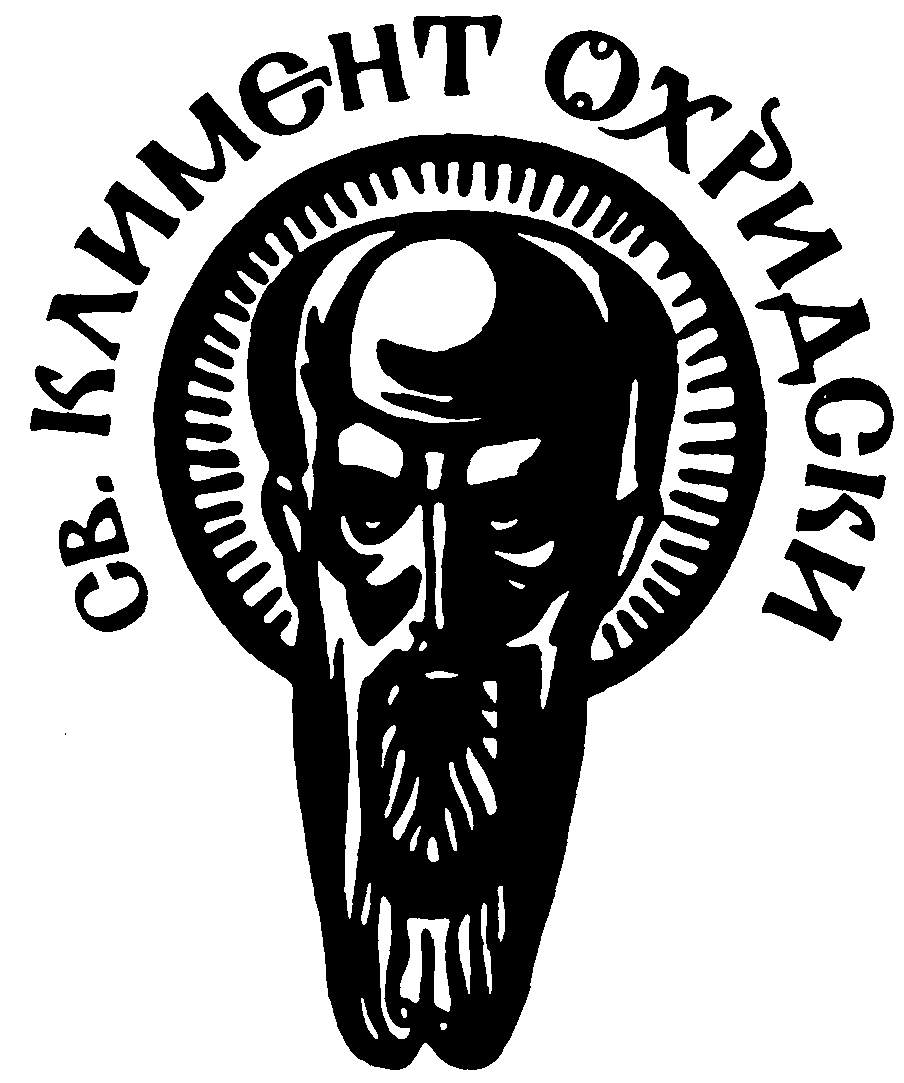
**Софийски университет „Св. Климент Охридски”**

**Факултет по математика и информатика**

****

**Обектно – ориентирано програмиране**

**Курсов проект на тема:**

**„Склад“**

**Изготвила: Весела Илиянова Стоянова**

**Ф.Н. 71949**

**I курс Бакалавър**

**Специалност: Информационни системи**

**Ръководител: Калин Георгиев**

**07.06.2020 г.**

**Съдържание:**

[1. Увод: 3](#_Toc42271923)

[1.1. Описание и идея на проекта: 3](#_Toc42271924)

[1.2. Цел и задачи на разработката: 3](#_Toc42271925)

[1.3. Структура на документацията: 3](#_Toc42271926)

[2. Преглед на предметната област: 3](#_Toc42271927)

[2.1. Основни дефиниции, концепции и алгоритми, които ще бъдат използвани: 3](#_Toc42271928)

[2.2. Дефиниране на проблеми и сложност на поставената задача: 4](#_Toc42271929)

[2.3. Подходи , методи (евентуално модели и стандарти) за решаване на поставените проблеми: 4](#_Toc42271930)

[2.4. Потребителски (функционални) изисквания (права, роли, статуси, диаграми) и качествени (нефункционални) изисквания (скалируемост, поддръжка): 4](#_Toc42271931)

