Improvements:

- Command pattern is integrated to segregate the front-end from the back-end. This
 design pattern was chosen because of the nature of the application (command reliant).
 There are a few advantages in using this design pattern:
 - Greatly decouples the application as each individual command will have their own classes (Invoker).
 - Easier to add new commands as it doesn't require a lot of changing existing code.
 - Creates a structure for the entire application. It's easier to determine which commands are available based on the files contained within the project.
 - o Removes the staircases of if-statements in the front-end class Editor.
- Singleton pattern is also integrated to create a global shared resource within the
 application (called Resources) which can be used by front-end and back-end classes. This
 design pattern was chosen so the invoker classes can work with the same image being
 operated on.
- There are new commands that has been implemented based on the user requirements in the specification:
 - A 'put' command has been implemented to store copies of modified images in the image cache.
 - A 'get' command has been implemented to retrieve an image from the image cache.
 - An 'undo' command has been implemented to remove the recently added filter from the image.
- On top of these, there are also new commands to demonstrate the extendibility of the application:
 - A 'flipH' command filter has been implemented which flips an image horizontally.
 - o A 'flipV' command filter has been implemented which flips an image vertically.
 - A 'cache' command has been implemented to show the list of images currently stored in the image cache.
- There are also a few improvements on existing commands:
 - The 'script' command can now use the third word from the user input which refers to the file location of the script file.
 - The 'help' command will now give a detailed instruction on how to use each command.
- As requested in the specification, the entire application has now been internationalised which makes localisation possible. Language files are placed within the properties files which are referenced throughout the entire code. (Current languages are UK English and French)
- Unhandled exceptions have now also been appropriately handled. This includes the exception thrown by applying filters on a non-existing image. Multiple user input possibilities have been thought out and dealt with to avoid run time errors.

- CLI exclusive print lines has now been removed within each command implementation.
 Instead message outputs are returned, and changes done to the image are achieved using the Resources class. This makes GUI extendibility possible.
- Better practices of programming have been used such as using arrays when applicable
 and avoiding code repetition using for loops if possible. Fields that didn't have access
 modifiers are now set to private. Classes and methods are encapsulated to increase
 code cohesion. Unrelated classes were also segregated to reduce coupling.
- Packages are used to appropriately separate source files. (e.g. command invokers have their own package called 'commands')
- Thorough documentation has been done throughout the entire code. This includes both source files and test files for better understandability. **The full generated Javadoc can be found in /javadocs/javadoc/index.**
- Unit Tests has been integrated to the project to increase the reliability of the application. Tests includes message outputs, localisation and image outputs (ensuring that the filter has been applied correctly)
 - Code coverage was also used to ensure the entire code has been tested. The full results of the code coverage can be found in /.jacocoverage/report.html/index.
- A System Analysis report has been generated to reveal any bad software development practices. The results of the full report are shown below this document.
- Git-hub has been used for version control and team collaboration purposes. The list of commits is also shown below this document.

Bugs and Issues:

- The usage of Singleton pattern in the Resources class makes it difficult to perform unit testing as each test cases should be independent. To fix this, a method to reset the resources were made which is executed before each test runs. This approach however isn't ideal as picked up by the static analysis report.
- Code coverage wasn't completely 100%
 - In the 'ScriptCommand' class, there is an IOExcepotion handler which is triggered by a BufferedReader. This wasn't tested because the requirements to trigger this exception are difficult to reproduce in the test environment. They are caused by external factors that cannot be easily simulated e.g. the sudden deletion of a source file being used in the test.

