Державний університет «Одеська Політехніка»
Інститут комп’ютерних систем
Кафедра інформаційних систем

Лабораторна робота №3

з дисципліни «Об’єктно-орієнтоване програмування»

Тема: «Інкапсуляція. Знайомство з мовою моделювання UML»

Виконав:
студент групи АІ-201
Мартинюк Д.В.

Прийняв:
доц. Годовиченко М.А.

Одеса 2021

Цель лабораторной работы:

- изучить принцип инкапсуляции и его реализацию в языке Java;
- ознакомиться с базовой нотацией диаграммы классов UML;
- научиться представлять классы в виде нотации диаграммы классов UML;
- научиться "читать" диаграмму и писать код исходя из диаграммы классов UML.

Ход работы:

1) Модификация классов

2) Создание UML-диаграмм

3) Класс IntStack

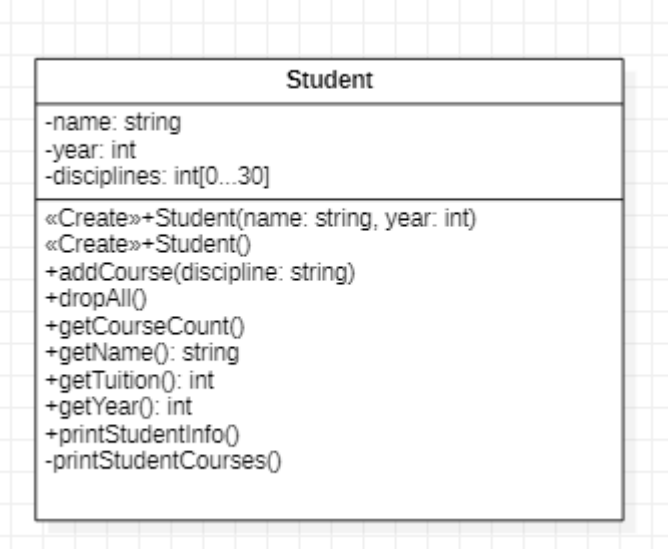
4) Создание класса по UML-диаграмме

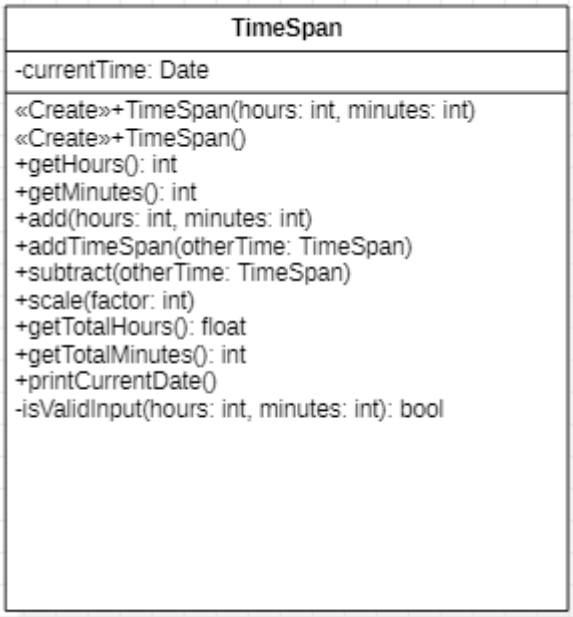
1. Модификация классов

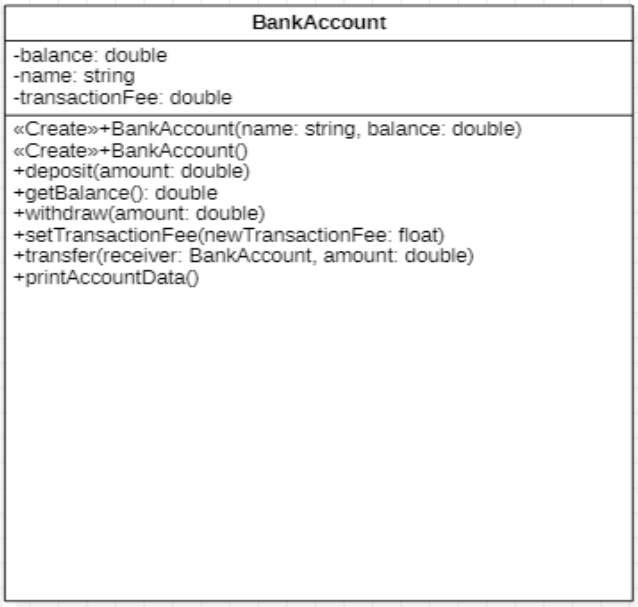
При выполнении лабораторной работы №2 модификаторы доступа были использованы заранее, нет нужды в модификации.

