myTA Hard Copy 7/30/2020, 9:15:58 PM

User Id : weizhe.goh@digipen.edu | Started : 2020.07.30 16:38:56 | Score : 100%

DigiPen Inheritance Assignment

Rules

Read carefully and check all rules you agree with:

- ■ Your code must represent your own individual work. Cheating of any kind (copying someone else's work, allowing others to copy your work, collaborating, etc.) will not be tolerated and will be dealt with SEVERELY.
- ■ Each exercise has description which must be strictly followed.
- ■ All programs must pass all tests in the main function (when given) to get the final grade. You are not allowed to make any change in the main function in this case.
- ■ Keep the code proper formatted (correct indentation, max line width is 40 characters).
- ■ Every week the instructor is available **during the lab time** to discuss following matters:
  - your disagreement with rule in this card,
  - misunderstanding of the current assignment specs,
  - solution for given problems.

★ Specs

Develop a new class Stack based on the one developed in the previous assignment.

- ■ New implementation must comply with new specs given below and specs given in the previous assignment. New specs are more important and override old specs.
- ■ The List class is the **base class** for the new Stack class.
- ■ The Stack class **protectedly** inherits the List class members.
- ■ The List class has members that have **common sense** for the list container. For example, pop and push (which are common for stacks) cannot be members of the list.
- ■ Do not expose the Head in the List. (Keep it private! Remove all getters and setters for the Head if you have them added in the previous assignment.)
- ■ You implementation must produce the expected output by the given tests in the main function.
- ■ Expected output is (without "): "1321Error"

★ Code 1 ☑

Run

```
#include <iostream>

template <typename T>
struct Node
{
    T data;
    Node* next;
};

template <typename T>
class List
{
protected:
    Node<T>* head;

public:
    List() :head(nullptr) {}

    void push_front(T value)
    {
        Node<T>* newNode = new Node<T>;

        newNode->data = value;
        newNode->next = head;

        head = newNode;
    }
```

Survey

- What is approximate number of hours you spent implementing this assignment?

```
2hours
```

- Indicate the specific portions of the assignment that gave you the most trouble

```
making list a declared identifier
```

```
};

template <typename T>
class Stack : public List<T>
{
public:
       void push(T value)
       {
           List<T>::push_front(value);
       }

        T pop() noexcept (false)
        {
            if (isEmpty() == true)
            {
                throw "Error";
            }

            T deleteValue = 0;

            deleteValue
            = List<T>::head->data;

            Node<T>* deleteNode=nullptr;

            Node<T>* current
            = List<T>::head;

            deleteNode = current;

            List<T>::head=current->next;

            delete deleteNode;
            return deleteValue;
        }

        bool isEmpty() noexcept
        {
            if(List<T>::head != nullptr)
                return false;
            else
                return true;
        }
};
```

```
int main() {
    Stack<int> s;
    std::cout << s.isEmpty();
    s.push(1);
    s.push(2);
    s.push(3);
    try {
        std::cout << s.pop();
        std::cout << s.pop();
        std::cout << s.pop();
        std::cout << s.pop();
    } catch(const char* msg) {
        std::cout << msg;
    }
    return 0;
}
```

```
1321Error
```

By signing this document you fully agree that all information provided therein is complete and true in all respects.	Responder sign: