In this lab, you will learn to use

- Inheritance and composition
- Method overriding
- Abstract and concrete classes
- Handling text-files

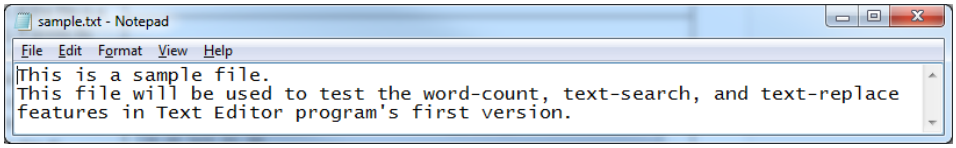

Figure 1: Regular doc

```
Enter file name
sample.txt
This file has 24 words
```

Figure 2: Regular doc's word count

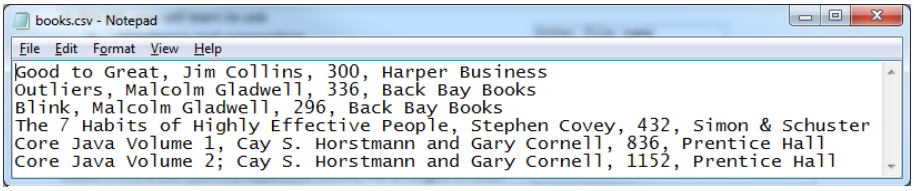

Figure 3: CSV doc

```
Enter file name
books.csv
books.csv has 6 rows and 4 columns
```

Figure 4: CSV doc's row and column count

Problem statement: Write a program that takes the name of a text file from the user. If the file is a regular text file (Fig.1), it prints its word count (Fig.2). If it is a CSV (comma separated values) file arranged in rows and columns (Fig.3), then it prints the number of rows and columns in it (Fig.4). Assume that the regular text file has whitespace as the word delimiter, and that the CSV file has all rows with equals number of columns (or commas)

Solution Design: The program has four classes as shown in Figure 5. DocAnalyst uses RegularDoc and CSVDoc that extend Document class. Document is an abstract class with one abstract method collectDocInfo(). Note that while the class diagram shows an Aggregation relationship, no member variables of RegularDoc or CSVDoc are shown in DocAnalyst. You may create them as member variables or just as local variables in main() method.

See Table 1 for detailed description of these classes.

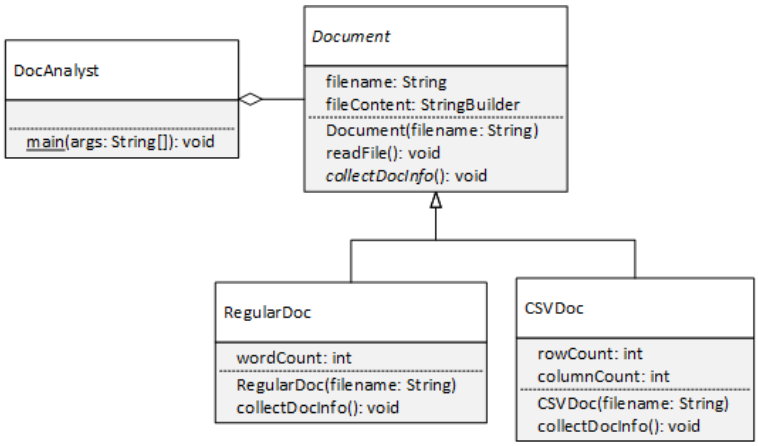

Figure 5: Class diagram

Instructions:

1. Download Document.java, TestDocs.java, sample.txt, books.csv from Canvas.
2. Create a package named **lab3** and import the two java files into this package.
3. Import the two data files - sample.txt, books.csv - into the Labs project folder
4. Create RegularDoc.java, CSVDoc.java, and then DocAnalyst.java with methods as shown in the class diagram above. Do not change any method signatures. Do not change Document.java.
5. Test your programs using TestDocs.java. Make sure your console output also comes out as expected.
6. Write your Andrew-id and name as top line comment in your RegularDoc.java, CSVDoc.java, and DocAnalyst.java. Zip them all into one folder named **<Andrew-id>lab3.zip**. Submit the zip file.

Rubric:

- 3 test cases: 4.5 points (1.5 points each)
- Console I/O: 4.5 points
- Submission: 1 point

Table 1: Class descriptions

Class	Method / variable	Description
Document	filename	This variable has the filename entered by the user. It is initialized in the Document's constructor.
	fileContent	This variable has content of the file read in the method readFile()
	Document()	Constructor that initializes the member variable filename
	readFile()	Opens the file with 'filename' and reads its contents into fileContent
	collectDocInfo()	Abstract method to be implemented by child classes
RegularDoc	wordCount	Stores the number of words in the file. Assume the whitespace is the word delimiter
	RegularDoc(filename)	Constructor that invokes the parent constructor
	collectDocInfo()	Uses the parent's fileContent to count the number of words. Stores this value in member variable wordCount
CSVDoc	rowCount	Stores the number of rows in the document. Assume that each row is separated by a newline character \n
	columnCount	Stores the number of columns in each row. Assume that each row has same number of columns, and each column is separated by a comma
	CSVDoc(filename)	Constructor that invokes the parent constructor
	collectDocInfo()	Uses the parent's fileContent to count the number of rows and columns. Stores these values in the member variables rowCount and columnCount
DocAnalyst	main()	Gets the program started, takes filename input from user, and depending on whether the file is a csv file (with extension .csv) or a regular file (any other extension), it instantiates the appropriate class. It then invokes the object's readFile() and collectDocInfo() methods. Finally it prints the word count or row/column count depending on the file type. *Hint: To find whether the filename entered by user is a .csv, one option is to use the String method .endsWith()*