Main

- + main(args): void | launches GUI and creates an instance of MealLIst and Food Data
- - launch(): void

MealList

- - mealItemList: List<FoodItem>
- - nutritionSummary: HashMap<String, Double>
- ----------
- + MealList()
- + addItem(foodItem): void | adds a food item to mealItemList
- + removeItem(index): void | removes a food item from mealItemList
- + analyzeMeal(): void | traverses the entire list and adds up all nutritions

FoodData

- - foodItemList: List<FoodItem>
- - indexes: HashMap<String, BPTree<Double, FoodItem>>
- ----------
- + FoodData()
- + loadFoodItems(filePath): void | loads a file containing a list of food
- + filterByName(substring): List<FoodItem> | filters the food list by name
- + filterByNutrient(List<String> rules): List<FoodItem> | filters the food list by nutrition
- + addFoodItem(foodItem): void | adds a food item to foodItemList
- + getAllFoodItems(): List<FoodItem> | return a list of all food items
- + saveFoodItem(String fileName): void | save the current foodItemList to a file

BPTree

- - root: Node
- - branchingFactor: int
- ----------
- + BPTree(branchingFactor) | creates a BPTree with a specified branching factor
- + insert(key, value): void| inserts a node
- + rangeSearch(key, String comparator): List<V> | return a list of values that fits a specified range
- + toString(): String | converts the object to a String representation

Node (asbtract)

- keys: List<K> | a list of keys
- ----------
- Node()
- ----------
- insert(K, V): void | inserts a key-value pair
- getFirstLeafKey(): K | returns the key of the first leaf
- split() : Node | splits a node when there is an overflow
- rangeSearch(K, comparator): List<V> | returns a list of values that fits a specified range
- + toString(): String | converts the object to a String representation

InternalNode

- children: List<Node> | a list of Node as the children of this internal node
- ----------
- InternalNode()
- getFirstLEafKey(): K | returns the key of the first leaf
- isOverflow: boolean | checkes if the # of children exceeds (branchingFactor - 1)
- insert(K, V): void | inserts a key-value pair
- split(): Node | splits a node when there is an overflow
- rangeSearch(K, comparator): List<V> | returns a list of values that fits a specified range

LeafNode

- values: List<V> | a list of nutrition values extracted from FoodItem
- next: LeafNode | reference of the right adjacent node
- previous: LeafNode | reference of the left adjacent node
- ----------
- LeafNode()
- getFiirstLEafKey(): K | returns the key of the first leaf
- isOverflow(): boolean | checkes if the # of children exceeds (branchingFactor - 1)
- insert(K, V): void | inserts a key-value pair
- split(): Node | splits a node when there is an overflow
- rangeSearch(K, comparator): List<V> | returns a list of values that fits a specified range

FoodItem

- - name: String | the name of this food item
- - id: String | the unique identification code of this food item
- - nutrients: HashMap<String, Double> | contain the nutritional information of the food item
- - index: int | a number determined by the FoodItem's position in the list
- ----------
- + FoodItem(String id, String name) | initialize the instance's id and name
- + getIndex(): int | return the index pertaining to the list
- + getName(): String | return the name of this FoodItem instance
- + getID(): String | return the id of this FoordItem instance
- + getNutrients: HashMap<String,Double> | return a list of all nutrient values, along with the name of nutrient
- + setIndex(): void | set this instance's index pertaining to the list
- + addNutrient(String name, double value): void | add additional nutrient info to the instance
- + getNutrientValue(String name): double | return the value of a specific nutrients, identified by a String passed in

GUI

- + start(): void | launch java FX
- + handle(ActionEvent event): void | recognize and handle the actions passed in from user interface

BPTreeADT
• + void insert(K, V) • + rangeSearch(key, comparator): LIst<V> • + toString(): String

FoodDataADT
• loadFoodItems(filePath): void • + filterByName(substring): List<FoodItem> • + filterByNutrient(List<String> rules): List<FoodItem> • + addFoodItem(foodItem): void • + getAllFoodItems(): List<FoodItem> • + saveFoodItem(String fileName): void