Курсов проект по ООП

Система за управление на база от данни

Специалност: Компютърни науки

Курс: 1

Факултетен номер: 7MI0800207

Разработил: Михаил Цветков

СУ „Св. Климент Охридски“

# Увод

Проектът има за цел да създаде удобна за потребителя система за управление на база от данни, поддържаща основни операции за работа с таблици, като също и дава възможност за конвертиране на текстови файлове от вида *comma separated values* в таблици и обратно.

За постигане на този резултат са разработени посочените в условието функции:

* *open, close, save, saveas, help, exit*
* *import , export* – за добавяне и извеждане на таблица от базата
* *showtables* – за показване на наличните таблици
* *describe, print, select, count* – за извличане на информация от таблица
* *addcolumn, update, delete, insert, rename, agregate* – за промяна на данните или структурата на таблица
* *inner join* – за извършване на основна операция с таблици

Както и: *drop* – за премахване на таблица от базата

Поддържаните типове данни за съдържанието на таблиците са *int, double, string*, както и тип *null,* който се държи като един от горните три, но не е валиден тип на поле в таблица.

Важна забележка: Някои от желаните в условието принципи на работа на системата са умишлено промени, за да направят работата на потребителя по-лесна и удобна.

# Структура

* [Преглед на предметната област](#_Преглед_на_предметната)
  + [Основни дефиниции, концепции и алгоритми](#_Основни_дефиниции,_концепции)
  + [Дефиниране на проблеми](#_Дефиниране_на_проблеми)
  + [Подходи, методи за решаване на поставените проблеми](#_Подходи,_методи_за)
  + [Функционални изисквания](#_Функционални_изисквания)
  + [Нефункционални изисквания](#_Нефункционални_изисквания)
* [Проектиране](#_Проектиране)
* [Реализация](#_Реализация)
  + [Реализация на основните класове](#_Реализация_на_основните)
* [Управление на паметта](#_Управление_на_паметта)
* [Заключение](#_Заключение)
  + [Обобщение на изпълнението на началните цели](#_Обобщение_на_изпълнението)
  + [Насоки за бъдещо развитие и усъвършенстване](#_Насоки_за_бъдещо)

# Преглед на предметната област

### Основни дефиниции, концепции и алгоритми

За да се придържаме към терминологията на базите от данни, навсякъде се използват термините „запис“ (*record)* и „поле“ (*field*) вместо съответно „ред“ и „колона“. Синтаксисът на командите е подобен на този в SQL, но е сравнително опростен (в допълнение към задачата е осъществена възможност командите да съдържат произволен брой интервали (стига да не са нула), както и ключовите могат да бъдат записани и с малки, и с големи букви). При реализацията на командите не са използвани почти никакви „сложни“ алгоритми, освен такива за обхождане на таблица по записи, по полета и по записи и полета.

### Дефиниране на проблеми

Основните логистични проблеми при реализацията на системата бяха свързани с представянето на таблиците в паметта, както и начинът на съхранение и достъпването на файловете, съдържащи таблиците.

### Подходи, методи за решаване на поставените проблеми

За решаването на тези проблеми бе създаден класът *Table*, съдържащ цялата информация за заредена таблица, съставена от полета (*TableField*), като полето се определя от тип, име, и вектор от стойности. За добавяне на таблица в системата от вече съществуващ файл, му се създава копие в директорията „*Tables*“, а името на таблицата и адресът на новия файл се записват във архивния файл. За да се абстрахира достъпването на таблиците от самите команди е реализиран класът *TableManager*, а за самите команди е реализиран *Command design pattern.*

