Design document

- **Overview of Class:**

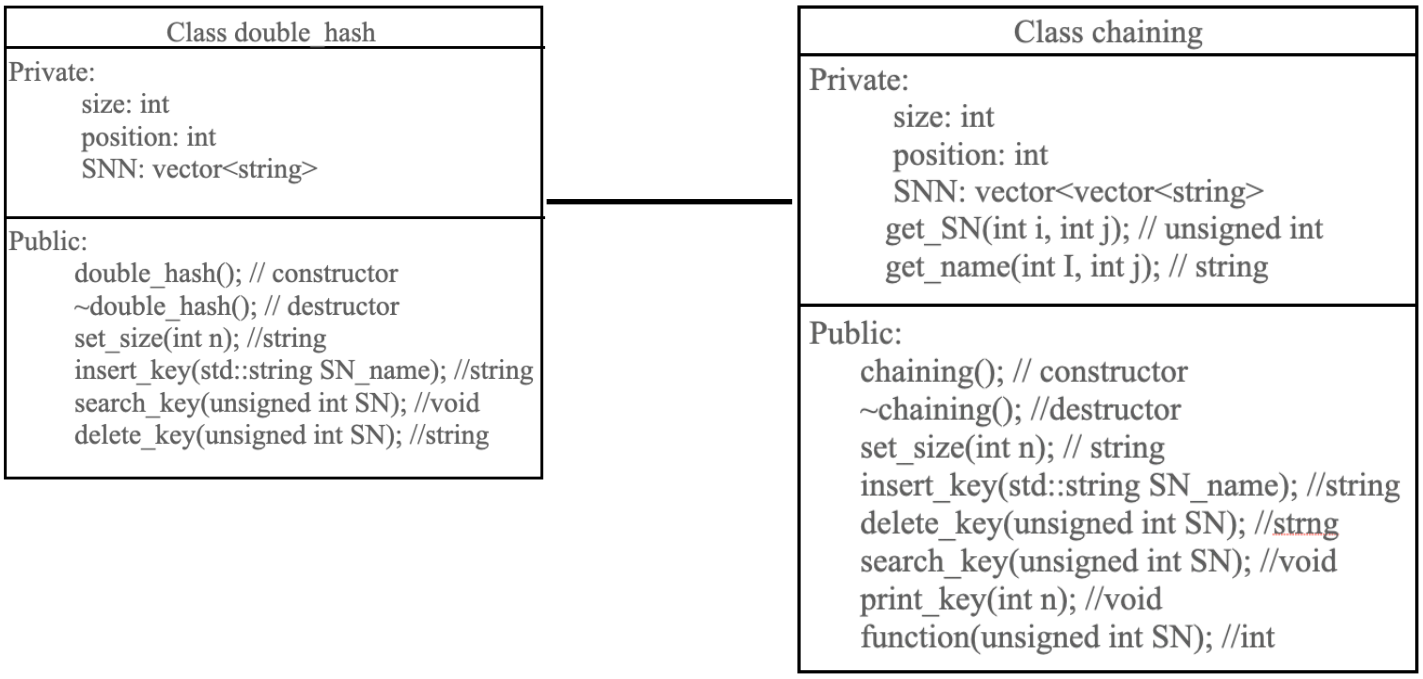
  class double_hash:

  The double_hash class creates a hash table that stores and deletes student numbers and last names. By inputting the number of student numbers to be stored, we can create a hash table with the size of the number of student numbers. By using two functions, we can store each student number and its related last name in a space in the hash table that has not been occupied. The double_hash class also allows users to search for student numbers in the table.

  class chaining:

  The chaining class stores student numbers and their related last names. By using the function (student number) mod (the number of student numbers needed to be stored), we can find the row the student number and last name needed to be stored. Then, by sorting the keys in the same row, we can insert the key in a descending way. Users can also delete, search, or print student numbers using this class.

- **UML Diagram**



- Detail on design decision:

  Class double_hash:

  One constructor and one destructor is created.

  Reason: to set the initial size, position, and the size of the hash table to 0;

  Class chaining:

  One constructor and one destructor is created.

  Reason: to set the initial size, position, and the row of the hash table to 0.

  No operators being override here.

  Function parameter:

  double_hash:

  1. Function without parameter: double_hash(); ~double_hash();

2. Function with parameter:

std::string set_size(int n);

the parameter n is passed by reference to set the size of the hash table;

std::string insert_key(std::string SN_name);

the parameter SN_name is passed by reference since user needs to input the student number and the last name, and store it into the hash table;

void search key(unsigned int SN);

the parameter SN_name is passed by reference since users need to input the student number (key) to find whether the key is in the hash table

std::string delete_key(unsigned int SN);

the parameter SN_name is passed by reference since users need to input the student number (key) they want to delete in the hash table;

chaining:

1. Function without parameter: chaining(); ~chaining();
2. Function with parameter:

std::string set_size(int n);

the parameter n is passed by reference to set the size of the hash table;

std::string insert_key(std::string SN_name);

the parameter SN_name is passed by reference since user needs to input the student number and the last name, and store it into the hash table;

void search_key(unsigned int SN);

the parameter SN_name is passed by reference since users need to input the student number (key) to find whether the key is in the hash table.

std::string delete_key(unsigned int SN);

the parameter SN_name is passed by reference since users need to input the student number (key) they want to delete in the hash table;

void print_key(int n);

the parameter n is passed by reference to indicate which row of the hash table needs to be output.

int function(unsigned int SN);

The parameter SN is passed by reference to get the row that the student number and the last name needs to be insert/search/delete.

unsigned int get_SN(int i, int j);

The parameter i and j are passed by reference to indicate the precise location(which row, which column) of the element in the vector, and then extract the student number from the element.