Limitations:

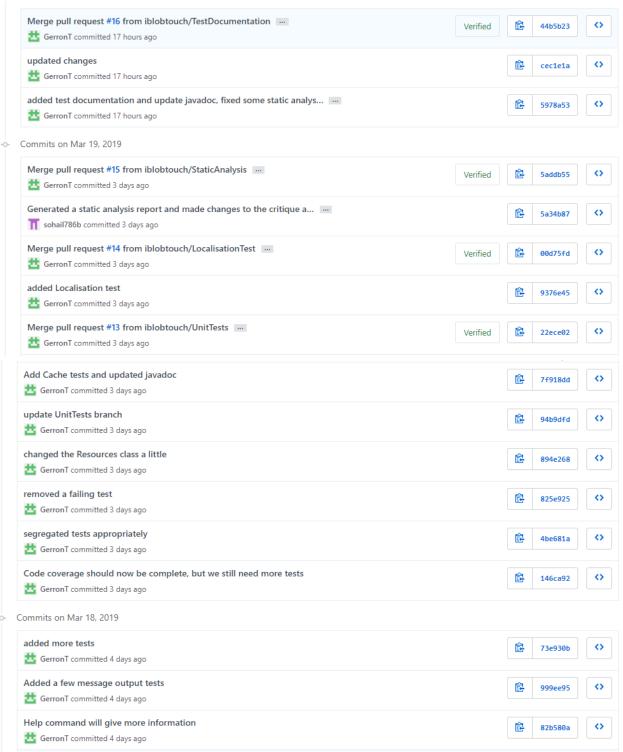
Testing that the image output contains the correct filter applied to it are difficult to
achieve especially for filters which changes the colour value of each pixel (mono). This is
because the actual image output is not exactly the same as its version while it was still
being processed by the application. The saving process may result in resolution changes
thus changing their pixel colour value. More on this in TestUtility.java file.

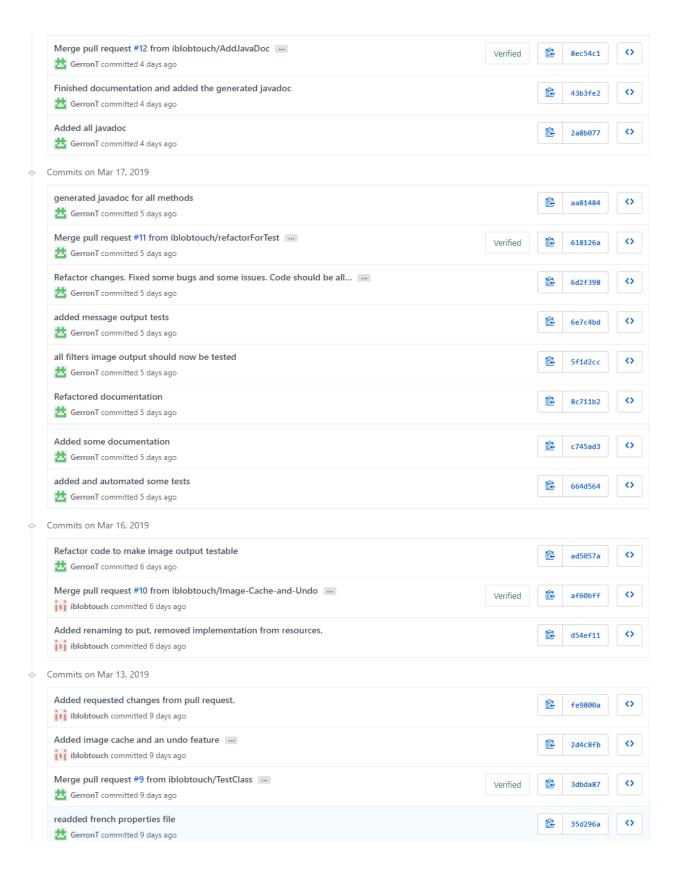
Future Improvements:

