Project 01, Object-Oriented Programming, CSE 271, Spring 2020
Department of Computer Science and Engineering, Miami University

File Read/Write, String, Array, Try-Catch-Finally, Exception Handling

Learning Outcomes:
- Implement basic and nested control structures, single and multi-dimensional arrays, input and output methods, and string methods as learned in CSE 174
- Use class libraries to assist in problem solving
- Use exception handling features to handle exceptions.

Important Note: Submit code written by only you. Otherwise, you will receive 0 points and will be reported to the Academic Integrity Office for violation.

Problem:

Implement a class named **FileParser**, along with specified methods, that can read words from a text file, parse those words (such as finding palindromes), and write those words to a new file.

More specifically:

1. Write the **FileParser** class so that it has the following methods:
   a) public static void **main**(String[] args) -- The main driver of the program. Prompts the user for an input file, an output file, and a choice for what kinds of words to output.
   b) public static String **clean**(String word)-- Takes a string and removes all non-letters, returning an all uppercase version. For example, this input: "A man, a plan, a canal. Panama." will produce this output: "AMANAPLANACANALPANAMA"
   c) public static void **parse**(File in, File out)-- Parses an input file, writing each word from the input on a separate line in the output file.
   d) public static boolean **isPalindrome**(String word) -- Determines if a word is a palindrome (reads the same forward and backward). The method is case-sensitive, so it will say that dad and DAD and d-a-d are palindromes.
   e) public static void **parsePalindrome**(File in, File out) -- Parses an input file, writing only the palindromes to the output file, in alphabetical order, one line at a time, without repetition. Palindromes are words that read the same forward and backward, ignoring digits and punctuation.
   f) public static int **value**(String word)-- Returns the monetary value of a word, found by assigning the value $1 to A, $2 B, and so on, up to $26 for Z. The method will ignore differences in case, so both A and a are each worth $1. It will also ignore any non-letters in the input.
   g) public static void **parseHundredDollarWord** (File in, File out) -- Parses an input file, writing only the $100 words to the output file, in alphabetical order, one line at a time, in uppercase, without repetition. $100 words are found by assigning $1 to A, $2 to B, and so on. For example ELEPHANTS is a $100 word because: E + L + E + P + H + A + N + T + S is 5 + 12 + 5 + 16 + 8 + 1 + 14 + 20 + 19 = 100. You can use the *value()* method defined above to compute the value of a word.
2. Create your own input test file and use it with your code to generate three output files.
3. Upload all five files (Java source code, input file, three output files).

Preliminaries:

- Familiarize yourself with palindromes (words that read the same forward and backward), such as RACECAR. See, for example, the Wikipedia article on palindromes, https://en.wikipedia.org/wiki/Palindrome.

File Read/Write, String, Array, Try-Catch-Finally, Exception Handling

- Familiarize yourself with the concept of "$100 words", such as ELEPHANTS. See, for example, http://www.donaldsauter.com/dollar-words.htm

Grading Rubric:

	Full credit	**Partial credit**	**No credit**
File handlin g (40 pts)	Your program correctly performs all specified file handling (including checking for the existence of the input file, reading, and writing)	Your program performs the basic file I/O operations, but with some errors, or not according to specifications	Your file reading and/or writing causes program to crash
String parsing (40 pts)	Your program correctly parses words read from the file, including identifying palindromes, $100 words, and "cleaning" strings.	Your program performs the basic string parsing functionality, but with some errors, OR it performs all the basic functionality correctly, but not according to specifications	Your program can read in words and write out the same words but does not perform any of the string parsing that was specified.
Testing (20 pts)	You submitted an input file and 3 well named resulting output files that clearly demonstrate your program's ability to do all the required parsing.	You submitted the required test files, but the test files fail to demonstrate some aspect of your program (for example, the input file does not include adequate tests of your program to read from paragraphs and clean a variety of Strings). OR, your test files are named in a way that makes it difficult for others to know which output files correspond to which tests.	You did not submit all the required test files (one input file, and the three corresponding output files)
Javadoc Comments for the class and methods (only deduction – 5 points)			

Submit the following 5 files to the course website:

- FileParser.java
- An input file created by you that best demonstrates all the features of your program.
- Three output files corresponding to the three options being performed on your input file. The output files should be named with the .txt extension, using filenames that clearly indicate which of the three tests produced the file.

SUBMIT DRAFTS: If you reach a "milestone" (your code may not be complete, but it compiles and works at least partially), then UPLOAD YOUR DRAFT TO THE WEBSITE. When you reach the next milestone, archive your previous submission, and upload your newest draft. This is a good way to protect yourself. Plus, when someone says to me, "I missed the deadline for submitting this", I can say in reply, "I don't accept late submissions, but at least you have your latest draft submitted that I can grade."