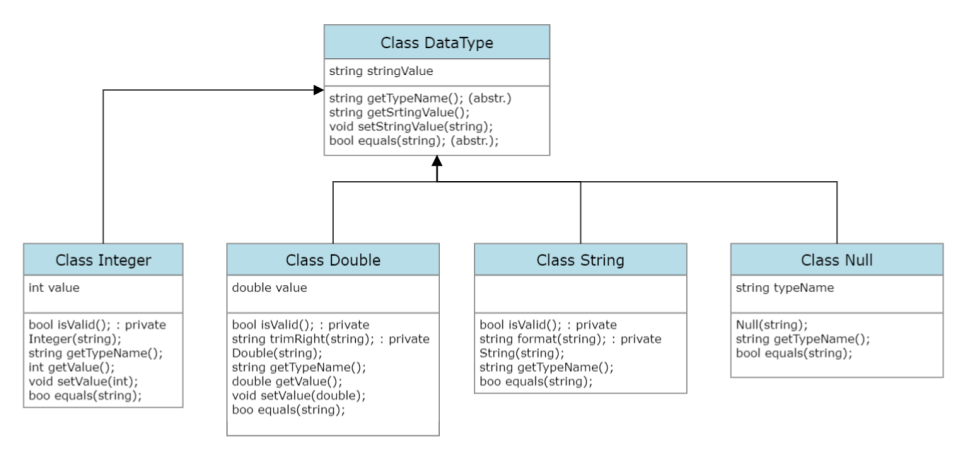
### Функционални изисквания

Системата може да изпълнява всички потребителски команди, описани по-горе, като в допълнение за улеснение на потребителя при работа с таблица не се изисква (но е възможно) предварително отваряне на таблицата (*open*), а системата автоматично отваря таблицата, която е поискана.

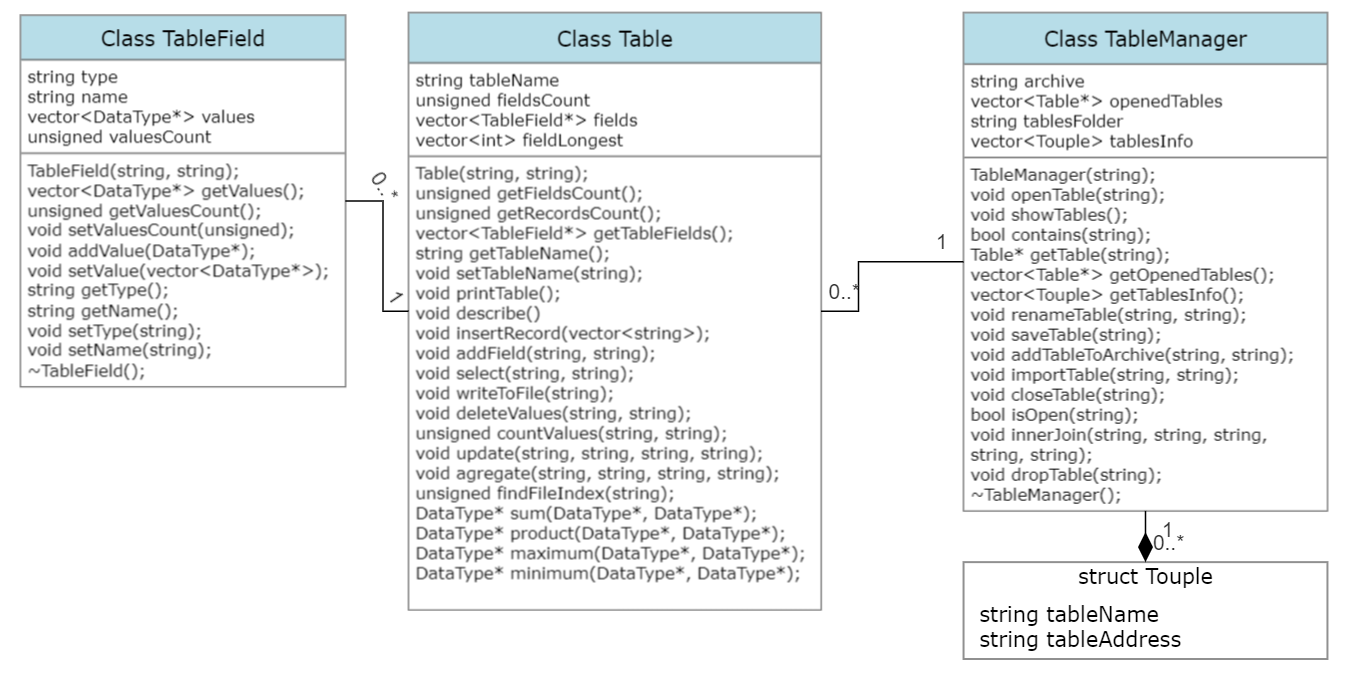
### Нефункционални изисквания

Благодарение на избраната архитектура, системата може да бъде поддържана лесно – улеснява се добавянето на нови команди и функционалности. Не е постигната добра преносимост поради факта, че системата е създадена с цел пряка комуникация с потребителя и се базира на визуално представяне на данните. От друга страна е на лице добра използваемост при работа с по-прости бази от данни, тъй като синтаксисът на командите е по-опростен от този на други системи за управление на бази от данни (MySQL, MariaDB, …).

