```
16              String word = parser.nextToken().toUpperCase();
17              String list = map.get(word);
18              if (list == null) map.put(word, "" + lineNumber);
19              else map.put(word, list + "," + lineNumber);
20            }
21            System.out.println(lineNumber + ":\t" + line);
22            line = in.readLine();
23          }
24          in.close();
25      } catch(IOException e) { System.out.println(e); }
26      System.out.println(map);
27      System.out.println("lines: " + lineNumber);
28      System.out.println("distinct words: " + map.size());
29    }
30
31    public static void main(String[] args) {
32      new Main(args[0]);
33    }
34  }
```

라인 32에서 main() 메소드는 명령어 라인 인자 args[0]을 클래스 생성자 Main()에게 전달한다. 이 인자는 주어진 텍스트 문서를 저장하고 있는 파일의 이름이다. 이 이름은 프로그램이 명령어 라인에서 실행 중일 때 실행-시간 시스템으로 다음과 같이 전달된다.

```
java Main Caesar.txt
```

또는 대안적으로 라인 32에 다음과 같이 직접 하드-코딩할 수도 있다.

```
32    new Main("Caesar.txt");
```

또는 라인 9 앞에 다음과 같이 지역 String 변수를 사용해 하드-코딩할 수도 있다.

```
9     String file = "Caesar.txt";
```

이들 중 첫 번째의 버전이 선호되는데, 그 이유는 프로그램을 다시 컴파일할 필요 없이 어떠한 텍스트 문서에 대해서도 실행시킬 수 있는 더 유연한 솔루션이기 때문이다.

Java에서 외부 텍스트 파일은 *입력 스트림(input stream)*이라고 하는 개개 문자들의 스트림으로 읽혀진다. 이러한 스트림들은 FileReader 객체에 의해 판독된다. 생성자

```
public FileReader(String file)
```

는 새로운 FileReader 객체를 생성하고 주어진 file 이름을 갖는 외부 텍스트 파일과 연결시킨다.