Friendly Image Processing Tool

5448 Semester Project

State of System

Our system is at the complete state (100%). Main features are all implemented which are,

- Generate pictures by picture type and allow user to download generated picture
- Collect images from user input list of URLs
- Analyze raw images collected from URL, detect picture type using object detection algorithms, and create analysis charts.

Class Diagram and Comparison

There are some key relationships in our final set diagram,

- Our system has three layers: UI view, view controllers, and logic controllers. UI Views are made using FXML. View controllers handle the UI update and request from UI to the logic controller. Logic controllers handle most of the logic operations.
- Generator has some generating methods for 4 Picture Types, Animal, Human, Vehicle, Other. Each method will select a random picture from the corresponding folder and return to the frontend. But, there is also an option Random. When the user selects Random, we will select a random picture from the whole gallery instead. Users can download the picture after it is fully generated and displayed.
- Collector reads url string and reads raw image from each URL provided. Collector has a Progress class whose value is observable. When collecting images, show the progress bar which is a listener to the Progress value.
- Analyzer has an ImageProcessor which uses OpenCV library Machine Learning Algorithms to detect the Picture type of all raw images collected from URLs. It then serializes them to maintain Picture attributes and stores them into the corresponding folders then updates the counts of the number of pictures for each Picture Type and all pictures then make a chart of the counting result then display to the frontend. Analyzer will show the latest update of picture analytics.
- ImageUtils provides utility for handling image data such as read, write, and convert from one image object to another.

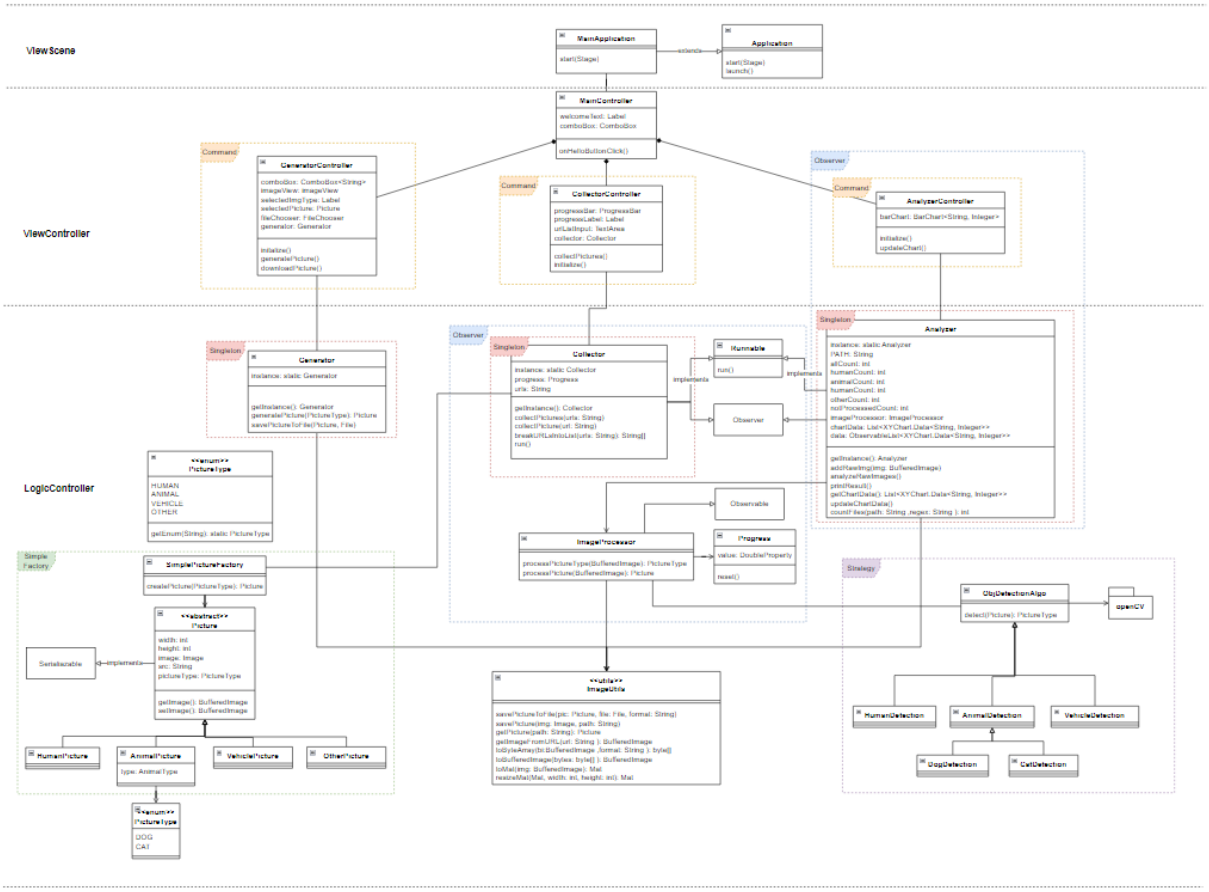
We maintain our design patterns throughout all three iterations,

- 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.