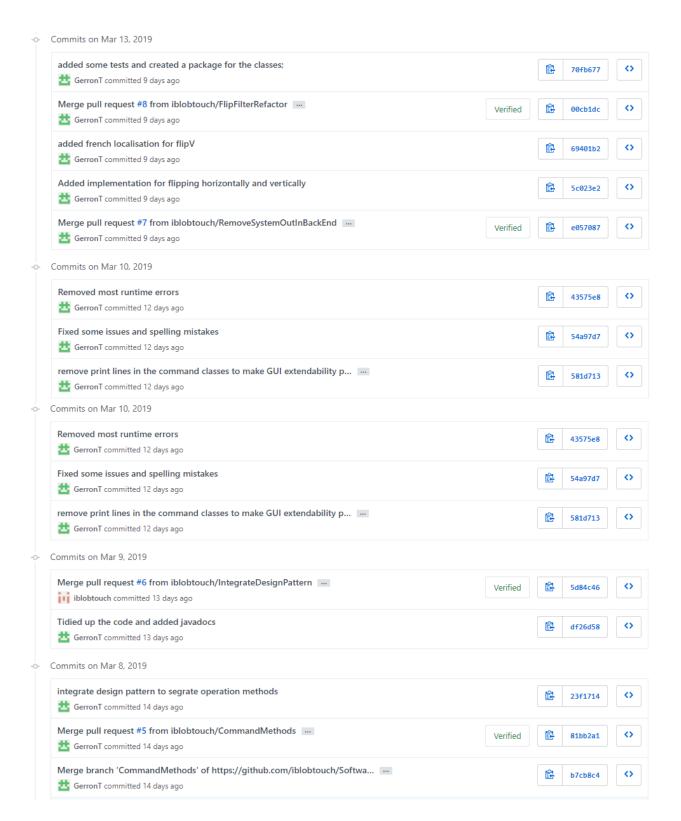
• In the future, it would be ideal to be able to increase the maximum of filters that can be applied on an image by changing just one parameter. Currently, changes in multiple files needs to be made to attain this.

