Problem Statement

- Problem statement:

  Write an application that will build a word concordance of a document. The output from the application is an alphabetical list of all words in the given document and the number of times they occur in the document. The documents are a text file (contents of the file are an ASCII characters) and the output of the program is saved as an ASCII file also.

Overall Plan

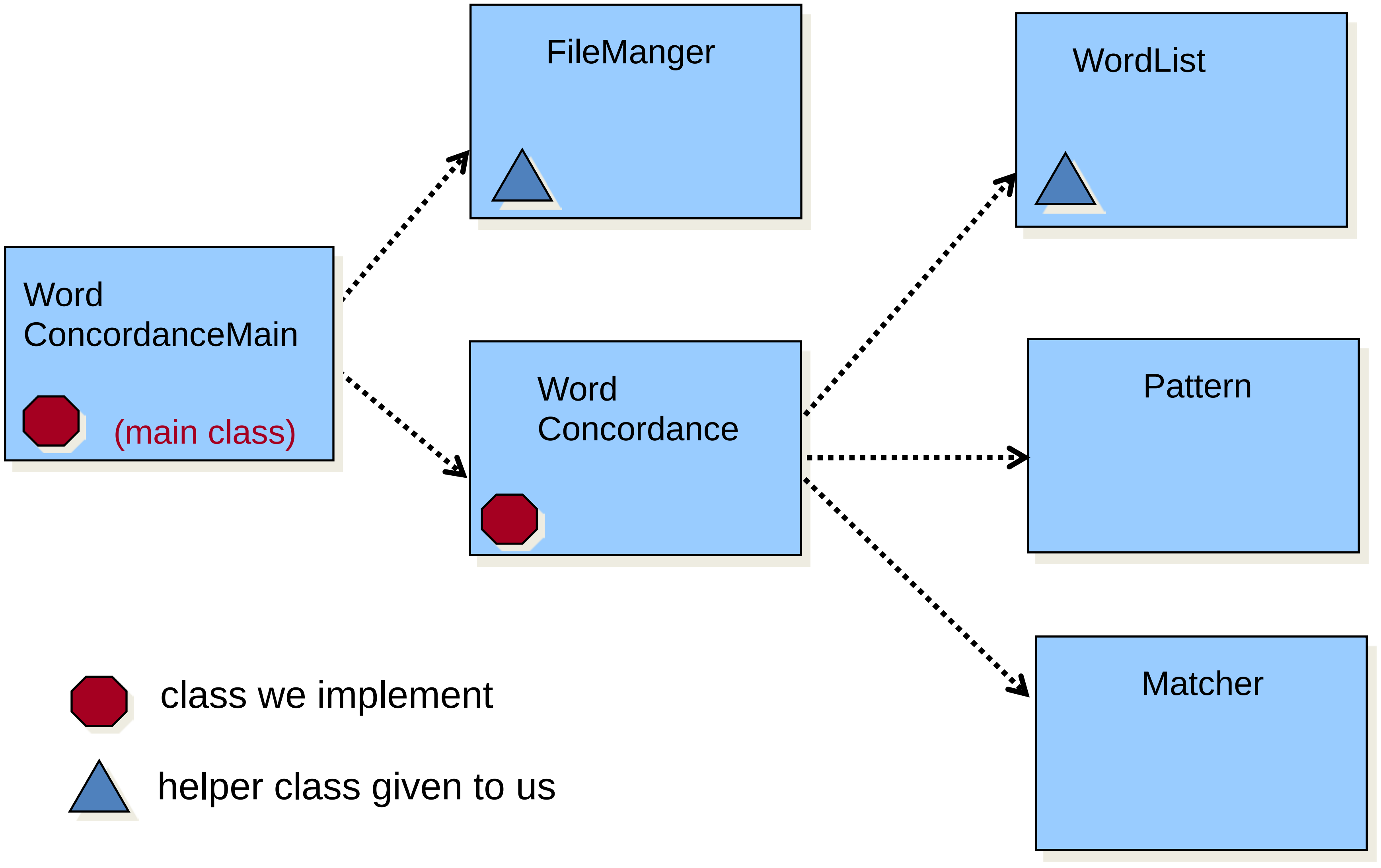
- Tasks expressed in pseudocode:

```
while ( the user wants to process
        another file    ) {

    Task 1: read the file;

    Task 2: build the word list;

    Task 3: save the word list to a file;
}
```

Design Document

Class	Purpose
WordConcordanceMain	The instantiable main class of the program that implements the top-level program control.
WordConcordance	The key class of the program. An instance of this class managers other objects to build the word list.
FileManager	A helper class for opening a file and saving the result to a file
WordList	Another helper class for maintaining a word list.
Pattern/Matcher	Classes for pattern matching operations.

Class Relationships



⬣ class we implement

▲ helper class given to us

Development Steps

- We will develop this program in four steps:

1. Start with a program skeleton. Define the main class with data members. Begin with a rudimentary WordConcordance class.
2. Add code to open a file and save the result. Extend the existing classes as necessary.
3. Complete the implemention of the WordConcordance class.
4. Finalize the code by removing temporary statements and tying up loose ends.

Step 1 Design

- Define the skeleton main class
- Define the skeleton WordConcordance class that has only an empty zero-argument constructor

Step 1 Code

Program source file is too big to list here. From now on, we ask you to view the source files using your Java IDE.

Directory: chapter07/step1

Source Files: WordConcordanceMain.java
WordConcordance.java

Step 1 Test

- The purpose of Step 1 testing is to verify that the constructor is executed correctly and the repetition control in the start method works as expected.

Step 2 Design

- Design and implement the code to open and save a file
- The actual tasks are done by the FileManager class, so our objective in this step is to find out the correct usage of the FileManager helper class.
- The FileManager class has two key methods: openFile and saveFile.

Step 2 Code

Directory: chapter07/step2

Source Files: WordConcordanceMain.java
WordConcordance.java

Step 2 Test

- The Step2 directory contains several sample input files. We will open them and verify the file contents are read correctly by checking the temporary echo print output to System.out.
- To verify the output routine, we save to the output (the temporary output created by the build method of WordConcordance) and verify its content.
- Since the output is a textfile, we can use any word processor or text editor to view its contents.

Step 3 Design

- Complete the build method of Ch9WordConcordance class.
- We will use the second helper class WordList here, so we need to find out the details of this helper class.
- The key method of the WordList class is the add method that inserts a given word into a word list.

Step 3 Code

Directory: chapter07/step3

Source Files: WordConcordanceMain.java
WordConcordance.java

Step 3 Test

- We run the program against varying types of input textfiles.
  - We can use a long document such as the term paper for the last term’s economy class (don’t forget to save it as a textfile before testing).
  - We should also use some specially created files for testing purposes. One file may contain one word repeated 7 times, for example. Another file may contain no words at all.

Step 4: Finalize

- Possible Extensions
  - One is an integrated user interface where the end user can view both the input document files and the output word list files.
  - Another is the generation of different types of list. In the sample development, we count the number of occurences of each word. Instead, we can generate a list of positions where each word appears in the document.