d. Class OrderedHashTable

This class represents a hash table that resolves collisions using a technique called separate chaining where the chains are ordered by student number. It is implemented by inheriting from the HashTable class. Since the data stored in this class is a vector<Student>, its mother class is HashTable<vector<Student>>. I chose to use vector class as a chaining container since its size can change dynamically, and it has methods such as emplace and erase where I can perform insertion and deletion from the position I want.

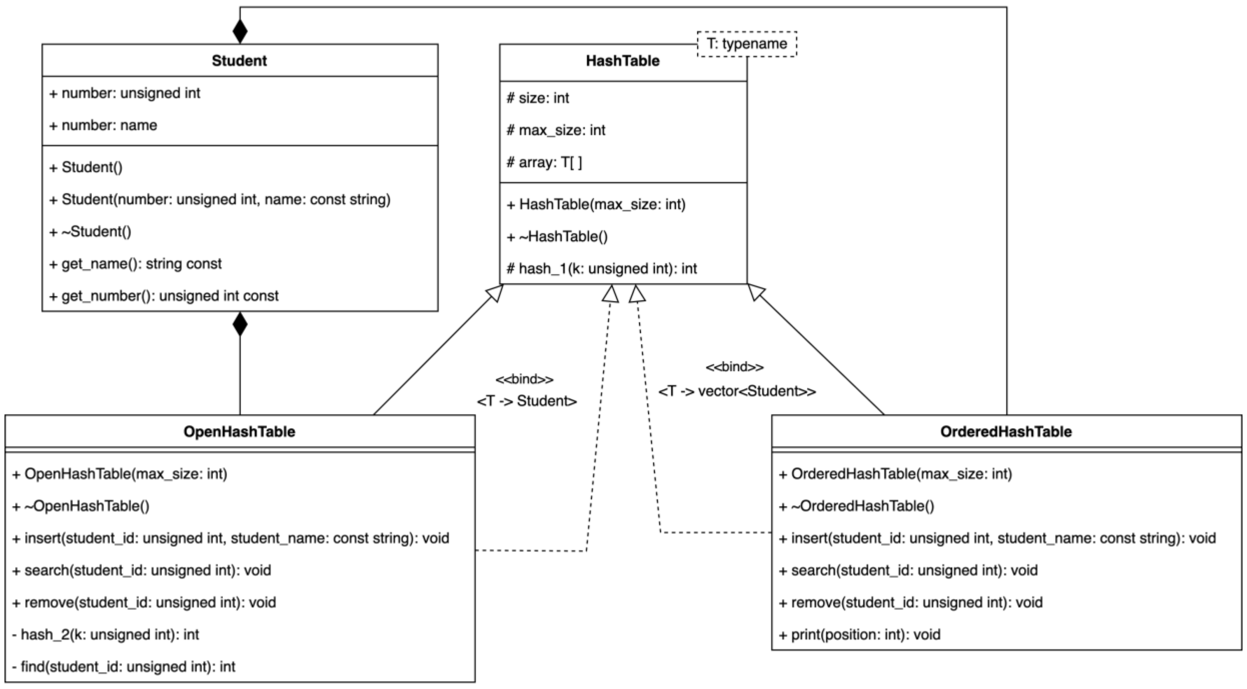
Member variables/functions:

- **+** void insert (unsigned int student_id, const string student_name):
  1. Using the primary hash function, find the appropriate index to insert the key and value
  2. If the key is the first key to be inserted in such position, simply emplace_back to the chain vector
  3. If the key is already in the chain, print out "failure"
  4. Else, by comparing the student_id (key) with keys in the chain, find the appropriate position to insert
  5. Insert the key and value and print out "success"
- **+** void search (unsigned int student_id):
  1. Using the primary hash function, find the chain in which the key is located from the table
  2. If the key was found in the chain at position p of the table, print out "found LN in p"
  3. Else, print out "not found"
- **+** void remove (unsigned int student_id):
  1. Using the primary hash function, find the chain in which the key is located from the table
  2. If the key was found in the chain, erase the key from the chain vector and print out "success"
  3. Else, print out "failure"
- **+** void print (int position):
  1. Print the chain of keys that starts at "position" in descending order
  2. Separate keys in the chain by one space.
  3. If the chain is empty, print "chain is empty"

Return types of all public member functions of Open/OrderedHashTable classes are void, which means, these classes never return a value. Instead, they will print out a message indicating the result of the operation in accordance with the project requirements.

*** +: public / #: protected / -: private**

2. UML Class Diagram


Student
+ number: unsigned int
+ number: name
+ Student()
+ Student(number: unsigned int, name: const string)
+ ~Student()
+ get_name(): string const
+ get_number(): unsigned int const
HashTable
T: typename
# size: int
# max_size: int
# array: T[ ]
+ HashTable(max_size: int)
+ ~HashTable()
# hash_1(k: unsigned int): int
<<bind>>
<T -> Student>
<<bind>>
<T -> vector<Student>>
OpenHashTable
+ OpenHashTable(max_size: int)
+ ~OpenHashTable()
+ insert(student_id: unsigned int, student_name: const string): void
+ search(student_id: unsigned int): void
+ remove(student_id: unsigned int): void
- hash_2(k: unsigned int): int
- find(student_id: unsigned int): int
OrderedHashTable
+ OrderedHashTable(max_size: int)
+ ~OrderedHashTable()
+ insert(student_id: unsigned int, student_name: const string): void
+ search(student_id: unsigned int): void
+ remove(student_id: unsigned int): void
+ print(position: int): void