File Read/Write, String, Array, Try-Catch-Finally, Exception Handling

Programming Notes:

1. Subtracting two char values will produce an int value. And, in fact, taking a letter and subtracting 'A' from it gives you a value that is *almost* what you want for the monetary value of that letter:

   'A' – 'A' is 0
   'B' – 'A' is 1
   'Z' – 'A' is 25
2. Arrays and ArrayLists are two reasonable ways to store the words you find. ArrayLists have a method built in that will tell you whether a particular value already exists in it. This can be useful when it comes to avoiding duplicating words. Only add a word to the list if it doesn't already exist in the list.
3. Java has a Collections class that has a static method that will sort an ArrayList of Strings. Feel free to use this, rather than implementing your own sorting algorithm. You can also use Arrays.sort() method to sort arrays.
4. Notice that only a little file handling happens in the main() method. main() is where the user enters an input filename, and then there is a check to see if that file exists. main() is also where the user enters the output filename. Once File objects are created from those filenames, those File objects are sent to other methods for the actual reading and writing.
5. Test your program with your own input files at first. Input files should be "pure text" files. Don't use a word processor to generate text files. Use a text editor instead.

Work incrementally!

You will not make it through this course if you can't master working incrementally. **DO NOT IGNORE ERRORS. DO NOT TAKE CODE THAT DOESN'T WORK, AND ADD MORE CODE TO IT.**

- Write a small amount of code, perhaps just part of a method, and test it.
  - Did it work? If so, repeat the process by adding on a little more code.
  - Did it fail? If so, then back up a little bit…remove something you are unsure of and get to something that DOES work. Then, gradually work your way through your difficulty.
- Stuck? Try to work through it on your own. You will become a better programmer as you become more independent. Your knee-jerk reaction should not be to go to someone else. You will not get better at programming if someone else tells you how to fix your problem.
- Still stuck? Ask for help!
- Is it possible to "put away" the part of your program that is not working, and work on other independent aspects? If so, go for it.
- **But whatever you do…if part X of your program doesn't work, and part Y depends on part X, then don't start on part Y. However, if part Z is completely independent of part X, then go ahead and write and test part Z instead.**

File Read/Write, String, Array, Try-Catch-Finally, Exception Handling

Sample runs:

Consider the following text file, demo.txt:

```
I found it interesting that the elephant's Radar2000 device was not working. A quarter of all
elephants get lost very easily, so something needs to be done about this situation.
An elephant's mom will be very upset if her son's Radar2000 device stops working.
```

Here are four sample runs of this program:

```
Enter input filename: dem.txt
File does not exist.  Goodbye.
```

```
Enter input filename: demo.txt
Found. What do you want to output?
1. Raw word list
2. Palindromes
3. $100 words
Choose: 1
Enter output filename: raw.txt
Finished printing raw word list.
```

```
Enter input filename: demo.txt
Found. What do you want to output?
1. Raw word list
2. Palindromes
3. $100 words
Choose: 2
Enter output filename: palindromes.txt
Finished printing palindromes.
```

```
Enter input filename: demo.txt
Found. What do you want to output?
1. Raw word list
2. Palindromes
3. $100 words
Choose: 3
Enter output filename: 100.txt
Finished printing $100 words.
```

Note regarding the example files on the next page (page 5):

- raw.txt file contains each word on a separate line, including punctuation
- palindromes.txt and 100.txt contain each palindrome and each $100 word, but the files do NOT include duplicates. So, even though "elephants" appears multiple times in the input file, it only appears once in the 100.txt file. Notice also that the word radar does not actually appear in the input file. Rather, radar2000 appears. But you will be evaluating words by first "cleaning" the words, eliminating any digits or punctuation before testing for palindromes or $100 words.

File Read/Write, String, Array, Try-Catch-Finally, Exception Handling

Here are the resulting files and their content (see previous page for sample runs and input file).

raw.txt	**palindromes.txt**	**100.txt**
I	A	ELEPHANTS
found	I	QUARTER
it	MOM	
interesting	RADAR	
that		
the		
elephant's		
Radar2000		
device		
was		
not		
working.		
A		
quarter		
of		
all		
elephants		
get		
lost		
very		
easily,		
so		
something		
needs		
to		
be		
done		
about		
this		
situation.		
An		
elephant's		
mom		
will		
be		
very		
upset		
if		
her		
son's		
Radar2000		
device		
stops		
working.		