[3. Проектиране: 5](#_Toc42271932)

[3.1. Общата архитектура на проекта е представена на фиг. 3: 5](#_Toc42271933)

[3.2. Диаграми (на структура и поведение – по обекти, слоеве с най – важните извадки от кода): 6](#_Toc42271934)

[4. Реализация, тестване: 6](#_Toc42271935)

[4.1. Реализация на класове: 6](#_Toc42271936)

[4.2. Планиране, описание и създаване на тестови сценарии: 10](#_Toc42271937)

[5. Заключение: 11](#_Toc42271938)

[5.1. Обобщение на изпълнението на началните цели: 11](#_Toc42271939)

[5.2. Насоки за бъдещо развитие и усъвършенстване: 11](#_Toc42271940)

[Връзка към хранилище в Github: 11](#_Toc42271941)

**Таблици и графики:**

[Фиг.1: снимка на екрана при стартиране на програмата 4](#_Toc42272004)

[Фиг.2: снимка на екрана при въвеждане на различни команди 5](#_Toc42272005)

[Фиг.3: Реализираните класове заедно с техните член-данни и методи 5](#_Toc42272006)

[Фиг.4: Header файловете и Source файловете в проекта 6](#_Toc42272007)

[Фиг.5: Методи и член-данни на класа AuditStatement 7](#_Toc42272008)

[Фиг.6: Методи и член-данни на класа ISODate 8](#_Toc42272009)

[Фиг.7: Методи и член-данни на класа Location 8](#_Toc42272010)

[Фиг.8: Методи и член-данни на класа Product 9](#_Toc42272011)

[Фиг.9: Методи и член-данни на класа Space 9](#_Toc42272012)

[Фиг.10: Методи и член-данни на класа Store 10](#_Toc42272013)

# **Увод:**

## **Описание и идея на проекта:**

В рамките на зададения проект трябва да се реализира програма, реализираща информационна система, обслужваща склад. Програмата съхранява и обработва данните за наличността в склада във файл.

## **Цел и задачи на разработката:**

Основната цел е програмата да бъде достатъчно лесна и удобна за използване. За целта в проекта е реализиран *CommandsExecutor*, който съдържа метод *showStartMenu*. В *main()* функцията на програмата ще можем да стартираме програмата само с извикването на този метод. Програмата ще изисква от потребителя да въведе някоя от следните функции*: open, help* или *exit*, с помощта на която да определи коя функция да бъде изпълнена. При въвеждането на невалидни данни (данни, различни от *open, help, exit*) на стандартния изход ще бъдат изведени отново възможните функции. Всички промени, които се правят, трябва да бъдат записвани във файл. Когато бъде записан нов файл, изпълнението на програмата продължава и се извежда съобщение за успешното записване на данните във файл. Програмата завършва, когато бъде въведена функцията *exit*.

## **Структура на документацията:**

Структурата на документацията се състои в няколко опорни точки. Първоначално се представят належащите проблеми, както и търсеното на достатъчно ефективни решения. След това се преминава към проектиране на проекта и накрая се преминава към реализация и тестване на проекта. Накрая се представят насоки за бъдещо развитие и усъвършенстване.

# **Преглед на предметната област:**

## **Основни дефиниции, концепции и алгоритми, които ще бъдат използвани:**

* Използвана е определена организация на пространството в склада. Той е разделен на 2 части като първата половина е за продукти, чиято мерна единица е килограм, а другата половина е за продукти, чиято мерна единица е литър. Създадени са константите

*const int SECTIONS\_COUNT = 4;*

*const int SHELVES\_COUNT = 6;*

*const int NUMBERS\_COUNT = 15;*

*const int PRODUCTS\_NUMBER = 30*;

като всяко местоположение на продукт се характеризира със секция, рафт и номер. В склада има 4 секции като на всяка секция и 6 рафта, на които има по 15 номера като на всеки номер могат да се поберат по 30 продукта.