# Проектиране

UML диаграми на архитектурата на проекта:

* Типове данни:
* Обща архитектура



* *Command pattern*

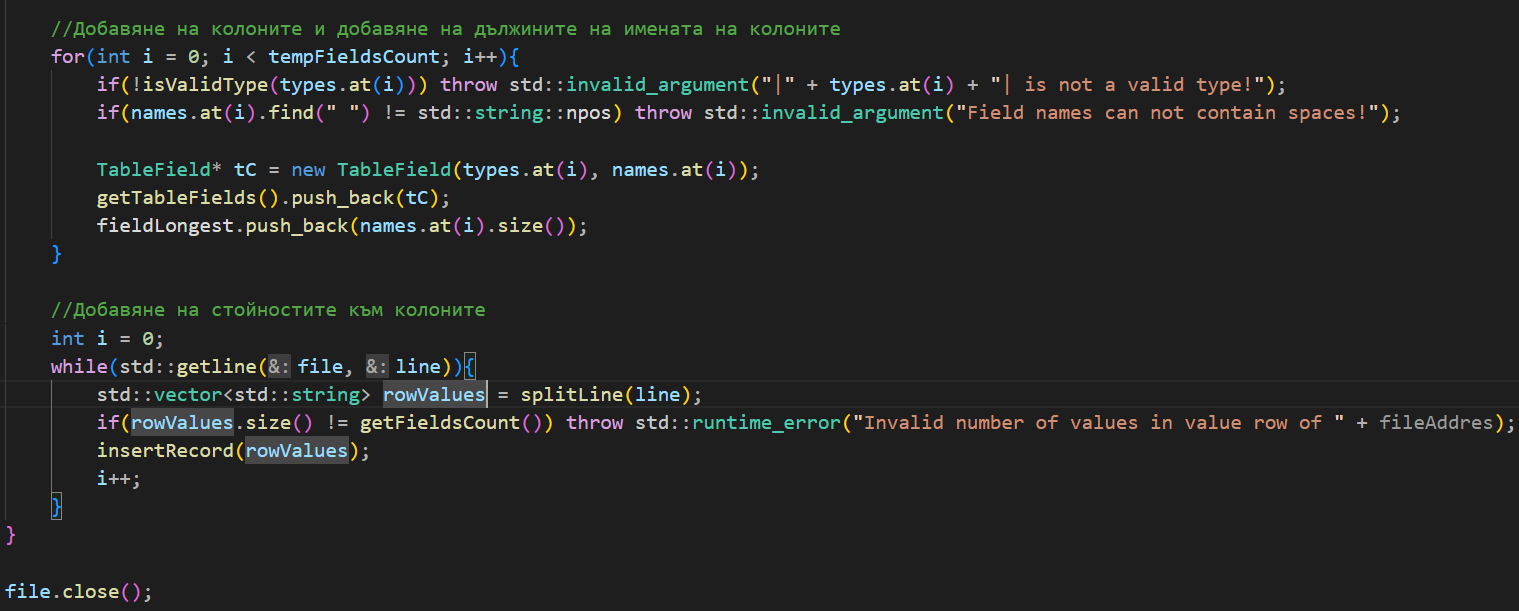
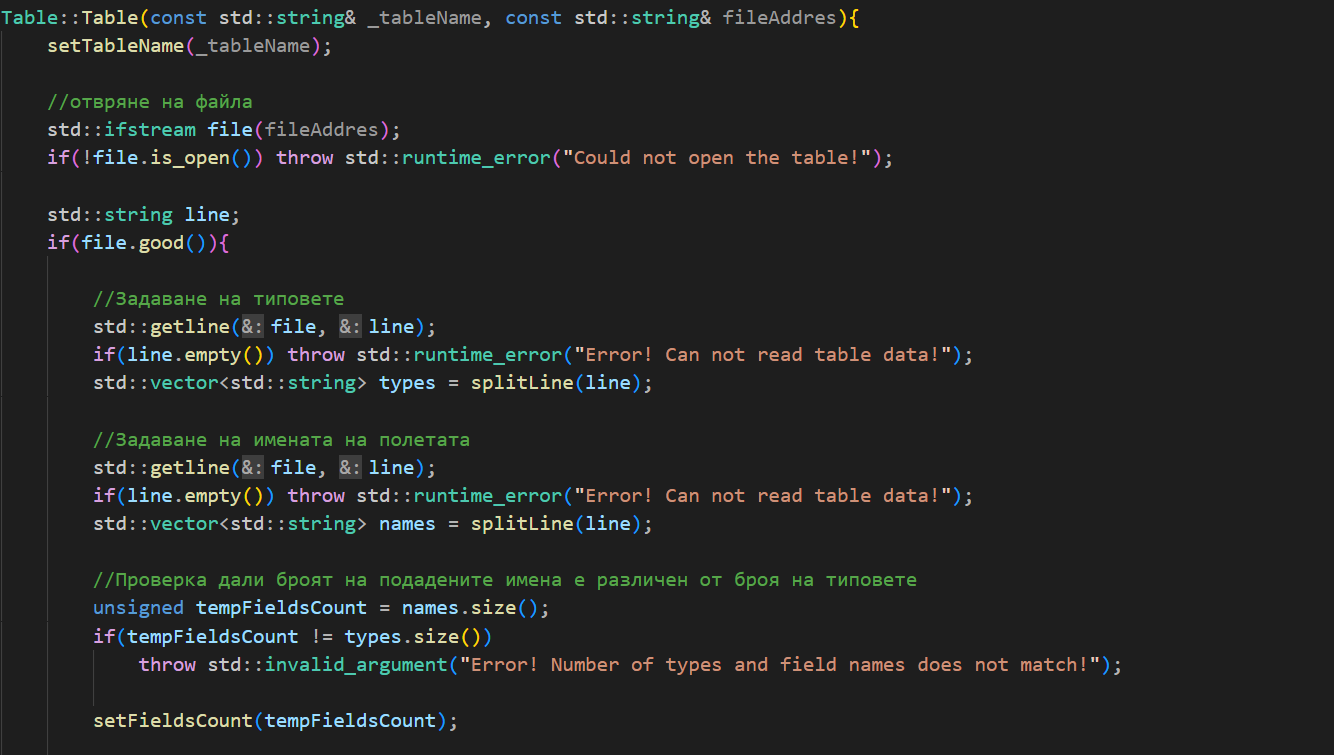


