TabManager

TabManager представлява конзолно приложение, което симулира уеб браузър при работа с табове. Всеки таб съдържа информация за виртуалните интернет страници, съответно URL адрес и съдържание. Виртуалните страници са представени с помощта на текстови файлове, чието име е URL адреса, последван от .txt разширение, а текстовото съдържание на файловете е съдържанието на страниците. Всички файлове с виртуални страници се намират в директорията content. TabManager реализира всяка от следните функционалности:

Реализира се програма с команден интерфейс, в която могат да се въвеждат описаните по-долу команди. За всяка от тях е посочена сложността, с която тя да работи. В описанията на сложностите приемаме, че N е броят на текущо отворените табове.

GO url

Командата зарежда страницата с URL в текущия таб. Автоматично трябва да се обнови timestamp-а на таба. Сложност: O(U+C), където U е дължината на , а C – тази на съдържанието му.

INSERT url

Има два варианта. (1) Ако URL-ът вече е зареден в някой таб на браузъра, този таб става текущ. Ако има няколко такива таба, текущ да стане първият от тях. (2) В противен случай, след текущия таб се добавя нов, в който се зарежда страницата с URL . За този нов таб автоматично се обновява неговия timestamp. Новият таб става текущ. Сложност: O(U+C), където U е дължината на , а C – тази на съдържанието му.

BACK Потребителят се прехвърля на таба, който е непосредствено преди текущия. Ако такъв няма (в момента сме върху първия таб), не се случва нищо. Сложност: O(1)

FORWARD Браузърът отива на таба, който е непосредствено след текущия. Ако такъв няма (в момента сме върху последния таб), не се случва нищо. Сложност: O(1)

REMOVE Премахва текущия таб. Текущ става този след него. Ако такъв няма, текущ става този пред него. Ако това е бил последният таб, отваря се нов и в него се зарежда адрес “about:blank”. Сложност: O(1)

PRINT Извежда на екрана информация за всички табове. Форматът е следния: (Между двата елемента има един интервал) Редът, на който се намира текущия таб, трябва да започва със знак за по-голямо (>), например: somesite.com

123123123 > www.example.com 123456789 Somethingelse.com/index.html 123123321

Табовете да се извеждат точно в реда, в който се пазят в браузъра. Сложност: O(N)

SORT by

Сортира всички табове лексикографски и прави текущ първия в наредбата. може да приема две стойности: * URL – сортира табовете по URL, като ако има два с еднкъв URL, те се подреждат по timestamp; * TIME – сортира табовете по техния timestamp, като ако има два с еднакъв timestamp, те се подреждат по URL. Сложност: Да се използва алгоритъм за сортиране със сложност O(N log N). Крайната сложност на сортирането може да бъде по-голяма, тъй като сравнението на низове не винаги е бърза операция. Крайният резултат трябва да бъде със сложност не по-лоша от O(UNlog N), където U е максималната дължина на URL от тези заредени в браузъра.

SEARCH word

По подадена дума , търси във всички отворени табове и извежда на екрана списък на онези от тях, които съдържат в своето съдържание. Табовете в резултата трябва да се изведат номерирани от 1 до N. След това потребителят трябва да може да избере един от тях и той става текущ. Сложност: Не повече от O(W+ T*log(T)), където W е дължината на , а T е броят на табовете, които я съдържат.

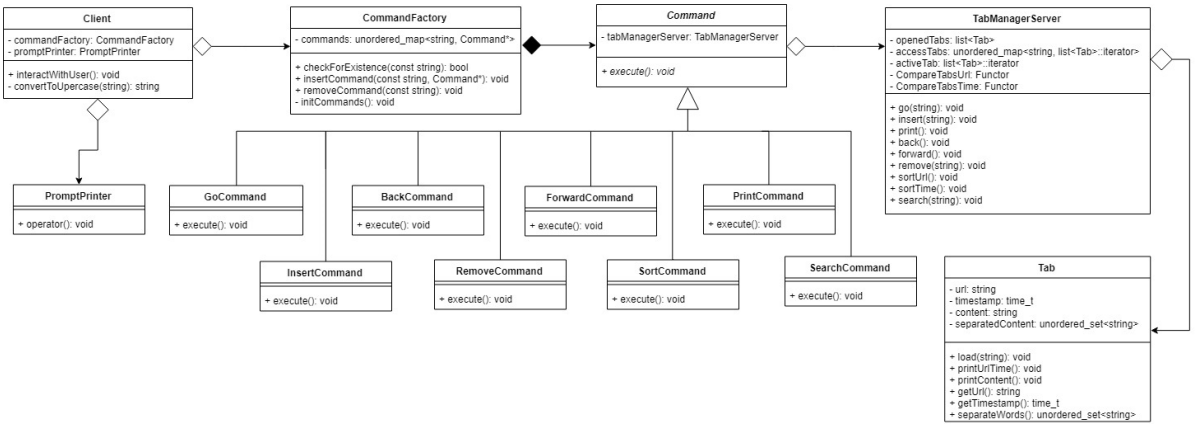
Зареждането на съдържанието на страници трябва да става по следния начин: приемаме, че в работната директория на програмата (обикновено същата, от която я стартираме), ще има поддиректория content, която съдържа текстови файлове. Когато зареждаме URL , програмата ви трябва да опита да зареди файл с име „content/.txt“ и приемаме, че неговото съдържание е съдържанието на страницата. Например ако потребителят реши да зареди “example.com” и след това “ABC”, програмата ви трябва да опита да зареди съдържанието на следните файлове:

content/example.com.txt

content/ABC.txt

Ако при зареждане на даден URL, съответният му файл не бъде намерен или възникне проблем при прочитането му, приемаме, че страницата е празна и съдържанието ѝ ще бъде празният низ.