* Проектът работи с дати, които са конструирани в *ISO 8601* формат, който представлява международен [стандарт](https://bg.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82) за обмен на [дати](https://bg.wikipedia.org/wiki/%D0%94%D0%B0%D1%82%D0%B0) и данни за [време](https://bg.wikipedia.org/wiki/%D0%92%D1%80%D0%B5%D0%BC%D0%B5). Стойностите за дата и време се подреждат от най-значителната към най-незначителната: година, месец, ден, час, минута, секунда и част от секундата. В проекта датите са представени само с година, месец и ден. В класът *ISODate* е представен метод *constructDate*, който конструира дата в *ISO 8601* формат от *string*. За повече информация <https://bg.wikipedia.org/wiki/ISO_8601>.
* Някои от функциите, дефинирани в *CommandsExecutor* са разделени на две части в зависомост от това кога се извикват. Такива функции са help() и *showMenu()*. Функцията help е разделена на *showHelp()* и *showAdvancedHelp()* като *showHelp()* се извиква, ако при стартиране на програмата, потребителят иска да види кратка информация за командите, a *showAdvancedHelp()* се извиква, когато вече е зареден един файл и потребителят иска да провери информация за командите. Функцията *showMenu()* е разделена на *showStartMenu()* и *showAdvancedMenu()*, като *showMenu()* се извиква за първоначалното меню, а *showAdvancedMenu()* се извиква, когато вече е зареден файл.

## **Дефиниране на проблеми и сложност на поставената задача:**

Има множество проблеми, които трябва да бъдат преодоляни с цел успешната реализация на проекта. Те включват гарантирането на удобен и лесен за използване интерфейс. Друг проблем е, че въведените от потребителя данни, трябва да бъдат валидирани. Друг проблем е свързан с функционалността на проекта. Въведените от потребителя данни, освен че трябва да бъдат валидирани, то трябва да бъдат записвани и във файл, който да бъде достъпен и след приключване на програмата. Също така проектът работи с дати, които трябва да бъдат валидирани в специален формат.

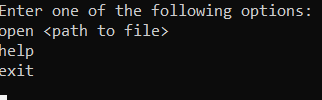
## **Подходи , методи (евентуално модели и стандарти) за решаване на поставените проблеми:**

Начините за разрешаване на поставените проблеми включват използване на различни методи. Всеки метод ще се грижи за коректността на изпълнението, както и изхода на програмата.

## **Потребителски (функционални) изисквания (права, роли, статуси, диаграми) и качествени (нефункционални) изисквания (скалируемост, поддръжка):**

Функционалността започва от *CommandsExecutor*, където се въвежда команда от потребителя, тя се обработва, разпознава коя е командата и я изпълнява.

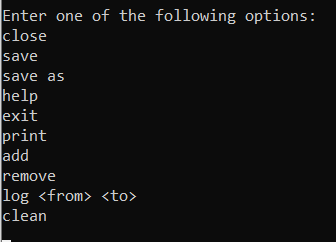
При отваряне на проекта, потребителят има 3 възможности – да отвори файл, да види кратка информация за възможните потребителски команди и да излезе от програмата.



Фиг.1: снимка на екрана при стартиране на програмата

При отваряне на вече съществуващ файл или започване на нов, потребителят има възможност да:

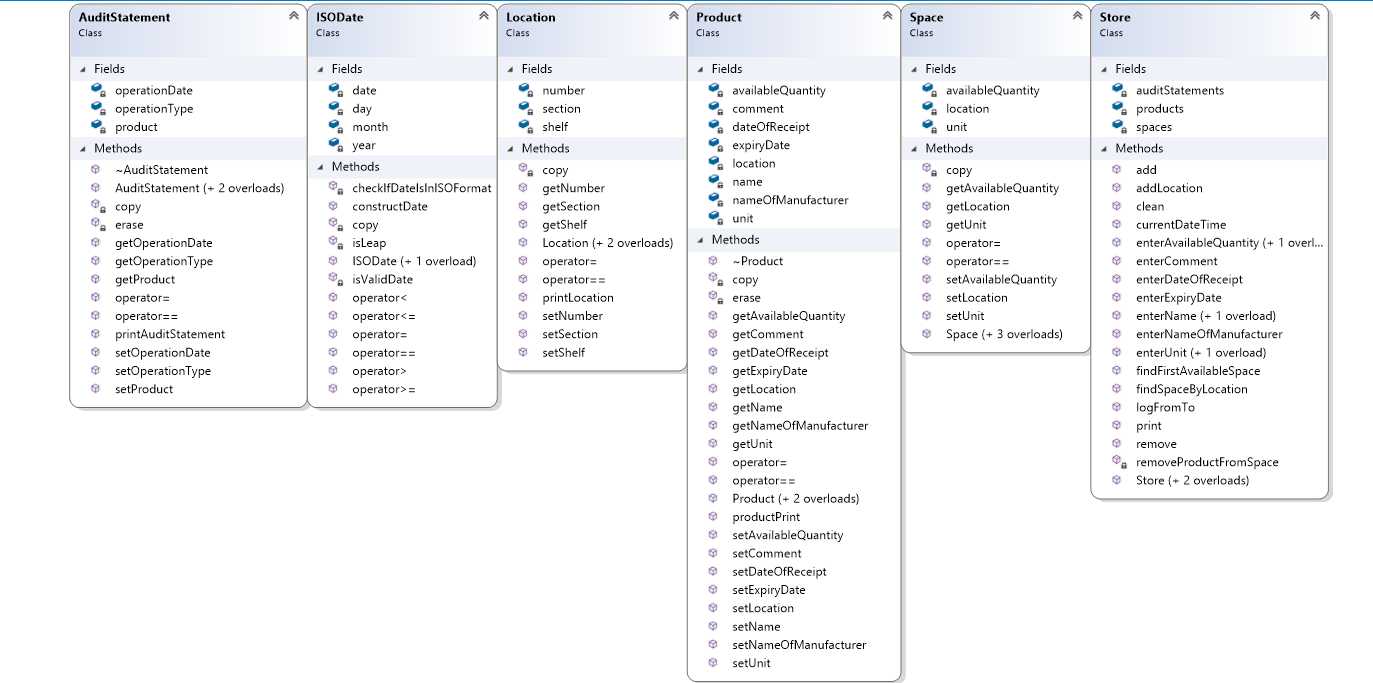
* затвори файла(*close*);
* да го запази със същото име(*save*);
* да го запази с ново име(*save as*);
* да види кратка информация за възможните потребителски команди(*help*);
* да излезне от програмата(*exit*);
* да види информация за наличните продукти в склада(*print*);
* да добави нов продукт(*add*);
* да премахне продукт(*remove*);
* да получи справка за всички промени в наличността в даден период(*log<from><to>*);
* да разчисти склада от всички стоки, на които е изтекъл срокът на годност(*clean*).



