# NotQuizlet

Collaborators: Michael Cates (group leader) and Mackenzie Raleigh

Group Name: GitGang

Program Release Date: Friday, November 1, 2024

Course: CSC 1310-104

A child looking at camera

Description automatically generated

## Overview

Our program is definitely NOT Quizlet. The user will definitely not be able to create a profile including their name, age, year in school, and collection of study sets. They won’t be able to enter lists of terms and definitions that they will be able to study and quiz themselves over as well as edit along the way. And they DEFINITELY will NOT be able to save their student information, including study sets to a txt file that they will be able to access long after the program stops running. Our program most definitely is NOT Quizlet.

## Program Flow

The vast majority of the program flow of NotQuizlet! is up to the user. From the moment the program begins the user is in control of all actions. With that in mind, there is a typical hierarchy to the functions and classes of the program that help in the storage of the user’s data.

The hierarchical structure of the program that follows is as such:

1. **Creation of a Student class object**
   1. The student class is at the top of the hierarchy, eventually holding objects of each of the other classes inside
   2. This is the data class of the program, holding the user’s name, age, year in school, and a HashTable holding all of their StudySet objects
2. **Creation of the HashTable class object**
   1. The HashTable class is designed to be second on the pyramid of classes in this program, as it is stored inside of the Student class but holds all of the user’s StudySet objects
   2. Each bucket in this hash table points to a StudySet linked list
3. **Creation of the StudySet class object**
   1. This class, while at the bottom of the hierarchy of the program is extremely vital
   2. It is where the user’s terms and definitions are stored for a specific study set
      1. Each study set has it’s own name and key
   3. Every function in this program is centered around being able to access this class

The entirety of the project can be summarized into these three Classes. They flow into each other from bottom to top and are defined from top to bottom.

## Classes breakdown

Each of the classes in this project are simple to break down into their moving pieces. They create a fairly, straightforward flow of data storage and retrieval.

**Student Class**

Private Storage:

1. User’s name - string
2. User’s age - integer
3. User’s year in school - integer
4. HashTable of StudySet objects

Public Storage:

1. Default constructor method:
   1. Name = N/A
   2. Age = 18
   3. Year = 1
2. Overloaded constructor method:
   1. All data is set to user input
3. Destructor method

Methods:

1. All necessary getter methods:
   1. Returns the information asked for such as name, age, year In school, etc.
2. All necessary setter functions
   1. Sets user data to the information passed through the function such as name, age, year in school, etc.
3. printStudent()
   1. displays all of the user’s information to them including the name of each of their study sets from the HashTable
4. addSet()
   1. adds the StudySet object passed through the function into the user’s HashTable
5. editASet()
   1. self explanatory, allows the user to edit any term or definition in any set
6. createASet()
   1. allows the user to create a study set, which is then passed through the addSet function to insert it into the HashTable
7. quizASet
   1. allows the user to quiz themselves over any set they would like and the function will return the questions they got wrong, along with the correct answer to those questions
   2. function will also return the user’s quiz grade, so they can track their progress
8. studyASet
   1. functionality is extremely similar to the quiz, just without the quiz grade and the function will immediately provide feedback on answers, rather than at the end
   2. the user will also be able to run this function indefinitely, meaning after every question, they are given the option to get another question or return to the menu

**HashTable Class**

Private Storage:

1. structure for HashEntrys
   1. key
   2. pointer to a value
   3. pointer to the next entry
2. table size – integer
3. array of hash entries

Public Storage:

1. Default constructor:
   1. Table size is set equal to zero
2. Overloaded constructor:
   1. Table size is set equal to user input
3. Destructor

Methods:

1. StudySet value:
   1. Returns a user’s study set from the given key
2. putValue(int key, StudySet \*value)
   1. inserts the passed StudySet into the key position of the HashTable
3. removeValue(int key)
   1. removes from the HashTable the value at the key position
4. printHashTable()
   1. displays all of the user’s study sets and their information for the user to review
5. getSize()
   1. returns the value of table size
6. setSize()
   1. sets the value of table size to the passed value
7. isEmpty
   1. returns a bool value of true of false representing whether the hashtable is empty or not

**StudySet:**

Private Storage:

1. string name
2. int key (which is set to the length of the name)
3. structure listNode to store the current node of a list along with the next node
4. listNode pointer to the list head
5. listNode pointer to the list tail

Public Storage:

1. default constructor
   1. sets head and tail to NULL
2. destructor
3. isEmpty()
   1. returns true or false on whether the list is empty or not
4. getLength()
   1. returns the length of the list
5. search(string value)
   1. returns the position of the value passed into the function
6. getNodeValue(int position)
   1. returns the value at the position passed into the function
7. appendNode(string value)
   1. adds the value passed into the function to the end of the list
8. insertNode(int position, string value)
   1. inserts the value passed into the function at the position of the list passed into the functio
9. deleteNode(string value)
   1. deletes the value passed into the function from the list
10. displayList()
    1. displays all list content in an easy to read format to the user
11. insertionSort()
    1. sorts the list utilizing the insertion sort algorithm
12. swap(int position1, int position2)
    1. swaps the values at the two locations in the list passed through the function
13. getName()
    1. returns the name of the list
14. getKey()
    1. returns the key for the list
15. setName(string)
    1. sets the name of the list to the string value passed into the function
16. setKey(int)
    1. sets the key for the list to the integer value passed into the function

## Function breakdown

NotQuizlet! Also utilizes an addFile() function which takes in the user’s Student object and a file name

* This function will take in that information in order to save the user’s student information, including the data from the Hash Table and save it to a txt file
  + this enables the user to save their study sets for future use without needing to run the program again to re-create them

## The Driver File

NotQuizlet.cpp - This file is where the program comes to life. It begins with displaying a small intro message to the user and then asking if the user would like to begin creating their student database.

* Afterwards, the program asks the user their name, age, and year in school
  + Once this information has been obtained, it will set the name, age, and year attributes of the Student object to those values
* The program then starts a while loop that displays the menu (list of tasks the program can run) to the user and allows the user to perform these tasks as many times as they would like
  + 1 – See the student information
  + 2 – create a study set
  + 3 – edit a study set
  + 4 – quiz over a study set
  + 5 – study a study set
  + 6 – end the program
* After the while loop has concluded and the user wants to end the program, they will be asked if they would like to save their student information to a txt file.
  + If so, they enter their desired file name and the addFile function is called
  + If not, the program will end