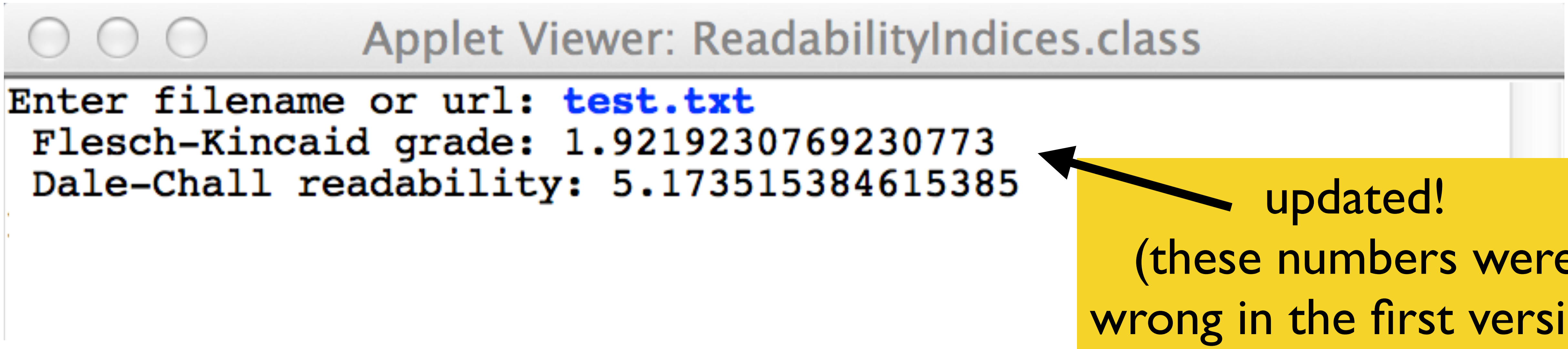
due Wed, 19 Feb @ 3:15pm

Flesch-Kincaid grade level test

Dale-Chall readability score

test.txt

```
I think, therefore I am.
I am (because I think).
I think I am, therefore I am. (I think?)
There is no period in this sentence
```



```
Enter filename or url: test.txt
 Flesch-Kincaid grade: 1.9219230769230773
 Dale-Chall readability: 5.173515384615385
```

updated!
(these numbers were wrong in the first version of these slides)

The ***Flesch-Kincaid grade*** estimates at what grade level a reader would have to be in order to comprehend a text.

The ***Dale-Chall readability difficulty*** estimates the same thing but uses a chart to interpret the number.

Score	Difficulty
0 – 5	Readable by an average 4th grader.
5 – 6	Readable by an average 5th or 6th grader.
6 – 7	Readable by an average 7th or 8th grader.
7 – 8	Readable by an average 9th or 10th grader.
8 – 9	Readable by an average 11th or 12th grader.
9 – 10	Readable by an average college student.
10+	Readable by an average college graduate.

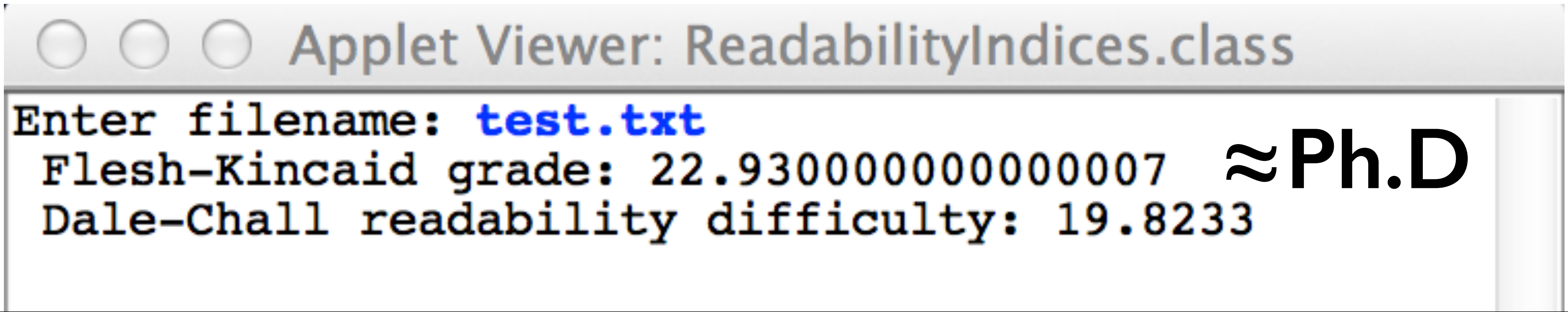
test.txt

```
Buffalo buffalo Buffalo buffalo buffalo
buffalo Buffalo buffalo.
```

