

**СОФИЙСКИ УНИВЕРСИТЕТ "СВ. КЛИМЕНТ ОХРИДСКИ"**

ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА

Документация на проект

по Обектно ориентирано програмиране

НА ТЕМА:

**Проект 12: Склад**

Изготвил:

Кенан Юсеин

фак. № 71947

спец. Информационни системи, 1 курс

**Глава 1: Увод**

1.1 Описание и идея на проекта:

Проектът „Склад“ реализира информационна система, обслужваща склад на магазин. Програмата съхранява и обработва данните за продуктите в склада.

1.2. Цел и задачи на разработката

Цели: постигане на четим код, отговарящ на ООП стандартите. Програмата трябва да обработва и съхранява продуктите на склада във файл и трябва да дава възможност на потребителя да може да ги достъпва и променя бързо и лесно.

При получаване на нова партида стока, потребителят трябва да може да добавя продуктите в програмата като въвежда съответна информация за съответните продукти.

Архитектурата и обема на склада се задават статично в програмата. Подреждането на стоката в склада трябва да става по специално определен начин от потребителя.

Програмата трябва да следи за срокове на годност, наличност и т.н. и да информира потребителя при необходимост.

Програмата трябва да води „Logs” на всяко едно действие направено от потребителя (добавяне на продукт, премахване, почистване не склада и т.н.), които да могат да бъдат свободно достъпвани от администратор.

1.3 Структура на документацията:

* Глава 1. Увод
* Глава 2. Преглед на предметната област
* Глава 3. Проектиране, Реализация
* Глава 4. Тестване
* Глава 5. Заключение

**Глава 2: Преглед на предметната област**

**Глава 3: Проектиране и Реализация**

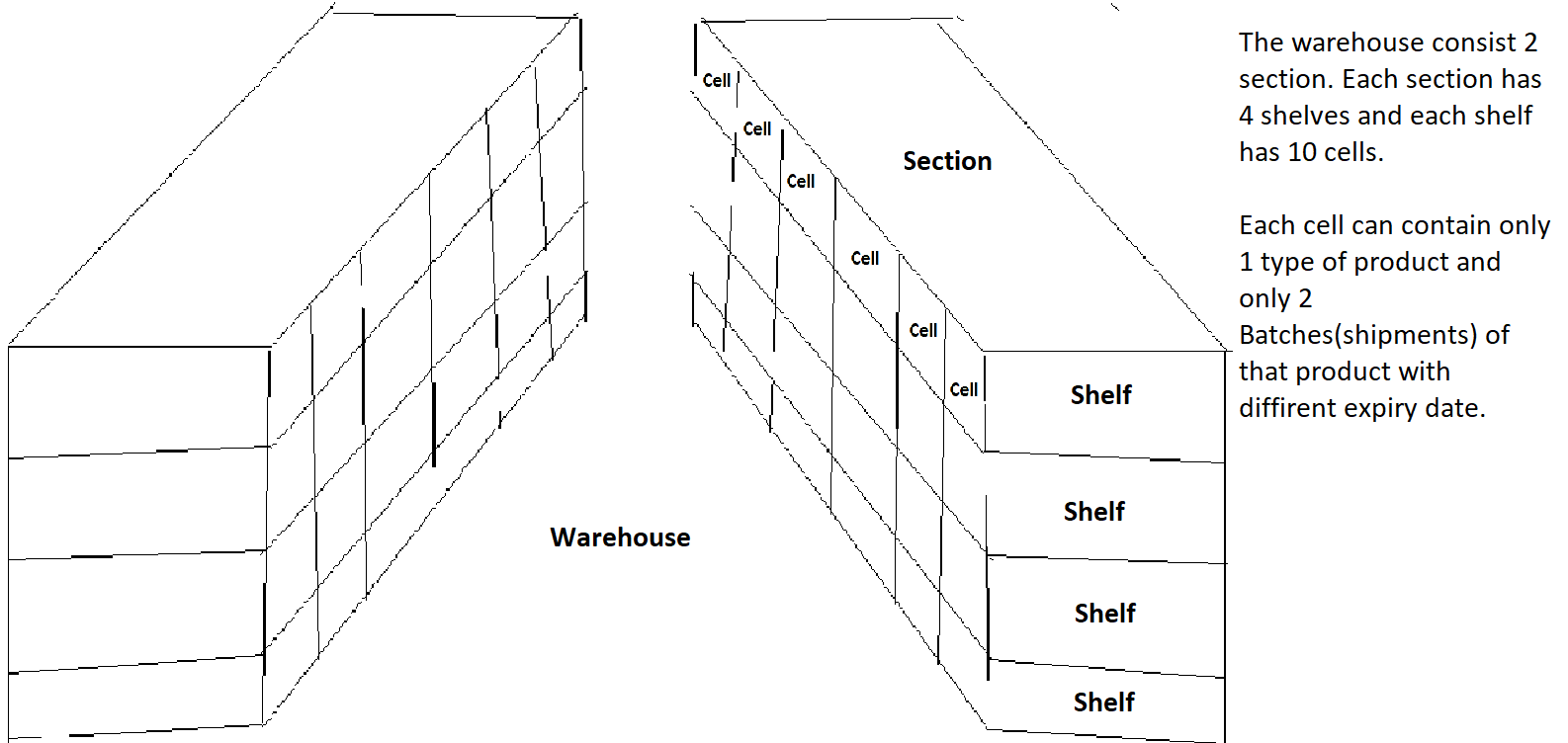
**Класът „Product”** съхранява следната информация за всеки въведен от потребителя продукт:

* име **(class String)**
* тип на **(class enumTypeProduct)**
* срок на годност
* дата на постъпване в склада
* име на производител
* мерна единица (килограми, литри)
* налично количество
* коментар (свободен текст)

Датите в програмата се обработват от клас **“Date”** (yyyy-mm-dd).