Фиг.2: снимка на екрана при въвеждане на различни команди

# **Проектиране:**

## **Общата архитектура на проекта е представена на фиг. 3:**



Фиг.3: Реализираните класове заедно с техните член-данни и методи

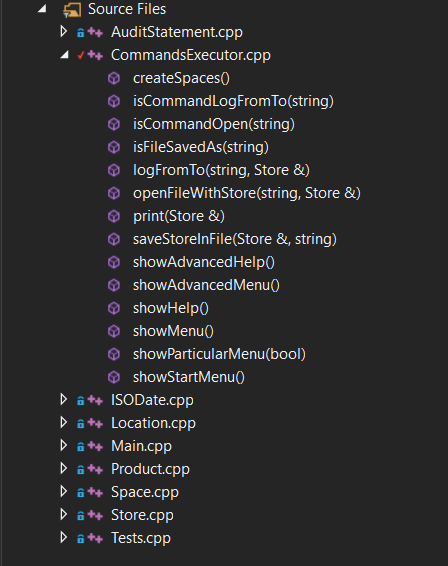
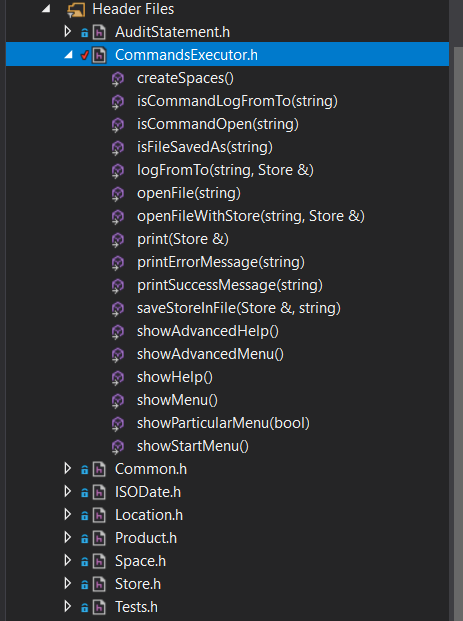
Обектно – ориентираният дизайн се състои в реализацията на файла *CommandsExecutor*, където се намират основните операции за реализацията на този проект. Това са операциите *print, add, remove, log <from> <to>* и *clean*, в допълнение на общите операции *open, close, save, save as, help, exit*.