2. Создание UML-диаграмм

UML диаграммы к классам из лабораторной работы №2:

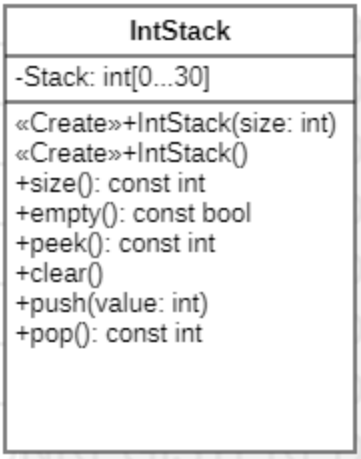



TimeSpan
-currentTime: Date
«Create»+TimeSpan(hours: int, minutes: int)
«Create»+TimeSpan()
+getHours(): int
+getMinutes(): int
+add(hours: int, minutes: int)
+addTimeSpan(otherTime: TimeSpan)
+subtract(otherTime: TimeSpan)
+scale(factor: int)
+getTotalHours(): float
+getTotalMinutes(): int
+printCurrentDate()
-isValidInput(hours: int, minutes: int): bool


BankAccount
-balance: double
-name: string
-transactionFee: double
«Create»+BankAccount(name: string, balance: double)
«Create»+BankAccount()
+deposit(amount: double)
+getBalance(): double
+withdraw(amount: double)
+setTransactionFee(newTransactionFee: float)
+transfer(receiver: BankAccount, amount: double)
+printAccountData()

3. Класс IntStack

UML диаграмма для класса IntStack



При реализации данного задания с помощью C++ возникли существенные

проблемы в реализации, проблема была связана с тем, что размещение объектов на стеке и на куче в языках программирования Java и C++ довольно таки разнятся.

Не смотря на то, что в Java статический массив тоже лежит на куче, он все равно считается статическим массивом , в связи с этим указатели на массивы на стеке могут быть присвоены указателям на массивы на куче, что является преимуществом.

В C++ стек и куча строго разделены, поэтому такие фичи не работают.

Несмотря на то, что в методических указаниях рекомендуется не использовать библиотечные коллекции , я был вынужден пренебречь данным условием ввиду отсутствия в C++ прямого функционала для решения этой задачи статическими массивами, было решено также не использовать устаревший и небезопасный подход с выделением памяти для динамических массивов с помощью операторов ***new*** и ***delete***(для возврата памяти на кучу) и взор был отдан в пользу ***std::vector*** по нескольким причинам:

а. По сути являет собой гибкий динамический массив

б. На его базе легко реализовать стек ввиду заложенного функционала

в. Вектор хоть и является полиморфной оберткой над динамическим массивом и обладает мощным функционалом, но все проблемы поставленной задачи не решает.

Код:

```
const int size()const { return this->Stack.size(); }
const bool empty()const { return this->size() == 0; }


const int peek()const { return this->Stack[this->size()-1]; }
void clear() { this->Stack = vector{ 0 }; }


const int pop()
{

	// Put returned value to store temporarily, return this value,
	// and get rid of that element in stack
	if (this->empty()) { return 0; }
```

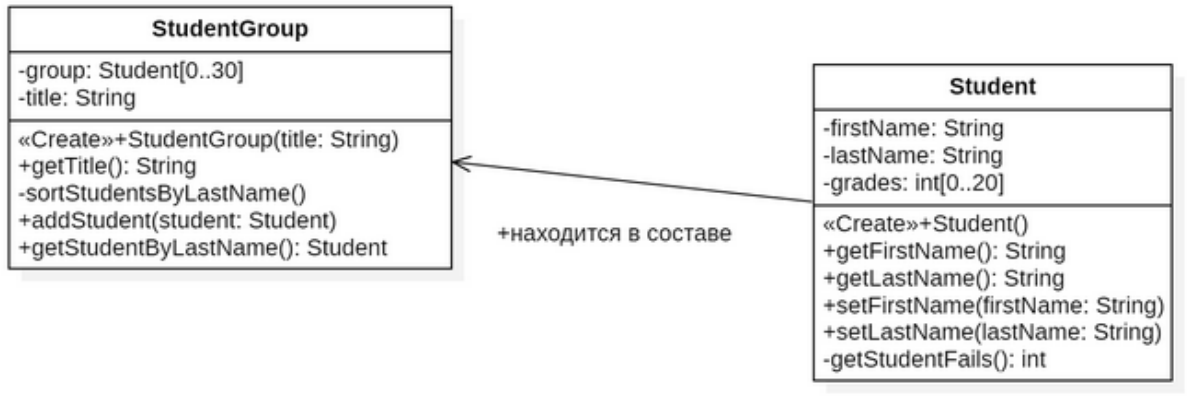
```
        int TopOfStack = peek();
        this->Stack.erase(Stack.end() - 1);

        return TopOfStack;

}
void push(int value) { this->Stack.push_back(value); }
```

4. Создание класса по UML-диаграмме

Была дана UML диаграмма:



Геттер возвращает объект по ссылке:

```
const Student& GetStudentByLastName()const {}
```

Достаточно важный момент, ведь получаемый объект может много весить(в данной ситуации не столь критично, объект string не столь большой) и время затраченное на копирование при вызове метода может сказаться на скорости выполнения, поэтому принято не POD члены возвращать по ссылке.

Выводы к лабораторной работе:

В ходе выполнения лабораторной работы было разработано несколько UML диаграм, а также освоено базовый функционал для их разработки в приложении

StarUML, разработано классы с примерами реализации базовой инкапсуляции.