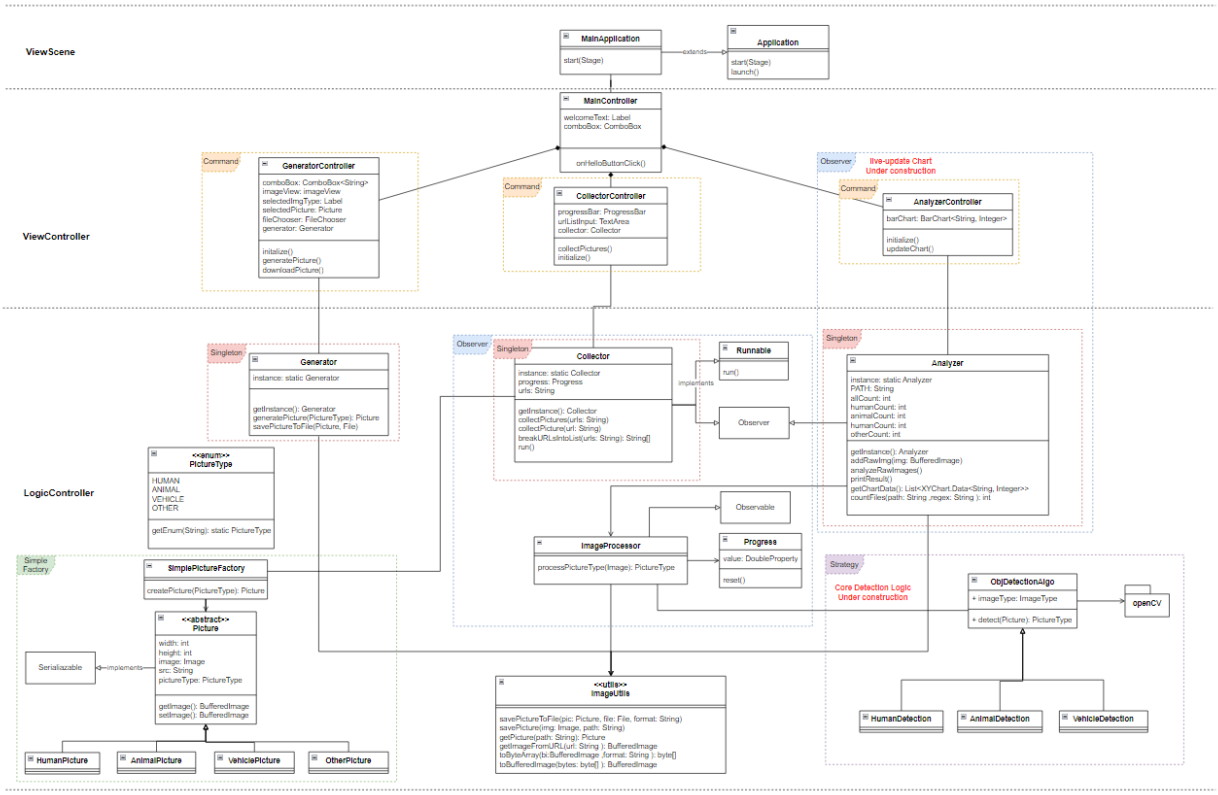
- Factory Pattern is used to create Picture objects to store rawImage from URL response along with PictureType which is the result of the picture analysis.
- Strategy Pattern is used to perform needed operations for object detection in the picture analysis process.

Next, we have some key changes for each iteration of our project. They are as follows,

