DailyFoodLogApplication UML

Private:

- **DataHashTable*** the hash table pointer with open addressing where the data read from file while stored. Integer is the key value given by user, T is template type to hold data read.
- **DataBST*** the pointer to the root of the BST which holds the sorted addresses of the data
- **FoodHashTable*** the hash table which will index all of the foods users can add to their daily logs.

Public:

Required methods for project minimum functionality

+ **addDayLog(foodStack*)** this method adds a new daily food log with a stack of given foods
+ **removeDayLog(KEY**) this method removes a daily entry for an primary key
+ **retrieveDayLog(KEY)** retrieve the data for a day log with the given primary key
+ **printHashTable( )** print the data in the hash table in key - value format
+ **printTree( )** print the BST data in indented tree format

Needed methods for app specific functionality

+ **addFoodToDay(*foodPtr)** adds a food item to the stack of foods held on the current day. Uses the sorted BST to track foods. Can only be applied to current day and none past.
+ **removeFoodFromDay(*foodPtr)** searches the current stack of foods in the day on hand and if given food exists, removes first found instance of it. Can only act on current day.
+ **addFoodToList(string name, int protein, int carb, int fat)** creates a new food entry in the Food BST by first checking if one by same name exists, and if not creating the object and finding its place in the sorted tree

Optional Functionality:

+ **suggestFood( )** offers a set of options for food combinations to fulfill the remaining macronutrients needed in an incomplete day based on the average number of foods eaten in days beforehand, all using the sorted Food BST
+ **graphData( )** presents the number of days on target, days missed, average consumption, etc. Can also draw a simple bar bar graph or dot plot in the console