# 

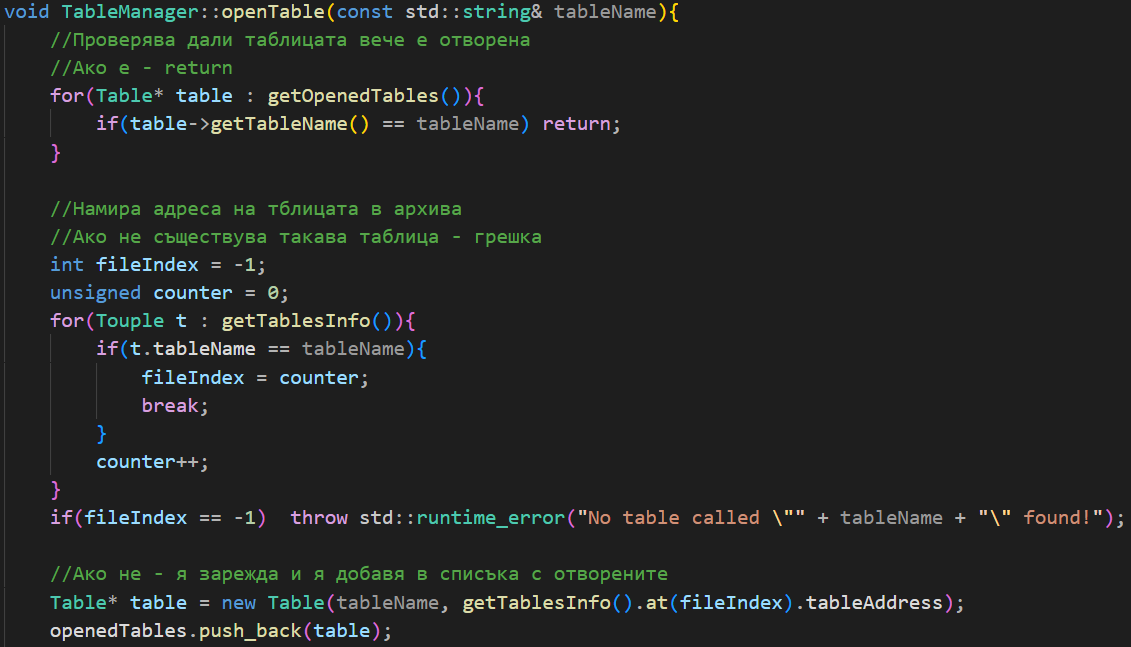
# Реализация

### Реализация на основните класове:

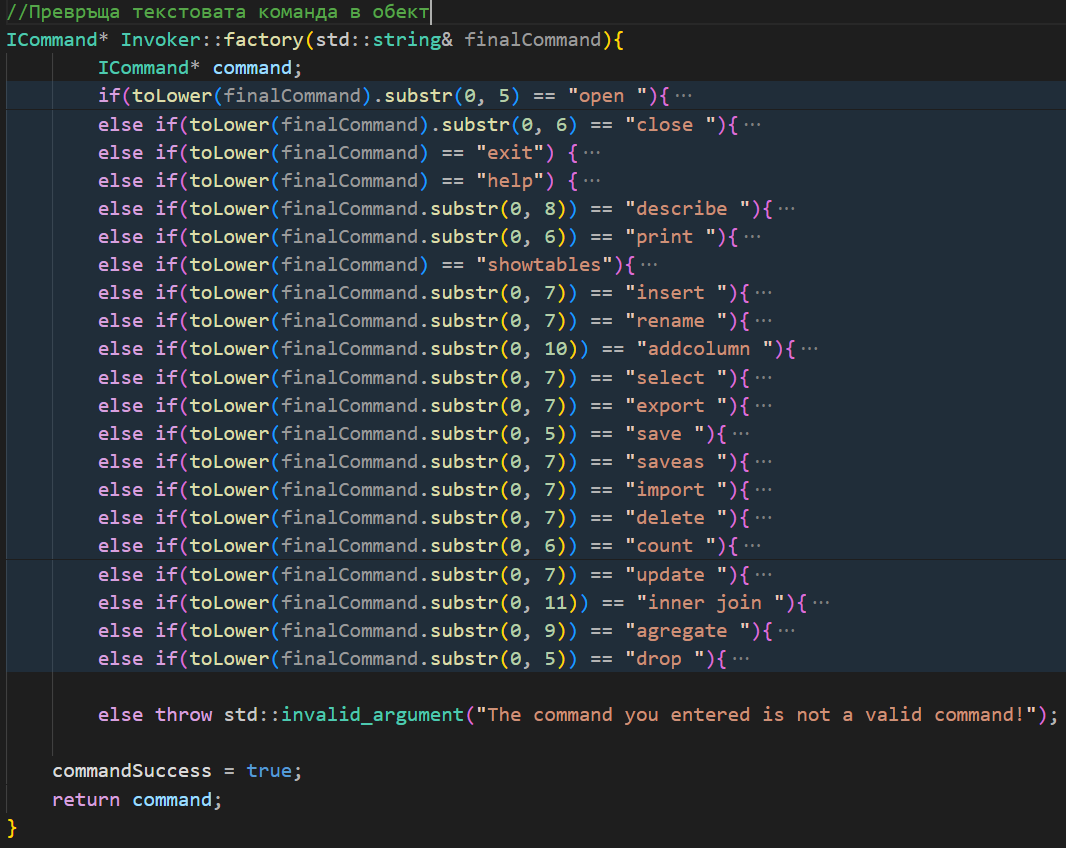
* клас *Table* (създаване – запълване на таблица)



* клас *TableManager* (отваряне на таблица)



* клас *Invoker* (методът factory за създаване на команди)



### Управление на паметта

Основните данни (заредените таблици), както и самите команди при работа на системата се съхраняват в *heap*. При изход от програмата данните в таблиците биват изтрити непосредствено от деструктора на класа *TableManager,* а паметта заета от командите се освобождава при смяна на командата и при изход от програмата.

# 

# Заключение

# Обобщение на изпълнението на началните цели

Изпълнени са основните изисквания за функционалност на системата. Не е изпълнено единствено изискването за създаване на диалогов режим при принтиране на таблици. Добавени са допълнителни удобства за потребителя с цел улеснение на работата със системата.

### Насоки за бъдещо развитие и усъвършенстване

Възможно е в бъдеще системата да предоставя възможност за работа с няколко бази данни вместо само с една. Също така има перспективи за добавяне на нови команди като *sort (order), create table, delete,* и др., създаване на *parser* за по-сложни заявки, различно форматиране на различните типове данни при принтиране и още много други.

Пълния код на решението може да бъде намерен на адрес: https://github.com/MihailTs/DBMS-Project