## **Диаграми (на структура и поведение – по обекти, слоеве с най – важните извадки от кода):**

# **Реализация, тестване:**

## **Реализация на класове:**

Използвано е разделно компилиране за по – добра организация и четимост – Фиг. 4:

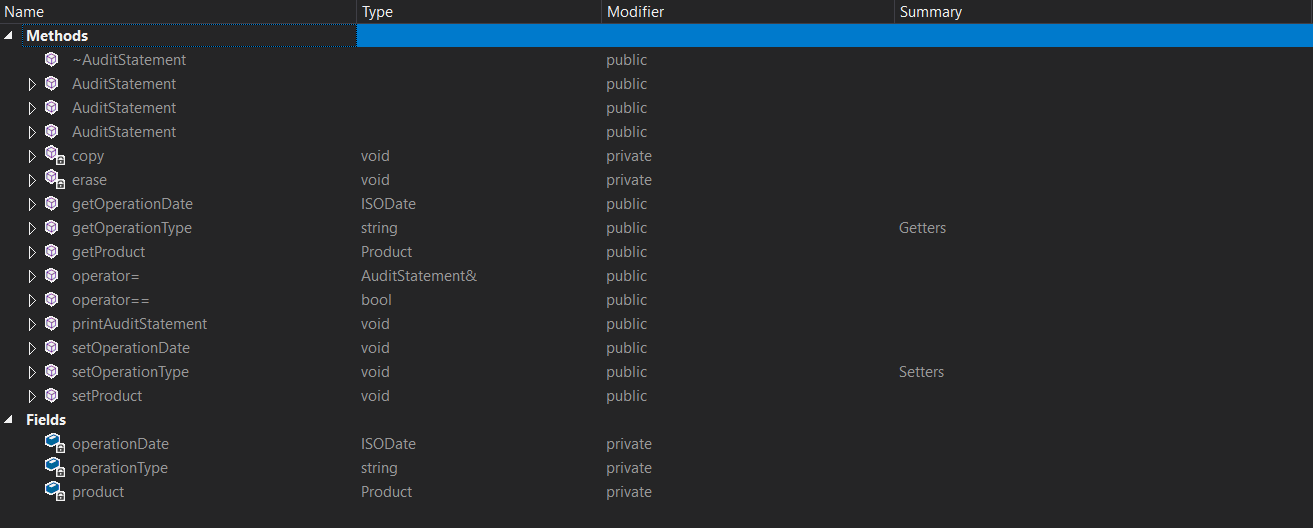


Фиг.4: *Header* файловете и *Source* файловете в проекта

В проекта са реализирани множество от класове, които са представени чрез голямата четворка или в т.н. канонична форма. Също така са реализирани помощни класове и функции за оптимизация и по-добра четимост. Използвани са възможно най-малко глобални променливи и е наблегнато на локалните променливи. Осъществена е връзка между отделните класове.

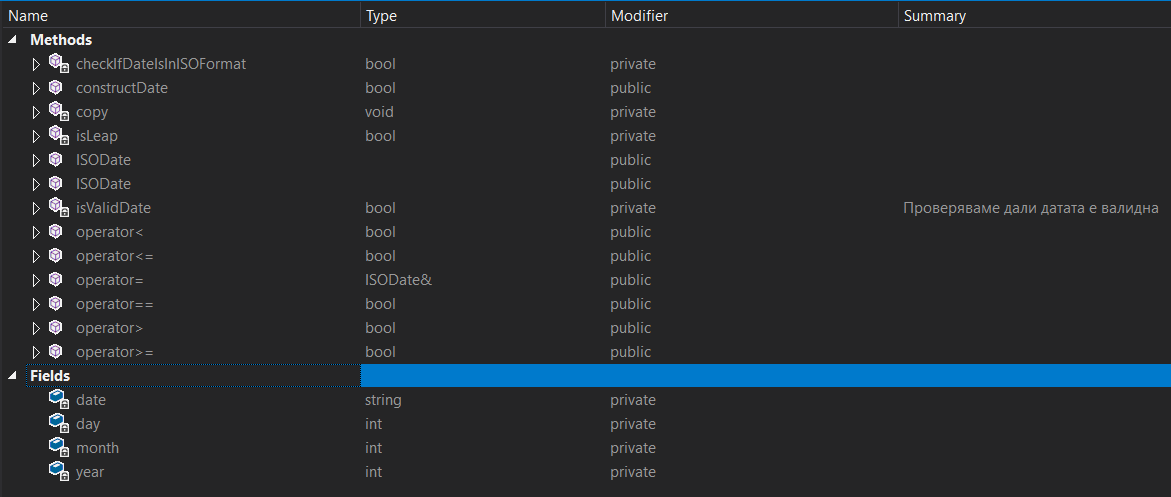
Функцията *open<path>* създава нов файл, ако не съществува файл с име *<path>* или отваря вече съществуващ такъв. При отваряне на програмата и въвеждане на функцията *help* се извиква функцията *showHelp()*, която дава информация за функциите *open, help* и *exit*. При въвеждане на функцията *exit* се излиза от програмата. При отваряне на нов файл потребителят има няколко възможности – *close, save, save as, help, exit, print, add, remove, log <from> <to>* и *clean*.

