Отчет по лабораторной работе номер 1

Энгельке Сергей Алексеевич. 931923

Лабораторная работа 1 состояла из двух частей. (Ч1 и Ч2 далее)

**1. Постановка задачи**

Ч1:

Запрограммировать шаблонный класс, реализующий стек. Стек основывается на односвязном списке.

Класс поддерживает: помещение обета в стек, извлечение объекта из стека, получение размерности стека. Добавить генерацию исключений( EstackEpmty ) + публичный метод char\* what(); Класса EstackEmpty

Ч2:

Реализовать класс PersonKeeper с методами readPersons и writePersons. Метод readPersons должен считывать информацию о людях из входного потока (файла), создавать на основе этой информации объекты класса Person, и помещать их в стек. Формат входного файла должен быть такой:

Фамилия Имя Отчество

**2. Предлагаемое решение.**

Ч1

"Стек" - это абстрактная структура данных, которая представляет собой список элементов, организованных по принципу LIFO (Last-In, First-Out).

"Шаблоны" - это средство языка C++, предназначенное для создания обобщенных алгоритмов и структур данных, позволяющее параметризировать типы данных.

Для реализации стека можно использовать однонаправленный связанный список, так как для структуры данных "стек" не требуется знать каждому элементу о том, кто находится до или после него.

Реализация стека может включать следующие операции:

* Push: добавление элемента в вершину стека.
* Pop: удаление элемента из вершины стека.
* Peek (или Top): получение значения элемента в вершине стека без его удаления.
* IsEmpty: проверка, пуст ли стек.
* Size: получение количества элементов в стеке.

Реализован отлов ошибок стека. Стук пуст, являющийся потомком класса EStackException

Ч2

Класс PersonKeeper был разработан как шаблон проектирования Singleton (Одиночка). Этот шаблон обеспечивает, что в однопоточном приложении будет только один экземпляр данного класса, предоставляя глобальную точку доступа к этому экземпляру. Конструктор класса объявлен как приватный, что предотвращает создание экземпляров класса за пределами его реализации.

Имеется статический метод для доступа к экземпляру, который создает экземпляр класса если он не был создан ранее. Ленивая инициализация. Так же в приватном поле содержится указатель на единственный экземпляр объекта.

Методы класса

1. readPersons: Этот метод возвращает стек (stack) с данными о персонах. Возвращает стек, содержащий информацию о персонах.
2. writePersons: Этот метод записывает данные о персонах в заданный поток (stream). В качестве аргумента принимает стек с данными о персонах и записывает этот стек в указанный поток.

UML

PlantUML

@startuml

class PersonKeeper <<Singleton>> {

- instance: PersonKeeper

- personsStack: StackTemplate<Person>

------------------------------

+ getInstance(): PersonKeeper

+ readPersons(filename: String): void

+ writePersons(filename: String): void

}

class Person {

- fullName: String

---------------------

+ Person()

+ Person(fullName: String)

+ getName(): String

}

class Node<T> {

- next: Node<T>

- value: T

---------------------

+ Node(value: T)

+ Node()

}

class StackTemplate<T> {

- head: Node<T>

- size: size\_t

---------------------

+ StackTemplate()

+ StackTemplate(stack: StackTemplate const&)

+ ~StackTemplate()

+ push(value: T): void

+ pop(): T

+ getSize(): size\_t

+ isEmpty(): bool

+ print(): void

}

PersonKeeper \*-- Person : Композиция

StackTemplate \*-- Node : Композиция

PersonKeeper ..> StackTemplate : Зависимость

PersonKeeper --> getInstance

namespace exceptions {

class Exception {

- message: char\*

+ explicit Exception(arg\_message: const char\*)

+ Exception(arg: const Exception&)

+ ~Exception()

+ what() const: const char\*

}

class ExceptionData {

+ explicit ExceptionData(arg: const char\*)

}

class ExceptionFile {

+ explicit ExceptionFile(arg: const char\*)

