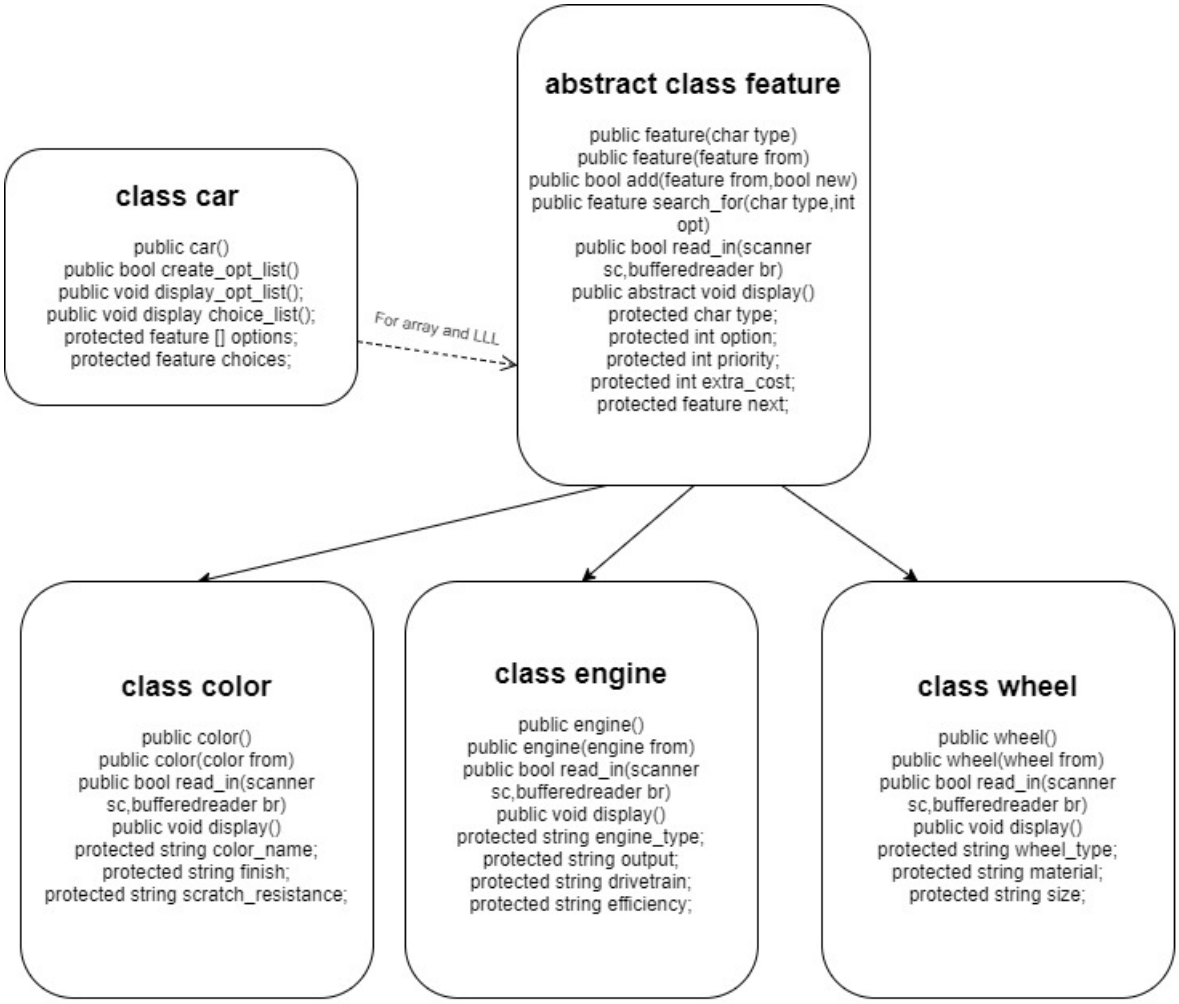
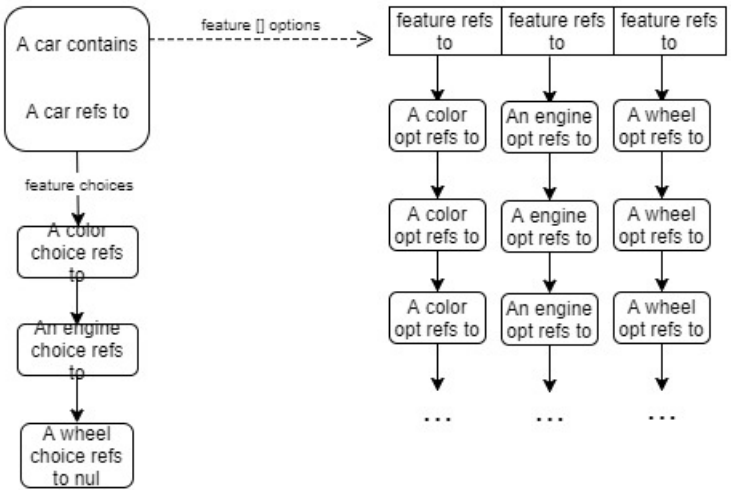
CS 202 DESIGN AND UML DIAGRAM FOR PROGRAM 4-5