Класът *AuditStatement* е създаден с цел в него да се записват данните при добавяне и премахване на продукт. Този клас се използва главно за функцията *log<from><to>,* тъй като за нея трябва да бъдат записани данните на всички продукти, които са добавени или премахнати. Този клас съдържа методите и член-данните от фигура 5:



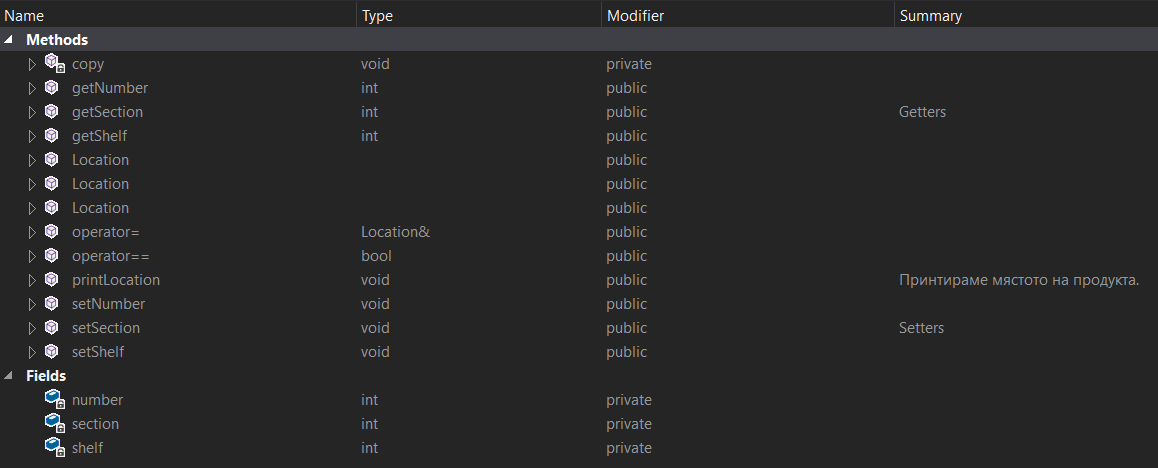
Фиг.5: Методи и член-данни на класа *AuditStatement*

Класът *ISODate* е създаден с цел конструиране на дати в *ISO 8601* формат. Този клас съдържа методите и член-данните от фигура 6:



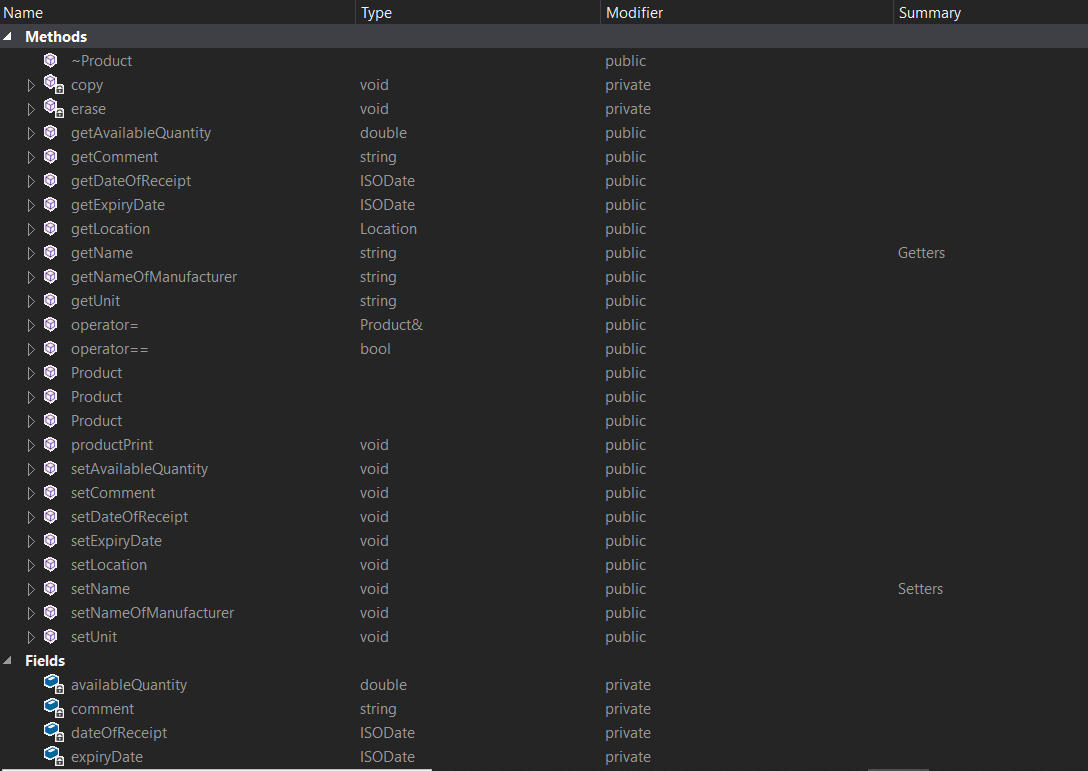
Фиг.6: Методи и член-данни на класа *ISODate*