Архитектура на приложението:



Класове и публични методи:

1. Class Tab

- **Tab() – default constructor:** създава обект от класа Tab с url адрес по подразбиране “about:blank” и timestamp – настоящия момент
- **Tab(string url) –** създава обект от класа Tab с url, аргумента на конструктора и timestamp – настоящия момент
- **void load(string url) –** задава url адреса на обекта на url, методът приема 1 аргумент и не връща нищо
- **void printUrlTime() –** извежда на конзолата url адрес, последван от timestamp, методът не приема аргументи и не връща нищо
- **void printContent() –** извежда на конзолата символен низ, представящ съдържанието на табовете, прочетено от текстовите файлове, методът не приема аргументи и не връща нищо
- **string getUrl() –** връща url адреса на обекта, методът не приема аргументи
- **time_t getTime() –** връща timestamp на обекта от клас Tab, методът не приема аргументи

- **set<string> separateWords() –** връща set от string, съдържащ content на обекта, разделено по думи

2. Class TabManagerServer

- **TabManagerServer() – constructor:** създава обект от класа TabManagerServer съдържащ списък с отворени табове с дължина 1
- **void print () –** извежда на екрана url и timestamp на всички табове, намиращи се в списъка с отворени табове, методът не приема аргументи и не връща нищо
- **void go(string url) –** зарежда в списъка с табове нов таб с url, подаден като аргумент на мястото на текущия активен, методът приема един аргумент и не връща нищо
- **void insert(string url) –** зарежда в списъка с табове нов таб с url, подаден като аргумент, методът приема един аргумент и не връща нищо
- **void back() –** променя текущият активен таб на този преди него в списъка, ако има такъв, методът не приема аргументи и не връща нищо
- **void forward() –** променя текущият активен таб на този след него в списъка, ако има такъв, методът не приема аргументи и не връща нищо
- **void remove() –** премахва текущия активен таб от списъка с отворени табове, методът не приема аргументи и не връща нищо
- **void sortUrl() –** сортира отворените табове в списъка по url адреси лексикографски нарастващо, методът не приема аргументи и не връща нищо
- **void sortTime() –** сортира отворените табове в списъка по timestamp лексикографски нарастващо, методът не приема аргументи и не връща нищо

3. Class Client

- **Client () – constructor:** създава обект от класа Client

- **void interactWithUser() –** представлява главният метод за въвеждане на символни низове от клавиатурата и извеждане на символни низове на конзолата, методът не приема аргументи и не връща нищо

4. Class PromptPrinter

- **void operator() -** предефиниран оператор () за изпълнение без аргументи, идеята е класът да се явява в ролята на функтор

5. Class CommandFactory

- **CommandFactory() -** конструктор по подразбиране за създаване на обекти от клас CommandFactory
- **bool checkCommandExistence(string command)** - проверява дали подаденият стринг като аргумент отговаря на някаква команда в списъка с команди на фабриката, и ако отговаря връща true, ако не – false, методът приема един аргумент и връща стойност от тип bool.
- **void insertCommand(string command) –** добавя нова команда към списъка с команди на фабриката за команди, методът приема един аргумент и не връща нищо
- **bool removeCommand(string command) –** премахва команда от списъка с команди, методът приема един аргумент и връща стойност от тип bool
- **Command* getCommandByType(string command) –** връща указател от клас съответния тип команда, подадена като параметър, методът приема един аргумент и връща стойност от тип bool

6. Class Command

- **Command() -** конструктор по подразбиране за създаване на обекти от клас Command
- **CommandStatus execute()** - чисто виртуален метод към абстрактния клас Command, методът не приема аргументи и връща стойност от тип enum CommandStatus

7. Class BackCommand

- **CommandStatus execute()** - предефиниран метод за изпълнение на командата BackCommand, наследен от абстрактния клас Command

8. Class ForwardCommand

- **CommandStatus execute()** - предефиниран метод за изпълнение на командата ForwardCommand, наследен от абстрактния клас Command

9. Class GoCommand

- **CommandStatus execute()** - предефиниран метод за изпълнение на командата GoCommand, наследен от аб1страктния клас Command

10. Class GoCommand

- **CommandStatus execute()** - предефиниран метод за изпълнение на командата GoCommand, наследен от абстрактния клас Command

11. Class InsertCommand

- **CommandStatus execute()** - предефиниран метод за изпълнение на командата InsertCommand, наследен от абстрактния клас Command

12. Class PrintCommand

- **CommandStatus execute()** - предефиниран метод за изпълнение на командата PrintCommand, наследен от абстрактния клас Command

13. Class RemoveCommand

- **CommandStatus execute()** - предефиниран метод за изпълнение на командата RemoveCommand, наследен от абстрактния клас Command

14. Class SortCommand

- **CommandStatus execute()** - предефиниран метод за изпълнение на командата SortCommand, наследен от абстрактния клас Command

15. Class SearchCommand

- **CommandStatus execute()** - предефиниран метод за изпълнение на командата SearchCommand, наследен от абстрактния клас Command