Friendly Image Processing Tool

Work Done

We had good progress in the first week of the project development. The current progress is around 65% and the work done are,

- Duy Duong
    - Design UI with FXML controllers and FXML styles
    - Handle event and update for UI display using Command Pattern
    - Implement Singleton Pattern for main logic classes (Generator, Analyzer, Collector)
    - Implement Observer Pattern for tracking Progress of collecting Pictures from URLs
- Ahmed Biby
    - Implement Factory Pattern for Picture class and subclasses
    - Build Util class to handle downloading, saving, generating, and processing for Pictures
    - Construct Strategy Pattern for Image Processor class with ObjectDetectionAlgorithm classes

There were some issues that we encountered along with solutions we came up with,,

- Setting up a Gradle project with extensions of JavaFX and OpenCV is a pain! We have to find the right versions for all three to be compatible with each other. Also, the OpenCV library folder is more than 1Gb which takes a long time to download and install..
- Unable to handle API single response of multiple images due to a variety of response structures. We switch over using one request - one image approach but allow users to input multiple URLs at once, separated by a next line symbol.
- Loss of Picture attributes if only save BufferedImage as .PNG. We extend Picture class with Serializable to serialize the whole Picture Object as .txt which preserves all Picture attributes.
- Java Observable doesn't trigger an update for the Progress UI view. We use JavaFX Property Bindings instead which includes ObjectProperty (observable) and PropertyListener (observer).

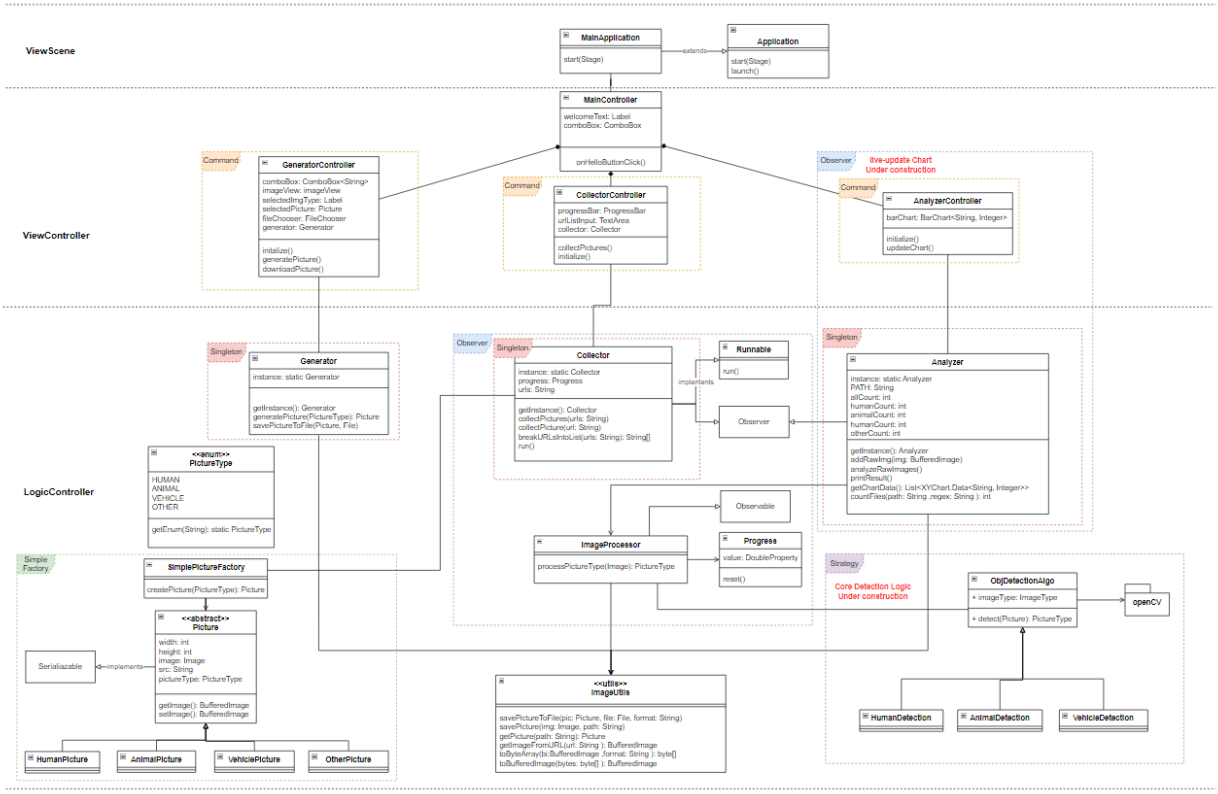
There are some use of design patterns existing in our system,

- Command Pattern is used to trigger event actions after an UI event occurs such as drop down selected and button clicked (using JavaFX Event and Action objects).
- Singleton Pattern is used for main classes (Generator, Analyzer, Collector). They act as services that can be delegated instead of coupling.
- Observer Pattern is used to track Progress value of collecting Pictures from URLs and constantly update ProgressBar on the UI.

- Factory Pattern is used to create Picture objects to store rawImage from URL response along with PictureType which is the result of the picture analysis.
- Strategy Pattern is used to perform needed operations for object detection in the picture analysis process.

Class Diagram

Here is the link to the Updated Class diagram. There are highlights in red which are codes under construction.



Plan for Next Iteration

There are two remaining complicated tasks which are making a live-update Chart for our gallery and finishing object detection algorithms using the OpenCV library. So the plan will be,

- Duy Duong
    - Implement Observer Pattern for live Chart update when analyzing rawPictures
    - Beautify the UI
    - Unit Testing for controllers
- Ahmed Biby
    - Complete core logics for ObjectDetectionAlgorithm classes
    - Unit Testing for Utils and core logics