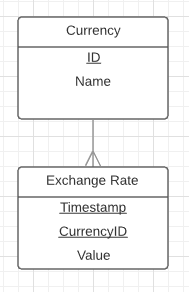
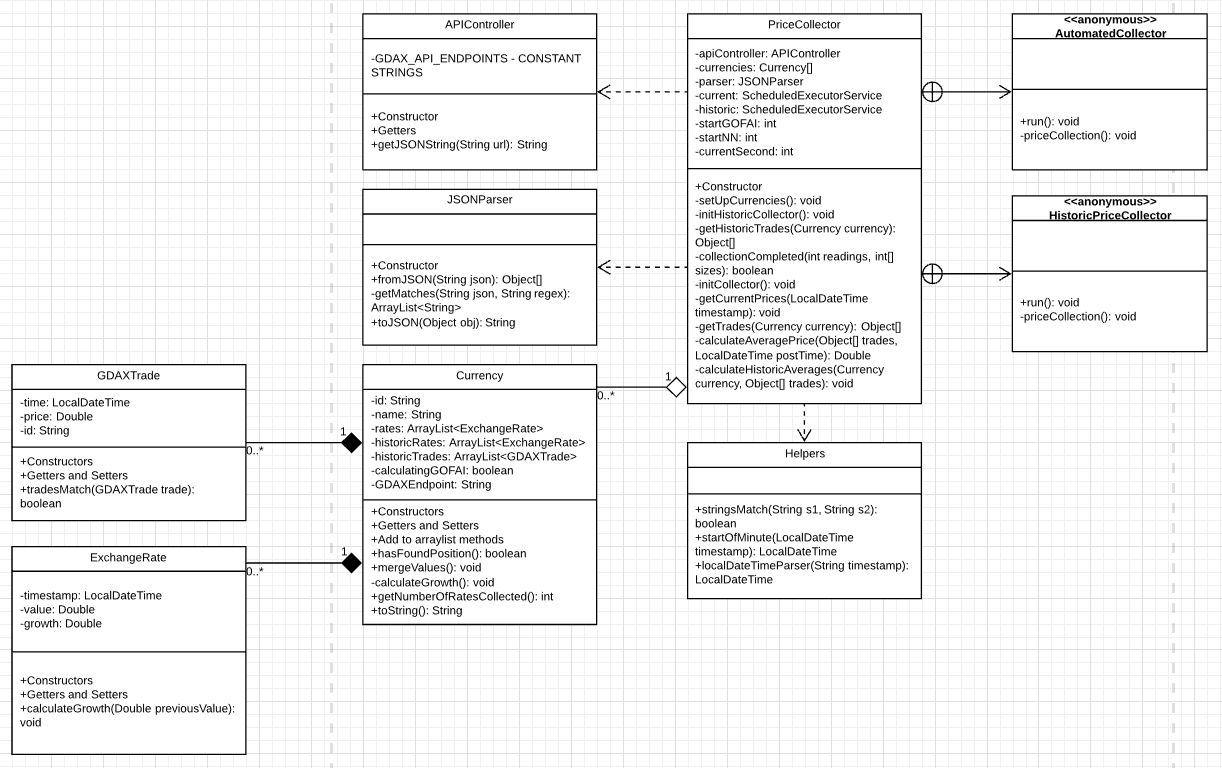
# System Design

## Entity Relationship Diagram



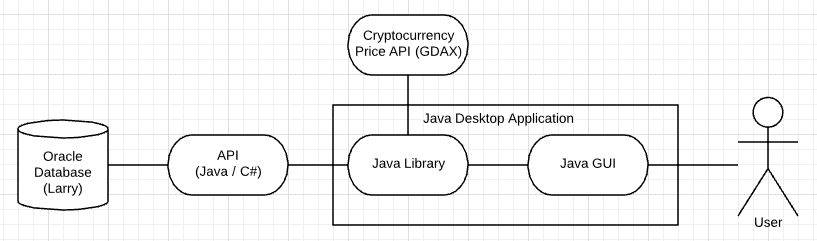
The data can be split into two tables – there are currencies and each currency has an exchange rate. The date and time combined with the currency’s ID field will retrieve the value of the currency at a given time. Storing additional fields within the exchange rate table may be required for a multi-client system. A single system could handle collecting and processing of data, leaving the end user system to simply get the recommendation from the database and act upon it.

Unified Modelling Language Diagram



The above diagram shows the implementation of the price collector within the Java library. It uses an API controller to collect JSON strings and then uses a parser to parse them and store them as a list of GDAXTrades for each Currency. When each minute is completed (for historic collection) it calculates an ExchangeRate for the minute and stores it in its relevant list of ExchangeRates related to the Currency. When sufficient data is collected it merges the lists and signals that the system is capable of performing predictions. *N.B. Obviously I didn’t plan on requiring a class of Helper methods, I have added to my original design as I have made alterations to implementation.*

## System Architecture Diagram



When the desktop application is activated the user is sees a graphical user interface. This GUI is populated by a library that collects data from a cryptocurrency exchange’s API endpoint, processes it and posts minutely exchange rates to an Oracle database through an API endpoint developed for the system.