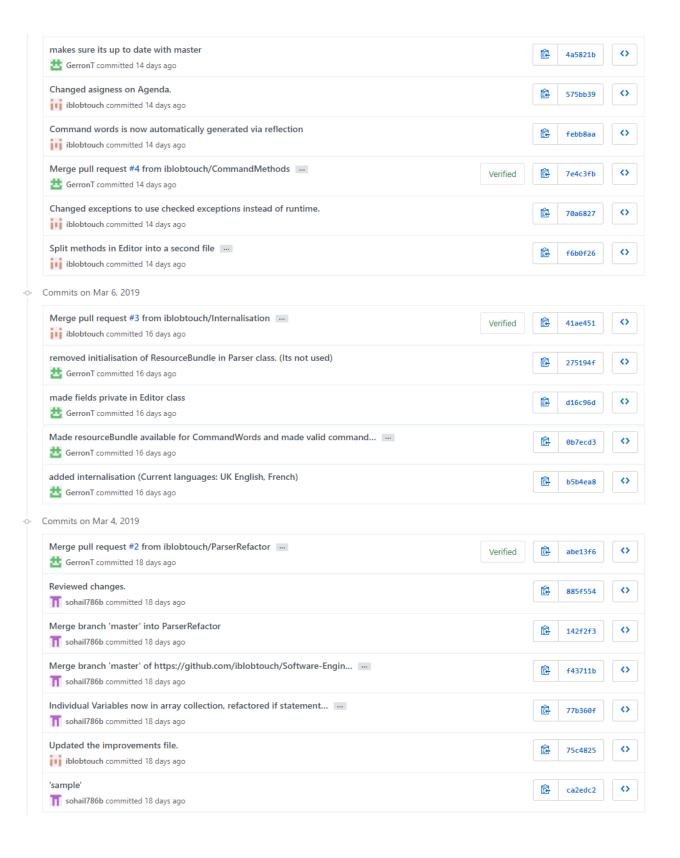
Git-hub Commits

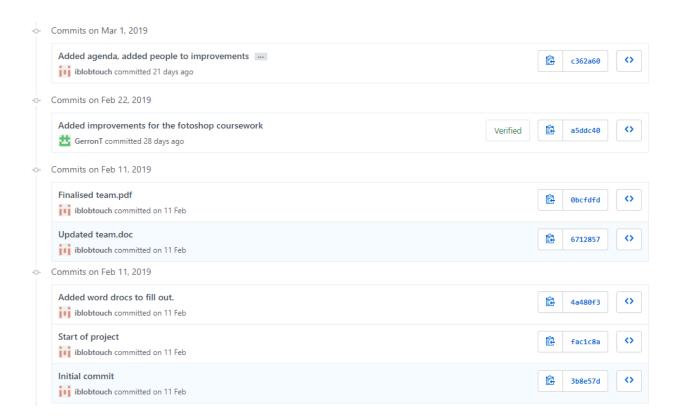
- Commits on Mar 21, 2019



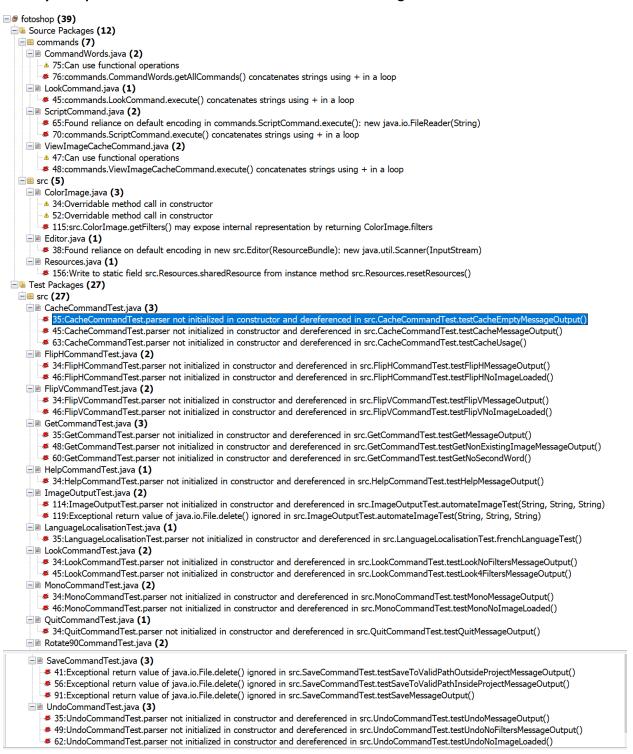








Static Analysis Report for the refactored version of the code - findBugs Tool



Below shows further description given by findBugs found by the static analysis report. Each reference in each class in the table is listed in order they appear in the report above.

- 1. Use functional operations instead of imperative style loop.
- 2. The method seems to be building a String using concatenation in a loop. In each iteration, the String is converted to a StringBuffer/StringBuilder, appended to, and converted back to a String. This can lead to a cost quadratic in the number of iterations, as the growing string is recopied in each iteration. Better performance can be obtained by using a StringBuffer (or StringBuilder in Java 1.5) explicitly
- 3. 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.
- 4. Calling methods that can be overridden can be dangerous in the contructor because in the moment when the overridden method is called the object is not fully initialized.
- 5. Returning a reference to a mutable object value stored in one of the object's fields exposes the internal representation of the object. If instances are accessed by untrusted code, and unchecked changes to the mutable object would compromise security or other important properties, you will need to do something different. Returning a new copy of the object is better approach in many situations.
- 6. This instance method writes to a static field. This is tricky to get correct if multiple instances are being manipulated, and generally bad practice.
- 7. This field is never initialized within any constructor and is therefore could be null after the object is constructed. Elsewhere, it is loaded and dereferenced without a null check. This could be a either an error or a questionable design, since it means a null pointer exception will be generated if that field is dereferenced before being initialized.
- 8. This method returns a value that is not checked. The return value should be checked since it can indicate an unusual or unexpected function execution. For example, the File.delete() method returns false if the file could not be successfully deleted (rather than throwing an Exception). If you don't check the result, you won't notice if the method invocation signals unexpected behavior by returning an atypical return value.

src.commands.CommandWords – 1, 2	src.commands.LookCommand – 2
src.commands.ScriptCommand – 2, 3	src.commands.ViewCommand – 1, 2
src.src.ColorImage – 4, 5	src.src.Editor – 3
src.src.Resources – 6	test.src.CacheCommandTest – 7

test.src.FlipHCommandTest – 7	test.src.FlipVCommandTest – 7
test.src.GetCommandTest - 7	test.src.HelpCommandTest – 7
test.src.ImageOutputTest – 7, 8	test.src.LanguageLoclisationTest – 7
test.src.LookCommandTest – 7	test.src.MonoCommandTest – 7
test.src.QuitCommandTest - 7	test.src.Rotate90CommandTest – 7
test.src.SaveCommandTest - 8	test.src.UndoCommandTest – 7