verb (used with object), buf·fa·loed, buf·fa·lo·ing. *Informal.*

5. to puzzle or baffle; confuse; mystify: *He was buffaloed by the problem.*

6. to impress or intimidate by a display of power, importance, etc.: *The older boys buffaloed him.*



```
Enter filename: test.txt
 Flesh-Kincaid grade: 22.930000000000007
 Dale-Chall readability difficulty: 19.8233
```

≈Ph.D

$$Grade = C_0 + C_1\left(\frac{\text{num words}}{\text{num sentences}}\right) + C_2\left(\frac{\text{num syllables}}{\text{num words}}\right)$$

$C_0 = -15.59$ $C_1 = 0.39$ $C_2 = 11.8$

$$Difficulty = D_0\left(\frac{\text{num difficult words}}{\text{num words}} \times D_1\right) + D_2\left(\frac{\text{num words}}{\text{num sentences}}\right) + D_3\,bonus$$

$D_0 = 0.1579$ $D_1 = 100$ $D_2 = 0.0496$ $D_3 = 3.6365$

This assignment is all about writing methods to calculate each of the variables.

Forget everything you know about:

syllables
words
sentences
lines

Only the specific definitions in the handout matter

lines, tokens, syllables, and words

A <u>file</u> is comprised of <u>lines</u>.

```
I think, therefore I am.
I am (because I think).
I think I am, therefore I am. (I think?)
There is no period in this sentence
```

```
I think, therefore I am.
```

A <u>line</u> is comprised of <u>tokens</u>.

```
I think, therefore I am.
```

I ▯ think , ▯ therefore ▯ I ▯ am .

A <u>token</u> has a number of <u>syllables</u>.

```
think
```

```
1
```

A <u>token</u> *is sometimes* a <u>word</u>.

✓ I ▯ ✓ think , ▯ ✓ therefore ▯ ✓ I ▯ ✓ am .

How do you count the syllables in a word?

`syllablesInWord`

`private int syllablesInWord(String word)`

- What type of loop should we use to do this?

- Count the number of vowels in the word
  - *Except* for:
    - Vowels that have vowels directly before them
    - The letter e, if it appears by itself at the end of a word
- Words that have 0 vowels according to the rules above (ex. "me") are reported by `syllablesInWord` as having 1 vowel

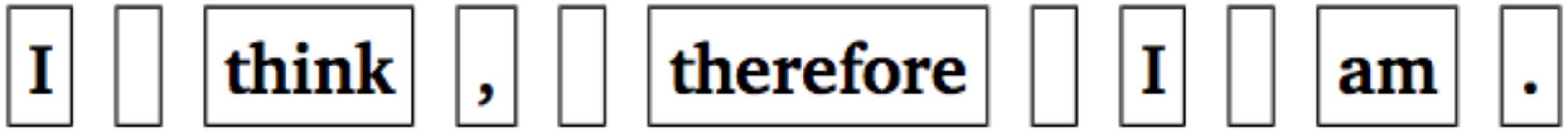
- <u>Style tip</u>: Write a helper method that determines whether a given character is a vowel (y counts!)

How do you turn a line into words?

`tokenize`

```
private ArrayList<String> tokenize(String input)
```

```
I think, therefore I am.
```



- A token is:
  - Any consecutive sequence of letters.
  - Any single character that isn't a letter.

- You'll need to understand how to:
  - convert characters to `String`
  - append onto a `String`
  - reset a `String` to empty

How do you store a group of tokens together?