**„StorageHouse”** представлява складът на проекта. Складът е статично зададен в програмата. Може да бъде променяно от администратора, но в нашия случай той се състои от:

* 2 Секции. **(struct Section)**
* Всяка секция се състои от по 4 Рафта **(struct Shelf)**
* Всеки рафт се състои от по 10 клетки **(struct Cell)**
* Всяка клетка може да съдържа САМО 1 ТИП продукт и само 2 партиди от съответния продукт (склада не приема повече от съответния продукт преди да се продадат последните 2 партиди)



**Реализация на „StorageHouse”:** Използват се вектори **(class Vector)** :

Vector<Section> warehouse; представлява вектор от секция, който е вектор от рафт, който е вектор от клетки. По този начин се реализира горе описаната архитектура на склада.

int totalCapacity; int nSectors; int nShelves; int nCells; - използват се за създаване на вектора ‘warehouse’ по стандартите на ООП за по лесна промяна на архитектурата на склада.

Реализирани са съответни конструктори за структурите Секция, Рафт, Клетка и за класа и setters and getters.

**Методи на класа:**

* **bool WHisFull();**

проверява дали склада е пълен.

* **int StorageHouse::getTotalWHQuantity() constю**

Връща общия брой на всички стоки в склада (събира quantity на всички продукти заредени във вектора.)

* **void loadWarehouse(Vector<Product>& allProducts);**

allProducts е вектор от всички продукти прочетени от файла в main.cpp, които са подадени при създаването на класа.

Метода зарежда всички продукти от вектора във вектора warehouse на класа, който представлява склада. Следните правила са спазени при зареждането:

* + Продукта се добавя ако вече всички продукти не са добавени и се гледа дали има вече съществуващ продукт и дали са повече от 1 партида, ако ДА продукта се отхвърля от склада до продаване на вече 2те партиди.
  + Всяка клетка приема само едноименни продукти, за осъществяването на което при добавяне на всеки следващ продукт се проверява дали вече съществува едноименен продукт и ако да то се добавя при него в същата клетка или същия рафт.
* **void addProduct(Product& product);**
  + Добавя нов продукт във вектора на класа по същите принципи, които се спазват при зареждане на склада.
  + Преди да се добави нов продукт първо се проверява дали има място в склада.
  + Ако вече има 2 партиди от този продукт добавянето се отменя до продаване на вече съществуващите продукти с цел ограничаване на ненужни продукти, на които би изтекъл срока на годност преди продаването на по-старите партиди.
* **void removeProduct(………);**

Премахва определен брой от съответен продукт от склада и извежда съобщение за партида и разположение на премахнатия продукт в склада.

* + Ако потребителя иска да премахне повече на брой от съществуващия в склада метода дава избор на потребителя дали да премахни всички останали или не.
  + Метода първо премахва тази партида, която е по-стара (срока на годност изтича по-рано)
* void print();

Принтира на конзолата всички налични продукти в склада и извежда подбрана информация за всяка от тях.

Извежда се и разположението на всеки продукт в склада.

* void clean(Vector<Product>& allProducts, const Date& current);

Изчиства склада от всички стоки, на които им е изтекъл срокът на годност. Извежда съответна информация за премахнатите продукти от склада.

**Warehouse.cpp (main.cpp) – Потребителски интерфейс**

Глобални променливи:

* Vector<Product> allProducts;

Използва се за съхранение на всички продукти от Product, които се четат от файл и се зареждат във вектора на класа StorageHouse по-късно.

* StorageHouse warehouse;

Декларация на класа склад.

* String filename;

Следи за името на отворения файл с продукти.

* Date currentDate;

Програмата започва с въвеждане на текущата дата, която се използва за следене на срокове на годност.

* Vector<String> logs;

Използва се за записване на всички действия на потребителя в съответен файл.

Потребителски интерфейс:

Използват се 2 менюта в програмата.

**Menu():** Това е главното меню на програмата, чрез който се работи с файловете. Реализирани са следните функционалности:

void Open(); void Close(); void Save(); voiSaveAs(); void Help(); void exit();

За следена дали вече е отворен/зареден файл се използва string-a filenamе. При затваряне или запаметяване на файл (освобождаване на паметта на програмата) стринга се нулира, което означава, че няма отворен файл.

**ProdManagement();** – Това е функционалното меню на програмата, чрез който потребителят работи със склада:

void Print(); void Add(Product & product);

void Remove(const String & rName, int rQuantity);

void Clean();

**void getWHinfo()** се използва за извеждане на най-важната информация за склада след всяко действие направено от потребителя.

Записват във файлове / четат от файлове:

void fileWrite(std::ofstream& output);

void fileRead(std::ifstream& input);

void readLogs();

**Глава 4: Тестване**

За тестване на програмата се използва products.txt, в който са въведени примерни продукти.

**Глава 5: Заключение**