The Strategy Pattern

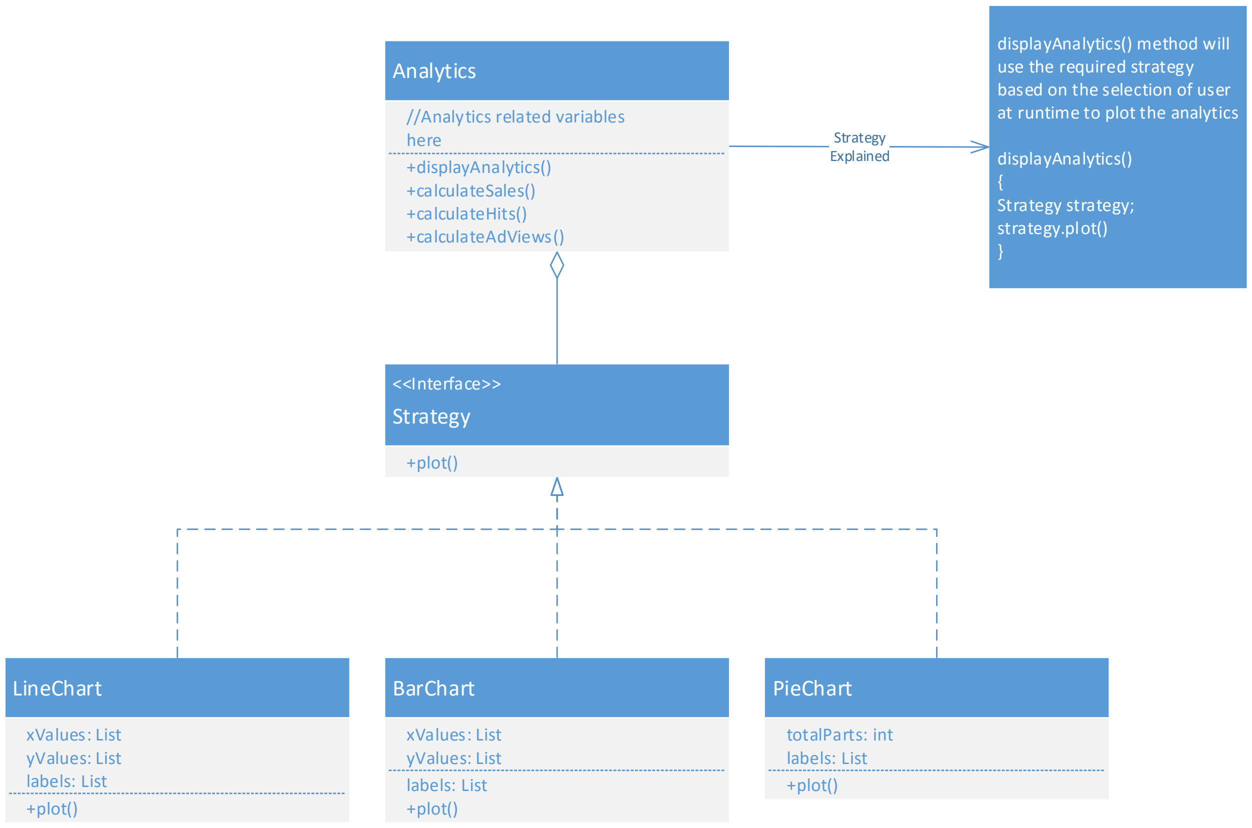
Object Oriented Goal: Keeping the implementation details as flexible as possible. Abstracting as much as possible.

Definition: Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it.

Implemented on classes:

1. Analytics
2. Strategy
3. LineChart
4. BarChart
5. PieChart

Design Snapshot:



Design Explanation:

There are three types of Analytics options and all the three types of analytics have different implementation details. It must be decided on runtime that which type should be used to process the Analytics. When the client selects the type of Analytics, the proper processing of Analytics needs to be done on runtime.

We can apply the strategy pattern to use the three types of Analytics processing options interchangeably. Whenever the client selects the type of Analytics, the proper processing strategy is created and processing of the Analytics is done based on the selection of the client. Below are some steps which explain the implementation of strategy pattern in our design.

- The Analytics class is the context class which will have a method which will require the processing of Analytics. In our case that method is the displayAnalytics() method.
- The Strategy is an interface which provides a function/method which has to be implemented by the classes which implement this interface. In this case, the plot() method is the method which will be implemented by the classes implementing Strategy interface.
- LineChart, BarChart and PieChart are the classes which have different implementation of the plot() method and are called on by the context whenever a strategy is desired to be performed.
- The plot() method contains different implementation in all the three classes and thus gives the client ability to select a strategy at runtime.