`ArrayList`

ArrayList

```
import java.util.*;
```

~~`import acm.export.java.util.*;`~~

- You can append an element to an `ArrayList` by calling

  ***arrayList*.add(*value*)**

- You can get the *n*th element of an `ArrayList` by calling

  ***arrayList*.get(*n*)**

- You can see how many elements are in an `ArrayList` by calling

how to iterate through an ArrayList

this is super important

you're going to do it a lot on this assignment

this font is small

```
ArrayList<String> tokens =
                 new ArrayList<String>();

tokens.add("Bob");
tokens.add("Stanford");

for (String token : tokens) {
  println(token);
}
```

```
Bob
Stanford
```

So, in this assignment,
a "line" is an `ArrayList<String>`.

How do you process an entire line (an `ArrayList` of tokens)?

```
syllablesInLine
wordsInLine
sentencesInLine
```

```
private int syllablesInLine(ArrayList<String> tokens)
```

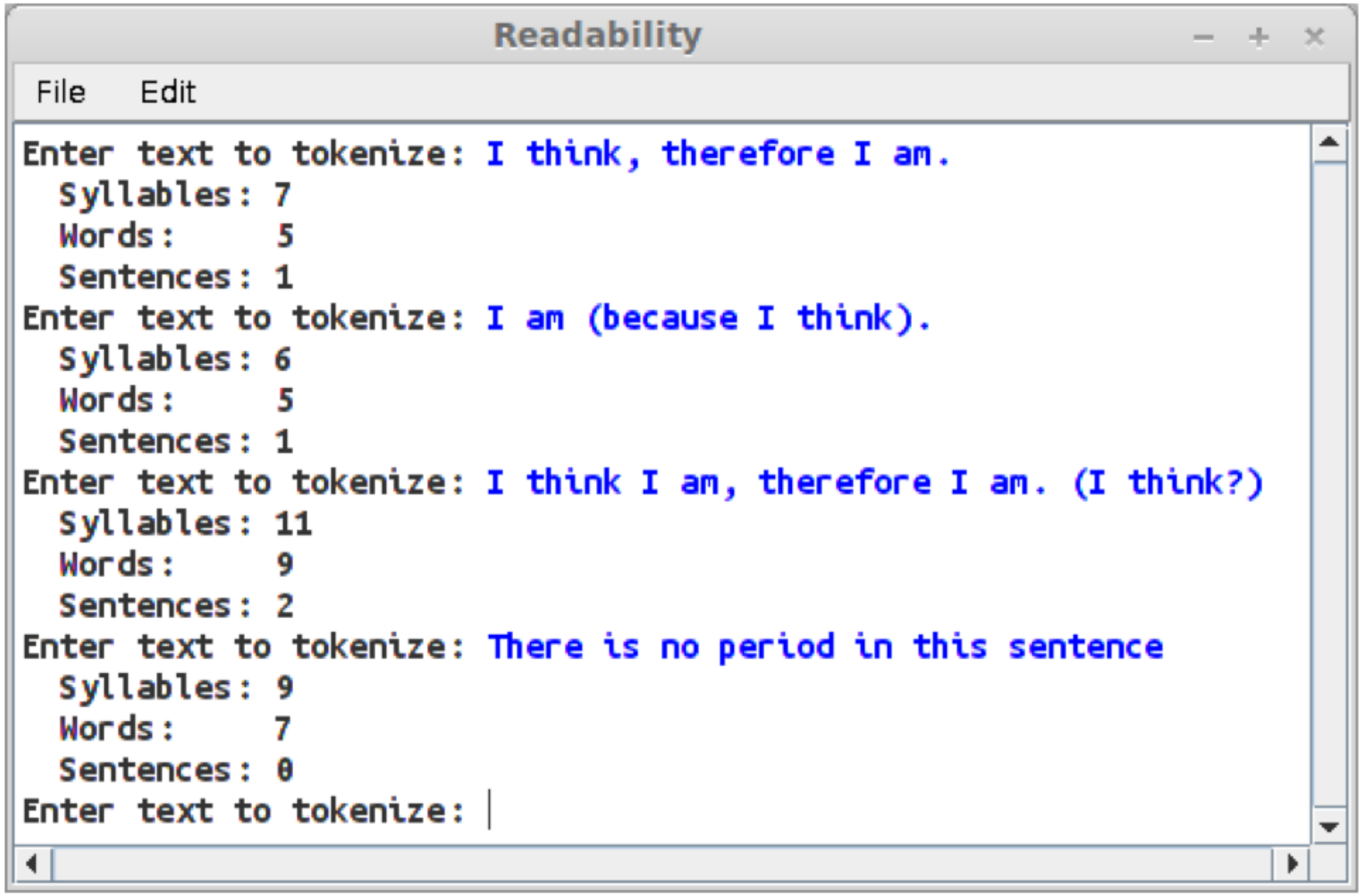
- Sum the value of `syllablesInWord` for each *word* in the line
- *word*: a token that starts with a letter

```
private int wordsInLine(ArrayList<String> tokens)
```