Класът *Location* е създаден с цел конструиране на местоположение на продукта. Този клас съдържа член-данните и методите от фигура 7:



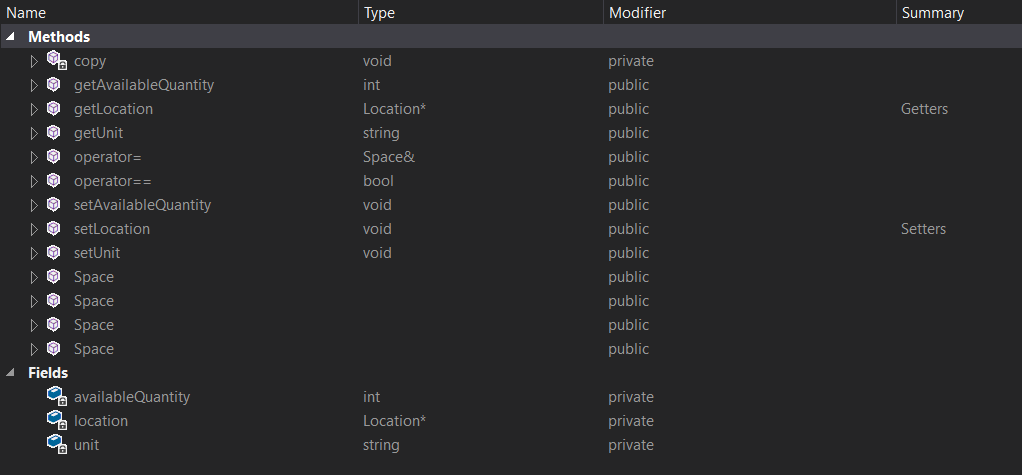
Фиг.7: Методи и член-данни на класа *Location*

Класът *Product* е създаден с цел да съдържа всички характеристики на продукта. Неговите член-данни и методи са описани във фигура 8:



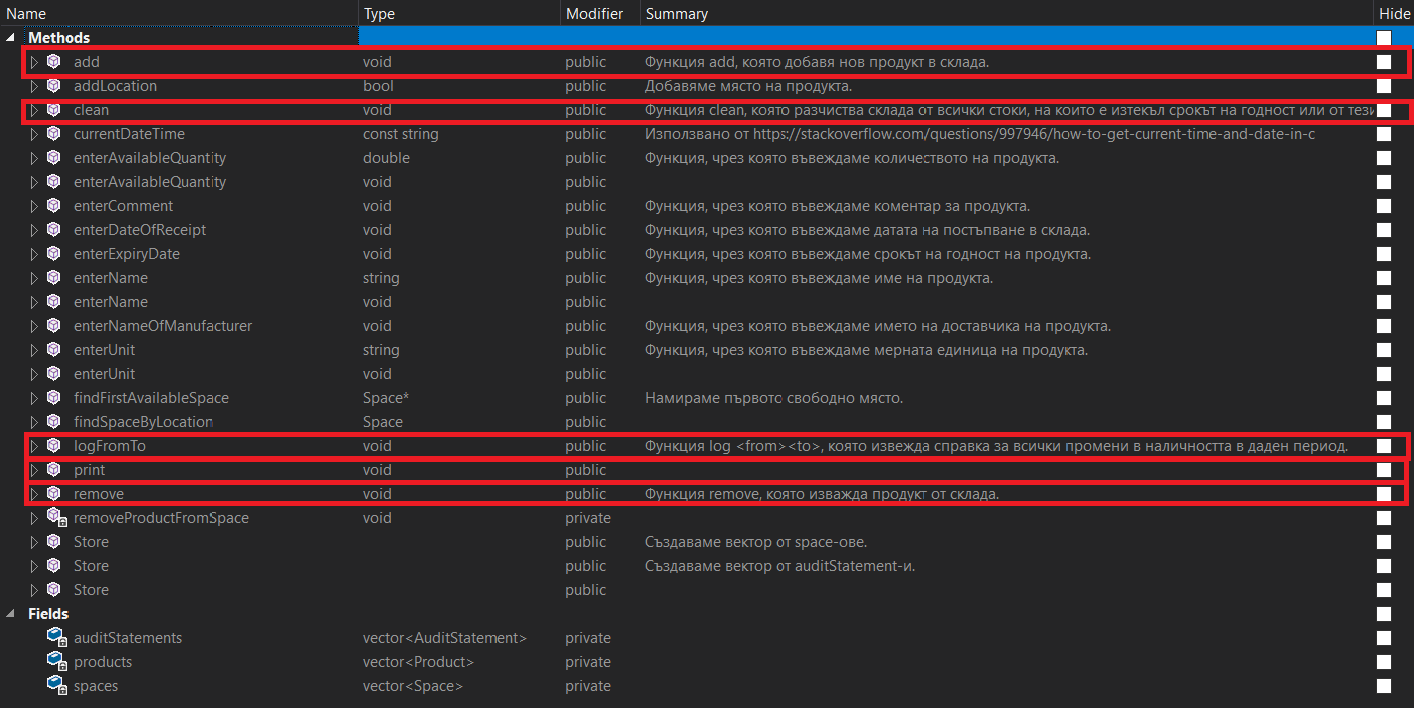
Фиг.8: Методи и член-данни на класа *Product*