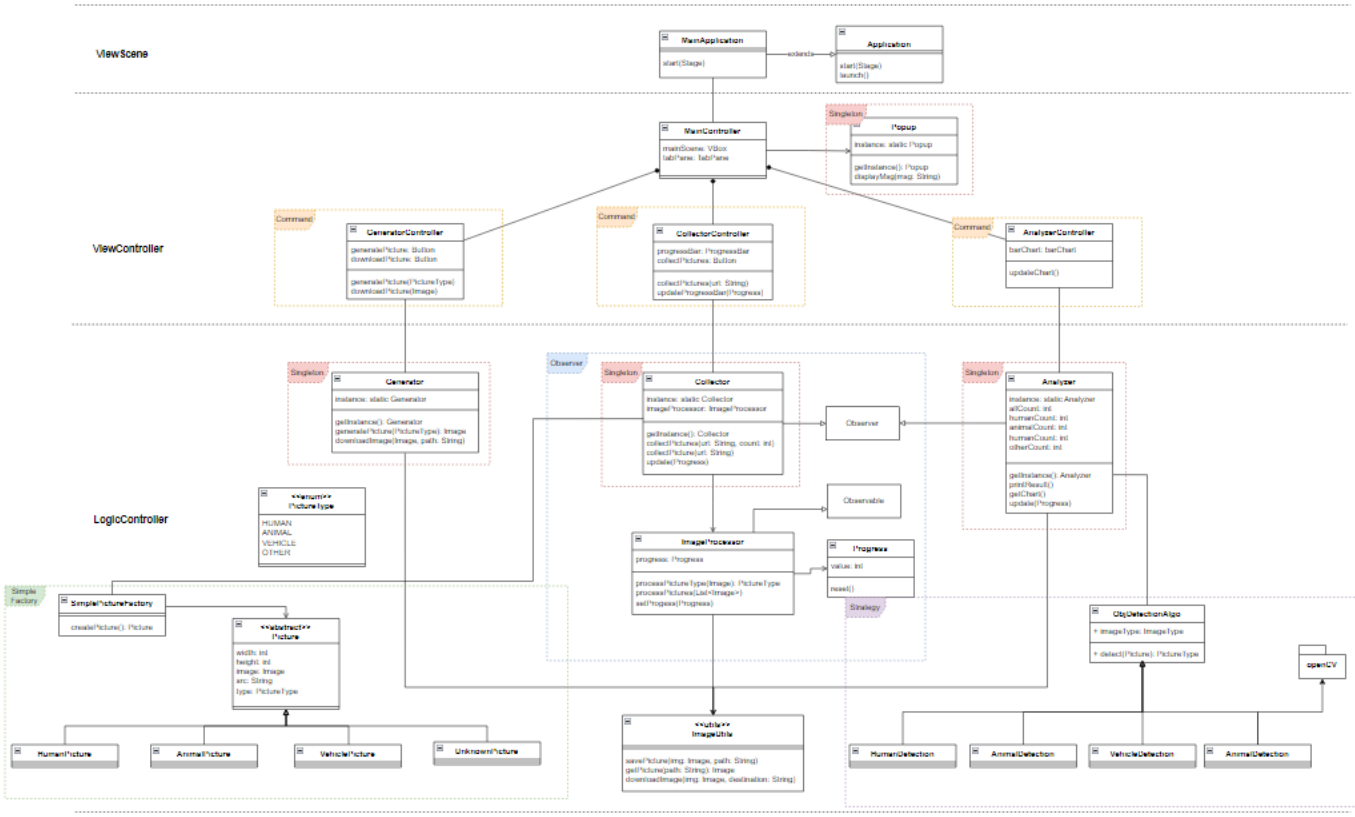
Project 7, due to limitations of trained data, Animal detection algorithms can only detect dogs and cats. We added an AnimalType enum for AnimalPicture to classify Dog or Cat. We also added specific algorithm classes, CatDetection and DogDetection as subclasses of AnimalDetection. Also, we shift the responsibility of creating Picture after object detection analysis from Analyzer to ImageProcessor since it's part of the image processing. This helps reduce workload for Analyzer and also make ImageProcessor more comprehensive.



Project 6, we were 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. Next, We ran into the issue of losing Picture attributes if we 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. To resolve this, we use JavaFX Property Bindings instead which includes ObjectProperty (observable) and PropertyListener (observer).



Project 5



Here are links to all three diagrams

1. Project 7
2. Project 6
3. Project 5

Third-party code vs. Original code

We used some third-party code examples as reference to enhance our Image util class for easier image data conversion,

1. Read Image from URL as BufferedImage
   http://www.java2s.com/example/java-utility-method/bufferedimage-from-url-index-0.html
2. Convert BufferedImage to byte array,
   https://mkyong.com/java/how-to-convert-bufferedimage-to-byte-in-java/
3. Convert byte array to BufferedImage,
   https://mkyong.com/java/how-to-convert-bufferedimage-to-byte-in-java/
4. Convert BufferedImage to Mat,
   https://answers.opencv.org/question/28348/converting-bufferedimage-to-mat-in-java/

OOAD Process

There are three main key design process elements that we experienced in this project.

1. Define system requirements at high-level, at software level, and at development level. This step truly is worth it. It allows us to see our project at different depths which help with encapsulation design.
2. Analyze components' responsibilities and relationships. This step allows us to balance the responsibilities and workload for each component. This also helps us simplify dependencies, reduce coupling, and increase cohesiveness.
3. Implement current system design and modify requirements for un-feasible features for exchange or future iteration. By putting our hand on implementation early, we can have a better understanding of how our system should operate. This suggests modification for better implementation and extensibility of our project each iteration.

Demonstration

We have a brief demo recording which can be accessed here

- https://cuboulder.zoom.us/rec/share/t4x9VxquXl1OT-2usgbYkUAv8uAVbwmzROvRZm3fkEhsHYnxQslo3V4fGWJhNlKU.ojDzw6_IeAQdBM-S?startTime=1682874709000
- Passcode: H55$K=ei