The attached java program, Budget Tracker, aims to create a solid foundation of custom data-structures that can be implemented to accept expenditures or yearly figures from one person, or yearly figures from multiple people. It does so, by using the object-oriented principles of inheritance, composition, polymorphism, generics, and enumerations. It also uses two design patterns including multiple singleton objects and multiple facades.

To accurately understand the inheritance and composition principles that were put into practice, please reference the attached U.M.L. diagram. However, in words, there are four compositions: Family has an ArrayList of Persons; Budget has an array of Expenditures; Expenditures have two enumerations, one fore regularity and one for types. There are two instances of inheritance: A user is a person, and a member is also a person.

Polymorphism was implemented effectively in the family object because the single user and multitude of members that exist in the references of the parent person object can exist in the family object’s ArrayList. Also, polymorphism is apparent in the driver class when instantiating person objects before determining their exact instance using the ‘instanceof’ keyword.

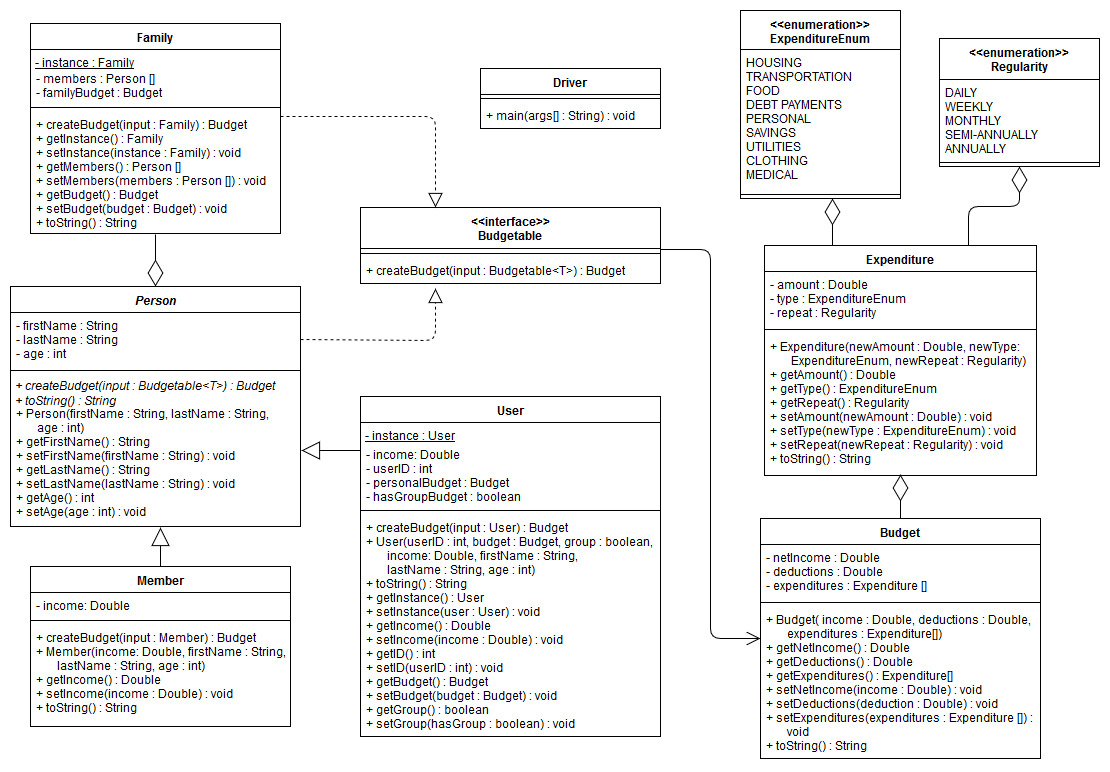
Generics were only implemented in the interface budgetable; that is, the interface budgetable has a method createBudget that accepts any implementation of the budgetable interface, including the related objects user and member, and the unrelated object that is a family. All are budgetable.

Enumerations were definitively implemented as to represent the regularity and type of expenditures, an important aspect of interpreting and suggesting budget changes to the user. The type of expenditures contains the type name, and percentage of yearly spending[[1]](#footnote-1) suggested for that type. The regularity of expenditures contains the regularity name, and the number of occurrences of that time in one year.

Two design patterns were used in this project, the first being singletons. The solid foundation that this project aims to create allows only one user to be created (the user of the program) versus the unlimited amount of member that are allowed. Also, this project only allows for the creation of one family object; that is, only one family object is necessary at a given time, so to create more would be undermining the purpose of this program.

Throughout this project I have refined my use of try-catch statements because of the interfaces representation in the command line – user input must be checked for correctness in my attempt to always fail gracefully upon user error. Also, throughout this project, I have realized that slight derivations from the initial U.M.L. diagram (see below) must be made for a minor reduction in complexity.

Because this projects goal was to create a solid foundation of a budget tracking software, there is less than a desired amount of output including (but not limited to) suggested allowances for spending, estimation of retiring age, and recommended savings amounts, both personal and for a family budget. That is not to say, however, that more data types could not be implemented for a larger plethora of informational gains. Also, if I were to further this project – the desire to output much of the data to an excel spreadsheet for visualization would be viable, if not exciting.



1. “How Much Money You Should Spend on Living Expenses - 2018 Budgeting Guidelines for Income.” Credit

   Counselling Society, www.nomoredebts.org/budgeting-guidelines. [↑](#footnote-ref-1)