BY TEJAS MENON; ASSIGNMENT BY PROFESSOR KARLA FANT



Data Structure



Note

Each instance of the class 'car' will not contain an 'option' array for program 5. I'm collating the option array here because only one instance of car will be made for the first part of the program. Additionally, note how the feature class performs the dual action of both creating options to read in from the external file and also the LLL of choices that the user selects. For program 5 this will be extended to randomly create the choice LLL using options, and each car will be arranged in an outer BST 'lot' class.

For Program 4, I will implement 5 classes (possibly 1 more with program 5), all of which will in totality allow for user selection, a pure virtual display, a base class external file reading in, and also allow functionality to store the user's choices in a LLL (One for choice of color, engine and wheel). This 'user' linked list generation will aid me in program 5, where each car must contain one such LLL randomly assigned for each feature. Each option under these features will also contain a priority listing between 1-3, as in a user will most likely not want color prioritized over engine, and similarly wheels over color.

For efficiency, the base class 'feature' will support functionality both for user selection and also cataloging features- in essence for both storage of all options read in from a file (for a particular feature) and also for curating the three user choices. This will be supported by way of upcasting the 'next' feature reference to store more options of the same feature and alternatively also point to another feature altogether. The three functions for supporting these two distinct operations would be the add() for the user LLL, which will use the feature passed in to store either into itself or to a new node appended at the end (depending on the boolean new). Prior to this, to find the feature to append to this user LLL, the car/client must first request a search_for() operation on the options array, which will use the char type and int opt to ascertain the corresponding option of a particular feature, the reference of which will be returned and passed to the add() function. To read in from an external file that will contain information for all three features, a peek requirement would be necessary- as in there should be a way to view what feature the next group of entries belong to. For this, I will need to use the bufferedreader and scanner objects- one for marking and returning to a prior point of read, and the other for convenient style of reading in (using bufferedreader for reading doesn't allow for a separate integer and float read in). Therefore, both will be needed to be passed into the base class add() that overrides the add() in feature. Subsequent to the reading in for the base, the add() in the base class will be called that then either requests the same function on the next feature option (after creating it) or returns after realizing eof() has been reached or the type of feature has changed. Each time a new option is created on the LLL, an option number will be read in from the external data file that will correspond to that option.

The general direction of execution

Before any user interaction, the program will load up the options array from the external data file using the functions listed above, and following this will display a menu of choices for the user requesting three choices to be made- one for color, engine and wheels. Each choice will be composed of a character denoting the type, and an integer denoting the option. After receiving this information, the search_for() and add() functions will create a new LLL of choices. These can then be displayed to the user upon program exit, and additionally the user will be able to create more cars and display them individually. For program 5, rather than relaying car creation control to a user, the 'car' class will be provided functionality to select three options randomly (one for each feature) and a higher level BST class will arrange car names by value.