- Sum the number of *words* in the line
- *word*: a token that starts with a letter

```
private int sentencesInLine(ArrayList<String> tokens)
```

- Sum the number times that '.', '?', '!' appear in the line
    - helper method



```
Enter text to tokenize: I think, therefore I am.
  Syllables: 7
  Words:     5
  Sentences: 1
Enter text to tokenize: I am (because I think).
  Syllables: 6
  Words:     5
  Sentences: 1
Enter text to tokenize: I think I am, therefore I am. (I think?)
  Syllables: 11
  Words:     9
  Sentences: 2
Enter text to tokenize: There is no period in this sentence
  Syllables: 9
  Words:     7
  Sentences: 0
Enter text to tokenize: 
```

How do you get all the lines from a file?

`processFile`

```
private ArrayList<String> fileContents(String filename)
```

```
import java.io.*;
```

```
try {
    BufferedReader br = new BufferedReader(new FileReader(filename));
    while (true) {
        String line = br.readLine();
        if (line == null) break;

        /* … process line … */
    }
    br.close();
} catch (IOException e) {
    /* … handle error … */
}
```

Let's put it all together to calculate our first readability index:

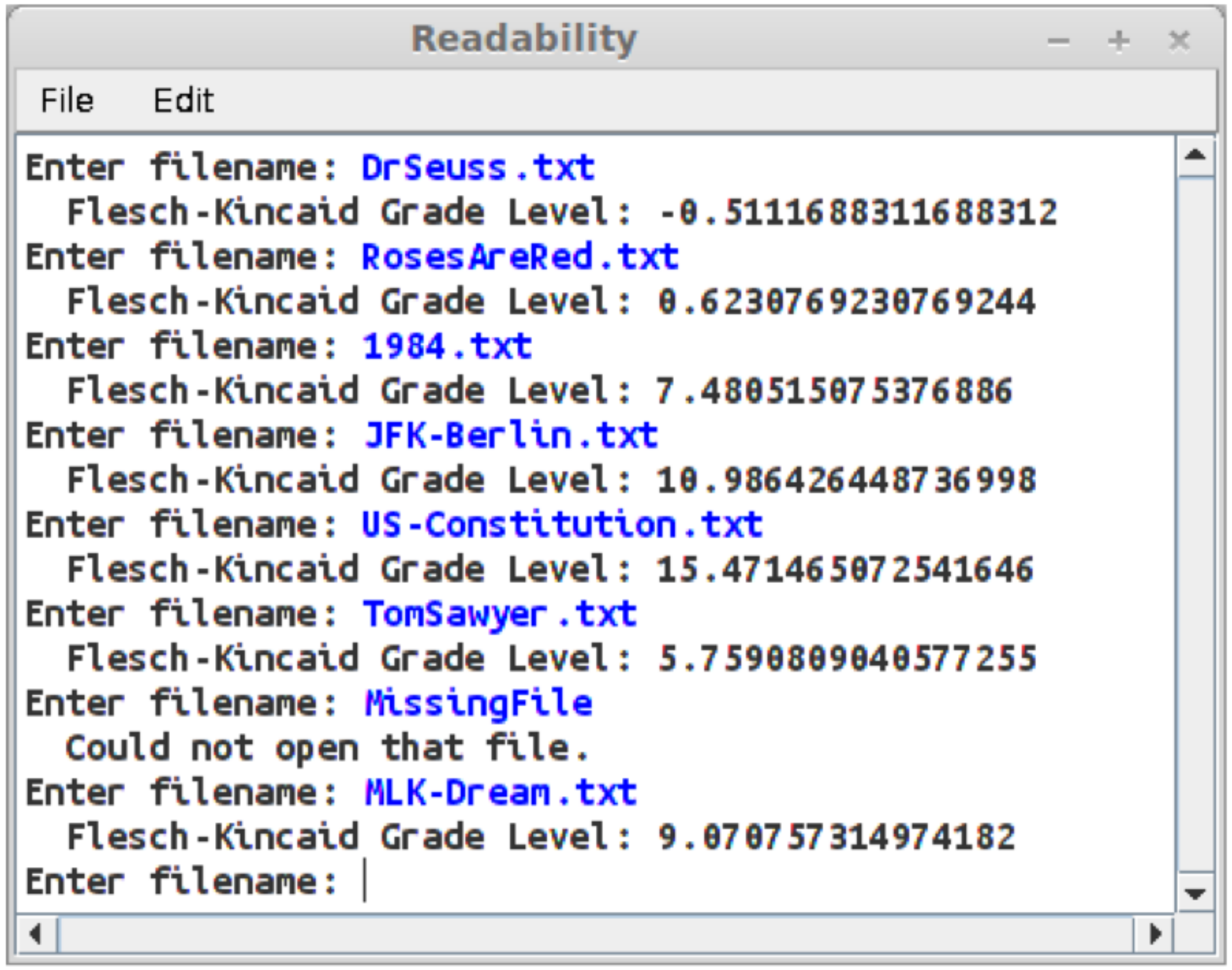
`fleschKincaidGradeLevelOf`

```
private double fleschKincaidGradeLevelOf(ArrayList<String> lines)
```

