**Тема на програмата**: “CLI игра - симулация на изследователска дейност на отдалечена планета”

Програмен език: **C++**

OS: **Linux**

Source код: <https://github.com/p-dim-popov/SpaceExploration-CLI_Game>

**Увод**

Идеята ми за този проект е да се опитам да направя работещо приложение, което е максимално разделено на модули. Всеки един от които може да бъде заменен с друг използвайки наследяване на интерфейсните класове, които изграждат приложението. Всеки клас на това приложение ще използва интерфейси като това ще позволи пълното използване на полиморфизма, абстракцията и наследяването, които езикът C++ ни предоставя. Където е възможно ще бъдат спазени и S.O.L.I.D. принципите на обектно ориентираното програмиране.

1. **Ще разгледам гореспоменатите S.O.L.I.D. принципи:**

* Single responsibility principle (принцип на единичната отговорност) - в ООП Single Responsibility Principle гласи, че всеки модул и/или клас трябва да има отговорност върху точно една част от функционалността, която софтуерът предлага
* Open/closed principle (принцип на отворената затвореност) - в ООП open/closed principle гласи, че всички софтуерни части (класове, модули, функции и т.н.) трябва да бъдат отворени за разширяване, но затворени за модификации. Това се осъществява с помощта на гореспоменатото използване на полиморфизъм, абстракция и наследяване. Декларират се интерфейси, които гарантират на програмата, че всеки техен наследник ще реализира декларираните методи и прочие.
* Liskov substitution principle - (принцип за субституцията на Лисков) - този принцип е един от най-трудните за първоначално осмисляне. В ООП принципът за субституция на Лисков гласи, че ако Y е подтип на X, то всички обекти от типа X могат да бъдат заменени с обекти от типа Y без да променят верността на дадената програма. Това може да се сложи в математически вид по следния начин:
  + *- Нека f(x) е свойство, доказуемо за обекти x от типа X, тогава f(y) е вярно за обекти y от тип Y, където Y е подтип на X.*
* Interface segregation principle (принцип за сегрегация на интерфейсите) - този принцип е лесен за разбиране. В ООП interface segregation principle гласи, че нито един клиент не трябва да е принуден да зависи от методи, които не ползва. По-точно - да се избягва добавянето на нова функционалност на съществуващ интерфейс. При такава необходимост се създава нов интерфейс и се налага наследяване на този интерфейс от класовете наследници. В С++, за разлика от С#, където не се позволява множествено наследяване (тази необходимост е премахната, тъй като се осъществява множествено наследяване на интерфейсни класове), тук един клас може да наследява много други класове.
* Dependency inversion principle (принцип за обратната зависимост) - в ООП идеята на dependency inversion principle е да се осъществи това “разделяне” на софтуерните модули. Този принцип гласи следните неща:
  + Модули от високо ниво не трябва да зависят от модули на ниско ниво. И двата вида трябва да зависят на абстракции.
  + Абстракциите не трябва да зависят на детайли. Детайлите трябва да зависят на абстракции

Осъществяването на този принцип се прави чрез така нареченото “инжектиране на зависимости” като всички зависимости на даден клас се подават като параметри към неговият конструктор.

Реализацията на тези 5 принципа наречени S.O.L.I.D. (в превод “стабилност”) позволяват на едно приложение да бъде преизползваемо, лесно поддържано, скалируемо и лесно тестващо се.

**Сюжет**

Главният герой сте вие и целта е да съберете екип от космонавти, за да се изпратят в космоса. Добавят се в космически кораб, добавят се планети за изследване и предмети на всяка планета. Задава се планета за изследване и се започва със събиране на предмети. Някои астронавти могат да умрат, други могат да оцелеят. Космосът е опасно място. Когато главният герой реши, може да пенсионира астронавт, който е рискувал живота си достатъчно в името на родината.

Започва се със събиране на екип от астронавти, спадащи към следните категории:

* Биолог - начални единици кислород: 70
* Геодезист - начални единици кислород: 65
* Метеоролог - начални единици кислород: 90

Командата, чрез която се добавят астронавти е:

|  |
| --- |
| ДобавиАстронавт <длъжност/категория> <име на астронавт> |

Всички астронавти притежават по една раница, която побира до 3 предмета.

Когато астронавт открие предмет, неговото количество кислород намалява с 10 единици. Благодарение на тайно откритие на биолозите, техният кислород намалява само с 2 единици при намерен предмет.

Въвеждат се планети и техните предмети, които астронавтите ще трябва да намерят и съберат по време на експедицията.

Командата, чрез която се добавят планети е:

|  |
| --- |
| ДобавиПланета <име на планета> [предмет\_1,предмет\_2, … предмет\_X] |

След това се посочва планета за изследване чрез командата:

|  |
| --- |
| ИзследвайПланета <име на планета> |

Избират се астронавти, годни за експедиция, такива които имат над 20 единици кислород и имат място в раницата си. Когато експедицията на дадена планета започне, астронавтите се приземяват и започват да търсят предмети един по един. Редът се определя случайно. При намиране на елемент, кислородът на дадения астронавт намалява, като има 33% вероятност, поради затруднения при взимането на предмета от планетата, астронавтът да изхаби двойно повече кислородни единици. Действието продължава докато или на планетата има неоткрити предмети, или кислородът на всички астронавти се изчерпа, или нямат повече място в раниците си. След това оцелелите астронавти се прибират.

Поради опасностите на открития космос има 50% вероятност всички астронавти, изпратени на дадена експедиция, да загинат или изчезнат.

Всеки един астронавт може да бъде пенсиониран и изпратен обратно в дома си по всяко време с командата:

|  |
| --- |
| ПенсионирайАстронавт <име на астронавт> |

Редът на действията, оказан дотук, е само примерен. Играта има механизми за справяне с грешен ред на действията чрез известяване на потребителя или игнориране на грешни команди.

**По-подробно описание на командите**: Играта се състои от няколко команди, чрез които се контролира редът на действията. Работи на принципа на команден интерпретатор. Командите са:

*ЛЕГЕНДА: [value] - описва опционален аргумент, <value> - описва задължителен аргумент:*

* ДобавиАстронавт <име на астронавт> <длъжност/категория> //добавя астронавт към съществуващата база данни от абстрактният клас Astronaut
* ДобавиПланета <име на планета> [предмет\_1,предмет\_2, … предмет\_X] //Добавя планета към данните на обекта PlanetRepository. Записването на предметите става чрез разделител “,” БЕЗ разстояние
* ПенсионирайАстронавт <име на астронавт> //премахва астронавт от гореспоменатата БД
* ИзследвайПланета <име на планета> //инициализира избор на астронавти за експедиция до посочената планета
* Отчет //показва текущото състояние на астронавтите (Име, Количество кислород, Предмети)
* Край //приключва програмата

//Всяка грешно въведена команда се игнорира и програмата очаква нов вход на данни

Програмата работи и в среда на Microsoft Windows, но въвеждането на командите се извършва на английски език.

* AddAstronaut <type> <name>
* AddPlanet <name> [item1,item2, … itemN]
* RetireAstronaut <name>
* ExplorePlanet <name>
* Report
* End

**Изложение**

Ще разгледам само най-важните части на програмата

**Main функцията**

Източници:

* Robert C. Martin (Uncle Bob), [*Design Principles and Design Patterns*](https://web.archive.org/web/20150906155800/http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf) *//SOLID*