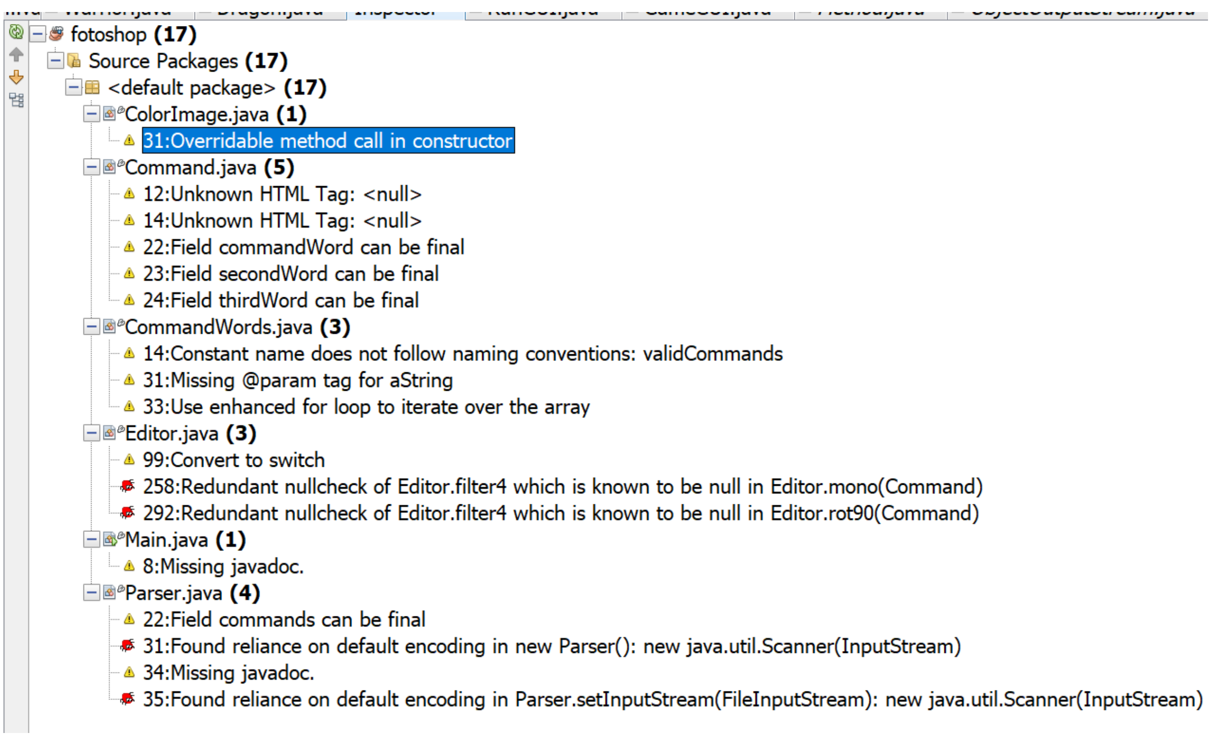
Critique:

- The CLI implementation (Front-End) and command functions (Back-End) were all implemented in one class (Editor). This demonstrates low cohesion increasing the module complexity and also makes it virtually impossible to extend the application to a different form of user interface without rewriting functionalities.
- Some classes contain fields which aren't set to private. Having non-private fields introduces the risk of improper field modifications. To promote low coupling, fields should only be processed within the class they were declared in. Accessors and Mutators can be integrated to access and modify these values as an object from a different class.
- There is a heavy usage of staircases of if-statements within procesCommand() in the Editor class to determine which command to execute depending on user input. Not only does it not take advantage of enums (to reduce overlooked typing mistakes) but this practice itself is not great because it shows that the class could be further modularised.
- Some command related methods have implementation mistakes. Specifically, rot90() inside the Editor class incorrectly adds 'flipH' as the filter. flipH is also available in the list of available commands despite the absence of its corresponding function.
- The output printed by the help command is quite insufficient as it only displays the list of commands and does not necessarily give instructions on how to properly use them. This may cause some slight confusion to the users.
- There are a few unhandled exceptions throughout the application, specifically within filter related commands (e.g. Applying the mono filter despite the absence of a loaded image will throw an exception). Unhandled exceptions are dangerous because it shows that the possibilities for user inputs has not been well thought out and has been left undealt with.
- The majority of supposedly back-end functions utilises System.out.println() which are CLI exclusive to print message outputs. This reduces the extendibility of the application to other interfaces such as GUI that doesn't support this type of function.
- Testing of individual functionalities is non-existent which makes the application unreliable.
- Some implementation approach could have been handled better. For example, a for loop and an array of Strings could have been used within the getCommand() inside Parser class to divide the user input into separate words (line 53). This would have reduced assignment repetition leading to a cleaner code. There is also no point in instantiating a command object upon entering an unknown command to return to the Editor class as none of the values set would have been used anyway (line 70).
- A static analysis report has been generated for this initial code using the findBugs tool to show any bad practice and usage of software development. The results are shown below.

Static Analysis Report for the initial version of the code – findBugs Tool



```
fotoshop (17)
  Source Packages (17)
    <default package> (17)
      ColorImage.java (1)
        31:Overridable method call in constructor
      Command.java (5)
        12:Unknown HTML Tag: <null>
        14:Unknown HTML Tag: <null>
        22:Field commandWord can be final
        23:Field secondWord can be final
        24:Field thirdWord can be final
      CommandWords.java (3)
        14:Constant name does not follow naming conventions: validCommands
        31:Missing @param tag for aString
        33:Use enhanced for loop to iterate over the array
      Editor.java (3)
        99:Convert to switch
        258:Redundant nullcheck of Editor.filter4 which is known to be null in Editor.mono(Command)
        292:Redundant nullcheck of Editor.filter4 which is known to be null in Editor.rot90(Command)
      Main.java (1)
        8:Missing javadoc.
      Parser.java (4)
        22:Field commands can be final
        31:Found reliance on default encoding in new Parser(): new java.util.Scanner(InputStream)
        34:Missing javadoc.
        35:Found reliance on default encoding in Parser.setInputStream(FileInputStream): new java.util.Scanner(InputStream)
```

ColorImage

- 31 Calling methods that can be overridden can be dangerous in the constructor because in the moment when the overridden method is called the object is not fully initialized.

Command

- 12,14 Error in Javadoc
- 22,23,24 Finds fields that can be made final, which can simplify synchronization and clarity

CommandWords

- 14 Checks that constant names follow the prescribed naming conventions. A constant is field which is static and final. The naming convention is defined using a regular expression, minimum and maximum length of the identifier. If a length is set to 0, the constraint does not apply at all. Set the match expression to empty string to disable regular expression matches. If Check only immutable types is checked, only primitives and immutable values are checked. Immutable values include null, zero-sized arrays (of any type), enumeration values and certain predefined JDK classes, such as java.awt.Color and the like. You may extend the list of types treated as immutable values with your own classes.

Editor

- 99 Marks cascades of ifs which can be converted to switch over Strings.
- 258,292 Redundant null check of value known to be null. This method contains a redundant check of a known null value against the constant null.

Parser

- 22 Finds fields that can be made final, which can simplify synchronization and clarity
- 31,35 Reliance on default encoding. Found a call to a method which will perform a byte to String (or String to byte) conversion and will assume that the default platform encoding is suitable. This will cause the application behaviour to vary between platforms. Use an alternative API and specify a charset name or Charset object explicitly.