$$Grade = C_0 + C_1\left(\frac{num\ words}{num\ sentences}\right) + C_2\left(\frac{num\ syllables}{num\ words}\right)$$

$C_0 = -15.59$ $C_1 = 0.39$ $C_2 = 11.8$

- *num words* is 1 if the file has 0 words
- *num sentences* is 1 if the file has 0 sentences



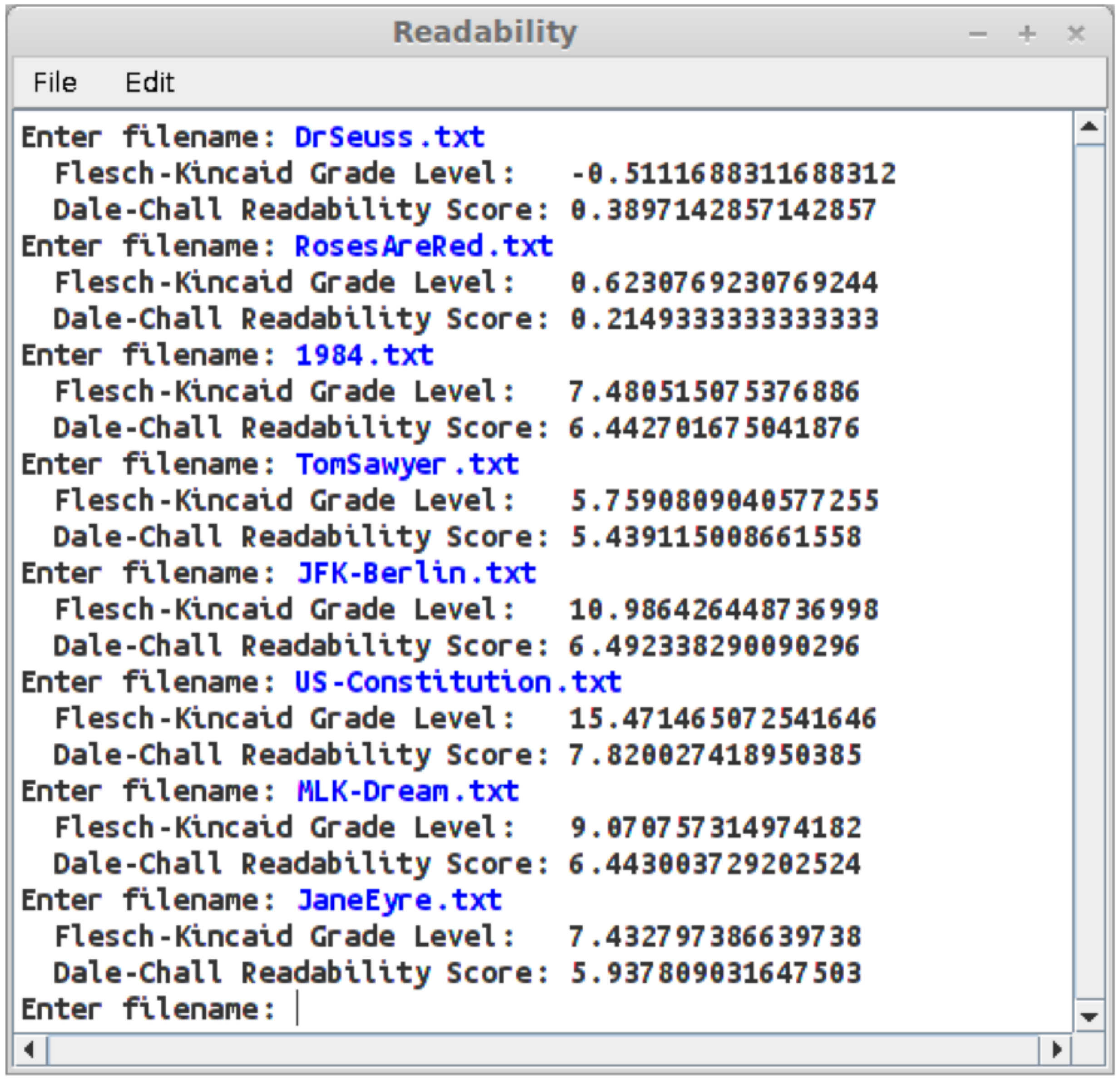
```
Enter filename: DrSeuss.txt
  Flesch-Kincaid Grade Level: -0.5111688311688312
Enter filename: RosesAreRed.txt
  Flesch-Kincaid Grade Level: 0.6230769230769244
Enter filename: 1984.txt
  Flesch-Kincaid Grade Level: 7.480515075376886
Enter filename: JFK-Berlin.txt
  Flesch-Kincaid Grade Level: 10.986426448736998
Enter filename: US-Constitution.txt
  Flesch-Kincaid Grade Level: 15.471465072541646
Enter filename: TomSawyer.txt
  Flesch-Kincaid Grade Level: 5.7590809040577255
Enter filename: MissingFile
  Could not open that file.
Enter filename: MLK-Dream.txt
  Flesch-Kincaid Grade Level: 9.070757314974182
Enter filename: 
```

`private double daleChallReadabilityScoreOf(ArrayList<String> lines)`

$$Difficulty = D_0\left(\frac{num\ difficult\ words}{num\ words} \times D_1\right) + D_2\left(\frac{num\ words}{num\ sentences}\right) + D_3\, bonus$$