}

class EStackEmpty {

+ explicit EStackEmpty(arg: const char\*)

}

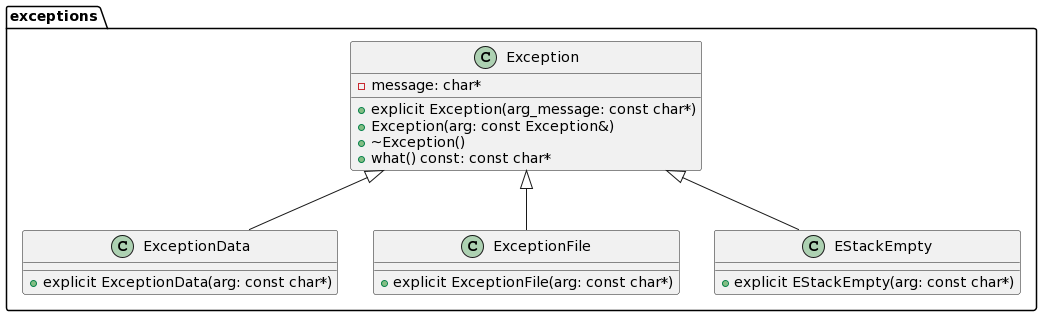
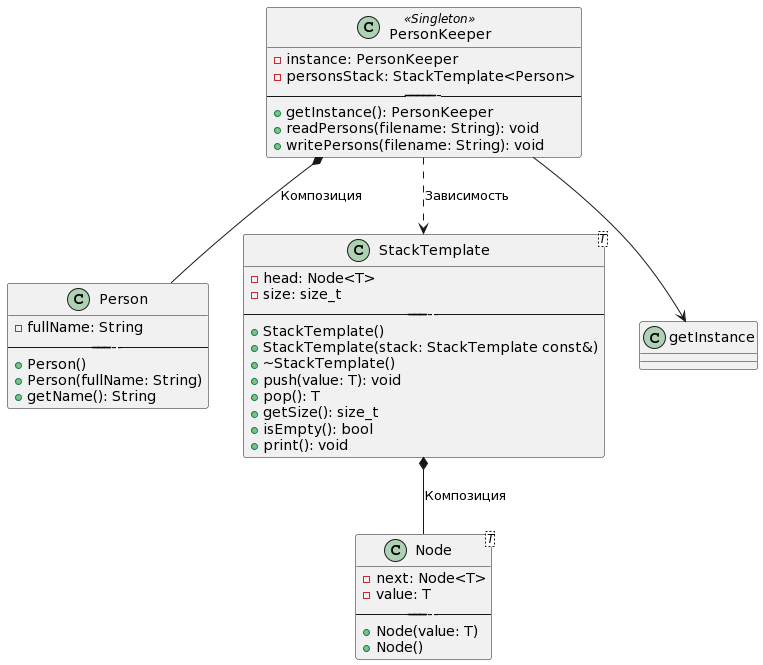
Exception <|-- ExceptionData

Exception <|-- ExceptionFile

Exception <|-- EStackEmpty

}

@enduml



**3. Код программы.**

<https://github.com/Amatikay/Lab_1_SDT.git>

**4. Инструкция пользователя.**

Ч1

Создаем объект с указанием типа данных

Stack<dataType> objectName;

Для заполнения стека используется метод Push

objectName.Push(dataType data);

Для извлечения элемента из стека

objectName.Pop();

Если попытаться извлечь элемент при условии, что стек пуст будет вызвано исключение

Ч2

Использую объявленный статический создающий метод, который будет использоваться для

получения одиночки.

(Далее я буду использовать куски кода из main.cpp)

PersonKeeper\* keeper = PersonKeeper::getInstance();  
keeper->readPersons("../data/input.txt");  
keeper->writePersons("../data/output.txt");

**5. Тестирование.**

Тестирование выполнено отдельным модулем GoogleTest.

Тесты можно увидеть в директории test. В отчете привожу снимок экрана, показывающие корректность тестов.

