Analyzing Your Own Code

Error #1	
Meta Variable	**Value**
Checker	FB.DM_DEFAULT_ENCODING
File	/home/y66tang/jack/JWATIP465/src/pipair.java
Function	pipair.ParserAndReader(java.lang.String)
Ordered	true
Event	
Variable	**Value**
Main	True
Tag	Defect
Description	Found reliance on default encoding: new java.io.FileReader(String).
Line	65

Error#1 is detected by Coverity because when FileReader() is called with the default encoding (as shown on line 65: FileReader fileReader = new FileReader(fileLocation);) .

The constructor of FileReader is designed to use the platform default encoding if encoding is not specified, which is generally *a bad idea* since the default encoding depends on the system settings of the computer and is usually the most popular encoding among users in that locale. Therefore if a file were encoded in a different encoding, such way of reading file would not be able to successfully read in the correct content.

A possible fix for this bug is to use new InputStreamReader(new FileInputStream(filePath), encoding) and ideally get the encoding from metadata about the file.

Error #2	
Meta Variable	**Value**
Checker	FB.WMI_WRONG_MAP_ITERATOR
File	/home/y66tang/jack/JWATIP465/src/pipair.java
Function	pipair.PrintMissingPairsWithConfidence()
Ordered	true
Event	
Variable	**Value**
Main	True
Tag	Defect
Description	pipair.PrintMissingPairsWithConfidence() makes inefficient use of keySet iterator instead of entrySet iterator.
Line	210

Error #2 is identified by Coverity as a defect because it uses KeySet iterator instead of entrySet iterator. The use of KeySet does the following: retrieving all the keys (accessing the whole map), and then for some keys accessing the map again to get the desired value. A more efficient way as Coverity suggests is to iterate over the map to get map entries (Map.Entry) (couples of keys and values) while accessing the map only once.
Map.entrySet() delivers a set of Map.Entrys each one with the key and corresponding value.

Instead of the following code:

```
196         // Get the first function calls from the first method
197         HashSet<String> functionCalls = FUNCTION_MAP.get(firstKey);
198         for(String firstFunctionCall: functionCalls){
199
                        … …
210             int combinationCount = occurrenceCalls.get(secondCall);
                        … …
}
```

We could use:

```
196         // Get the first function calls from the first method
197         Set <Map.Entry<String, HashSet<String>>> functionCalls =
FUNCTION_MAP.entrySey();
198         for(Map.Entry<String, HashSet<String>>firstFunctionCall: functionCalls){
199
                        … …
210             int combinationCount = occurrenceCalls.get(secondCall);
                        … …
}
```