Group 19

Project 3 Design Patterns

25 October 2021

Design Patterns

For project 3, our team incorporated various creational, structural and behavioral design patterns. The creational pattern that fits within the context of our project is the factory method. This design pattern allows for creating objects in a superclass and enables subclasses to alter the type of objects that will be created. We applied this pattern to alter certain parts of the JavaFX library to do what we needed it to. For example, we wanted our textboxes to only allow dollar amounts to be entered into them. To do this we used a provided method from JavaFX to override the method that is defined in the cell that deals with how cells are updated. Now, when the price text boxes are typed in, the overridden method checks that the typed text matches a regular expression, if it does not then the text is not changed. This is the process that the factory method design pattern describes, the superclass usually creates the string object and puts it into the text box regardless, but the override changes what kind of object is created, in that it must match a certain expression.

The structural pattern we used throughout our project is analogous to a composite structure. We apply this structure when we unpack all the user information stored in a transaction object. Specifically, the transaction object can be decomposed into date, sign(+/-), item, price and category. With this data we can provide the user with specifications about expenses per category through a pie chart. As well as providing the user with a view of their progress toward their budgeting goals.

The behavioral pattern that we used to present the user with budgeting information is the mediator. In our program we define a UserAccount, which stores all the information the user has entered in various objects. The UserAccount has a list of transactions, a list of categories, etc. This UserAccount is our mediator object between the UI and the information we have gotten and potentially altered. For example, all the information for a transaction is taken from the UI, then sent to a method in UserAccount, which then creates a new transaction and stores it in a list. The UI does not access our classes for Transactions or Goals.