Strings and File I/O

Strings

- Java String objects are *immutable*
- Common methods include:
  - boolean equalsIgnoreCase(String str)
  - String toLowerCase()
  - String substring(int offset, int endIndex)
  - String replace(char oldChar, char newChar)
  - int indexOf(String str)

File Input (Text)

- Option 1: Scanner/File
  - Scans input from given File
  - Input from file instead of System.in (keyboard)
  - Easy to scan ints, doubles, etc
- Option 2: BufferedReader/FileReader
  - BufferedReader allows you to read a line at a time instead of a character

File Input (Text)

- Option 1: Scanner

```
Scanner scan = new Scanner(new File("myfile.txt"));
String s;
while(scan.hasNext()) {//acts like an iterator
   s = scan.nextLine();
}
```

- Option 2: Buffered Reader

```
BufferedReader in = new BufferedReader(new
                              FileReader("myfile.txt"));
s = in.readLine(); //returns null when EOF reached
while(s != null) {
   s = in.readLine();
}
in.close(); //remember to CLOSE the file!
```

Tips

- Scanner must be imported from `java.util`
- File and Readers must be imported from `java.io`
- Must deal with `IOException`
  - Use try/catch for file operations or declare `throws IOException` in method header
  - `public static void main(String[] args) throws IOException { … }`

File Output (Text)

- FileWriter allows you to write to a file
- PrintWriter provides interface of System.out
- Remember to import correct packages and handle exceptions

```
PrintWriter out = new PrintWriter(new FileWriter("myfile.txt"));
out.println("String 1");
out.println("String 2");
out.close(); //remember to CLOSE the file!
```

Misc…

- Path names
  - Relative path name starts looking in current directory
    - Examples: “myfile.txt”, “mydirectory/myfile.txt”
  - Absolute path name starts from top-level directory
    - Examples “/home/srollins/cs112/myfile.txt”
      “C:\\srollins\\cs112\myfile.txt”
- Binary Files
  - FileInputStream/FileOutputStream read/write *bytes*

Exercises

1. Design and implement a program that prompts the user for a word and a file name and counts the number of times the given word appears in the file. Your program will keep a count of the number of lines in the input file and produce an output file that contains the line number and line for each line on which the word appears. For example, if your program was counting the number of occurrences of the word “stormy”, you might have the following entry in your output file to indicate that the word appears on line 5: “5: It was a dark and stormy night.” You can use the file dracula.txt to test your program -- search for the word “vampire”.