$D_0 = 0.1579$ $D_1 = 100$ $D_2 = 0.0496$ $D_3 = 3.6365$

- *difficult word:* a word with ≥ 3 syllables
  - write a helper method
- *bonus:* 1 if ≥ 5% of words are difficult, 0 if not
- *num words* is 1 if the file has 0 words
- *num sentences* is 1 if the file has 0 sentences



```
Enter filename: DrSeuss.txt
  Flesch-Kincaid Grade Level:   -0.5111688311688312
  Dale-Chall Readability Score: 0.3897142857142857
Enter filename: RosesAreRed.txt
  Flesch-Kincaid Grade Level:   0.6230769230769244
  Dale-Chall Readability Score: 0.2149333333333333
Enter filename: 1984.txt
  Flesch-Kincaid Grade Level:   7.480515075376886
  Dale-Chall Readability Score: 6.442701675041876
Enter filename: TomSawyer.txt
  Flesch-Kincaid Grade Level:   5.759080904057725 5
  Dale-Chall Readability Score: 5.439115008661558
Enter filename: JFK-Berlin.txt
  Flesch-Kincaid Grade Level:   10.986426448736998
  Dale-Chall Readability Score: 6.492338290090296
Enter filename: US-Constitution.txt
  Flesch-Kincaid Grade Level:   15.471465072541646
  Dale-Chall Readability Score: 7.820027418950385
Enter filename: MLK-Dream.txt
  Flesch-Kincaid Grade Level:   9.070757314974182
  Dale-Chall Readability Score: 6.443003729202524
Enter filename: JaneEyre.txt
  Flesch-Kincaid Grade Level:   7.432797386639738
  Dale-Chall Readability Score: 5.937809031647503
Enter filename:
```

Let's determine the readability of websites!

```
Scraper.pageContents
```

```
lines = Scraper.pageContents(url);
```

- Process it as a URL instead of a filename if the user input begins with:
  - "http://"
  - "https://"
- How can we determine something is at the beginning of a `String`?

Searching a String

- You can search a string for a particular character or string by using the **`indexOf`** method:

***string*.indexOf(*pattern*)**

- **`indexOf`** returns the index of the first match if one exists.
- Otherwise, it returns -1 as a sentinel.

testing and debugging

Testing

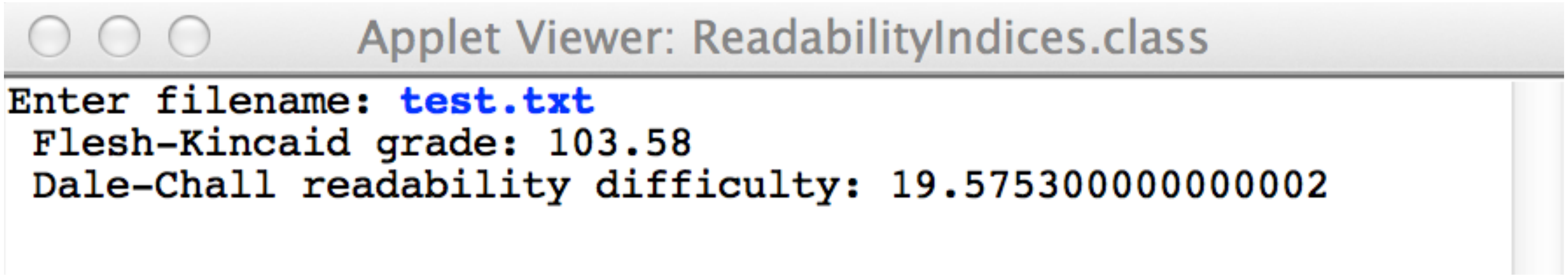
- How are you handling mIxEd cAsE words?
- Try creating your own test files
  - empty file
  - file with no sentences and one syllable
  - file with no syllables and one sentence
  - other ones?

Debugging

- Are you ever doing `integer` division where you should be using `double`?
- Which quantity is causing you to miss the target value?
  - Are you reporting too many words? Too few syllables?
- LaIR.
  - Seriously.
  - Syntax bugs are frustrating and that's one thing we're here to help you with.

test.txt

```
Antidisestablishmentarianism,
antidisestablishmentarianism,
antidisestablishmentarianism.
```



```
Enter filename: test.txt
 Flesh-Kincaid grade: 103.58
 Dale-Chall readability difficulty: 19.575300000000002
```

Score	Difficulty
0 – 5	Readable by an average 4th grader.
5 – 6	Readable by an average 5th or 6th grader.
6 – 7	Readable by an average 7th or 8th grader.
7 – 8	Readable by an average 9th or 10th grader.
8 – 9	Readable by an average 11th or 12th grader.
9 – 10	Readable by an average college student.
10+	Readable by an average college graduate.

- Follow the specifications carefully
- Comment
- Go to the LaIR if you get stuck
- **Incorporate IG feedback!**

- Have fun!