Класът *Space* е създаден с цел в него да се пази информация за мястото на продукта. Той използва местоположението, мерната единица и количеството на продукта, за да продуктите да бъдат подредени в склада. На фигура 9 са показани методите и член-данните на този клас.



Фиг.9: Методи и член-данни на класа *Space*

Класът *Store* е основен за този проект. В него са реализирани функциите *print, add, remove, log<from><to>, clean*. Той използва голяма част от методите и член-данните на останалите класове. На фигура 10 са изобразени методите и член-данните на класа *Store*.



Фиг.10: Методи и член-данни на класа *Store*

## **Планиране, описание и създаване на тестови сценарии:**

Първите стъпки от тестването на проекта включват тестване на конструкторите на всички класове. След това бяха тествани общите команди – *open, save, save as, close, help, exit*. Следващите стъпки от тестването са свързани със създаване на склад, който съдържа множество продукти, върху които се извършват различни операции. Първоначално, чрез функцията *add*, която добавя нов продукт, бяха добавени няколко продукта с различно име, срок на годност и мерна единица, както и няколко продукта с еднакви характеристики. Бяха въведени продукти, както с изтекъл, така и с неизтекъл срок на годност. След това беше извикана функцията *remove*, която изважда продукт от склада. Беше въведен продукт, който съществува в склада и такъв, който не съществуваше, за да бъде проверено поведението на програмата в различни ситуации. След това беше извикана фунцкията *log<from><to>*, която извежда справка за всички промени в наличността за определен период от време. Бяха въведени валидни и невалидни периоди – например когато датата *<to>* е по-малко от датата *<from>.* Например *log 2019-02-02 2018-05-05*. След това бяха въведени валидни периоди като бяха въвеждани периоди, в които няма нито една промяна в наличността, както и такива с няколко промени. След това беше извикана функцията *clean*, която разчиства склада от всички стоки с изтекъл срок на годност. В склада има продукти с изтекъл срок, които бяха премахнати след извикването на тази функция. Бяха създадени няколко броя склада, като на всеки бяха тествани всички функции.

# **Заключение:**

## **Обобщение на изпълнението на началните цели:**

Проектът е завършен успешно. Интерфейсът е достатъчно функционален и същевременно лесен за използване и удобен.

## **Насоки за бъдещо развитие и усъвършенстване:**

Част от бъдещите планове за развитие на този проект вкючват подобряване на функционалността на проекта, което ще бъде свързано с добавяне на нови характеристики на продуктите – например цена. Също така валидиране на входните данни - за всички продукти с мерна единица литър да не може да бъде въведена мерна единица килограм. Друга идея за бъдещо развитие на този проект включва изчисляване на загубите от продуктите, чийто срок на годност е изтекъл. Също така може да бъдат въведени и мерни единици грам и милилитър.

# Връзка към хранилище в *Github*:

<https://github.com/VeselaStoyanova>