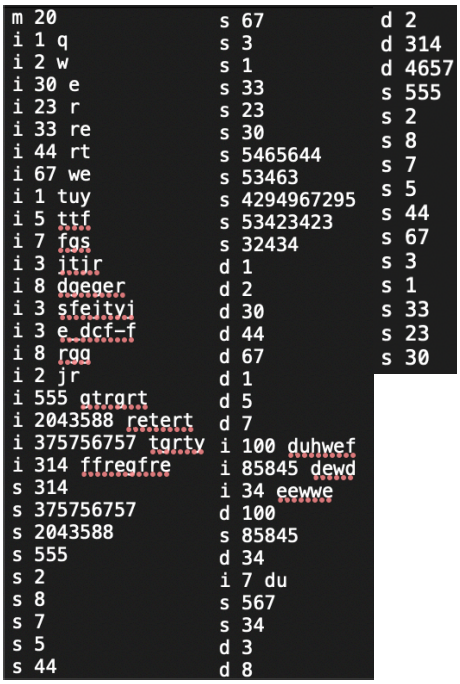
unsigned int get_name(int i, int j);

The parameter i and j are passed by reference to indicate the precise location(which row, which column) of the element in the vector, and then extract the student name from the element.

- Test cases:

double_hash: create a new table, and input data to check if the data can be insert in the right place, delete elements, and then check for the deleted element, and insert new elements again. Insert duplicate elements, delete elements that are not in the table. Check if the output last name is correct.

example test case open:



```
m 20
i 1 q
i 2 w
i 30 e
i 23 r
i 33 re
i 44 rt
i 67 we
i 1 tuy
i 5 ttf
i 7 fgs
i 3 jtjr
i 8 dgegeг
i 3 sfejtyj
i 3 e_dcf-f
i 8 rgg
i 2 jr
i 555 gtrgrt
i 2043588 retert
i 375756757 tgrty
i 314 ffregfre
s 314
s 375756757
s 2043588
s 555
s 2
s 8
s 7
s 5
s 44
s 67
s 1
s 33
s 23
s 30
s 5465644
s 53463
s 4294967295
s 53423423
s 32434
d 1
d 2
d 30
d 44
d 67
d 1
d 5
d 7
i 100 duhwef
i 85845 dewd
i 34 eewwe
d 100
s 85845
d 34
i 7 du
s 567
s 34
d 3
d 8
d 2
d 314
d 4657
s 555
s 2
s 8
s 7
s 5
s 44
s 67
s 3
s 1
s 33
s 23
s 30
```

Chaining: create a new table, and insert a few data into the table. Delete some of the data, and print the data remain in the table out to check if the order is descending, and check the empty row. Check if the output last name is correct.
m 20 i 100 a i 20 c i 90 d i 60 d s 100 p 0 p 1 d 200 d 90 i 60 k i 90 d p 3

- Performance consideration:
  - double_hash:

    set_size(int n): O(n) //initialize all the elements in the vector to “0” using for loop;
    Insert_key(std::string SN_name): O(n) // check duplication using for loop; use while loop to get the student number, and insert the key into the list linearly.
    search_key(unsigned int SN): O(n) // use for loop to check whether the key is in the hash table;
    delete_key(unsigned int SN): O(n) //use for loop to get the position of the student number stored in the table, and then compare the student number in the vector with SN linearly.

  - chaining:

    set_size(int n): O(n) //initialize each row of the table with one column and value “0”;
    Insert_key(std::string SN_name): O(n) // extract the student number from SN_name O(n); get the position and the size of the vector to avoid segmentation fault O(1); check if the row is empty O(1); check if the student number needs to be insert is the smallest in the row O(1); check if student number needs to be insert is neither the biggest or smallest, and insert the SN_name using for loop O(n).
    search_key(unsigned int SN): O(n) // use for loop to check whether the key is in the hash table;
    delete_key(unsigned int SN): O(n) // the the row SN may at linearly O(1), use for loop to search. For that key O(n) = O(n-i) + O(i)
    void print_key(int n): O(n); //print each column in row n using for loop;
    int function(unsigned int SN): O(1) // get the position = SN mod size O(1)
    unsigned int get_SN(int i, int j): O(n) // use the while loop to get the length of the student number, and return student number.
    unsigned int get_name(int i, int j): O(n) //use the while loop to get the length of the student number, and get student name by substr.