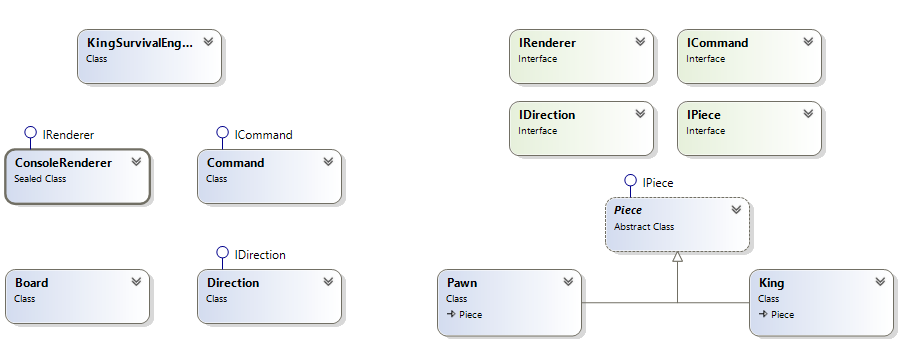
**Отбор King Survival 1**

****

**Н**ашият отбор има за задача да рефакторира и допълни дадения код за създаване на играта King Survival. В нея на стандартна шахматна дъска 8x8 полета имаме цар King и четири пешки PawnA, PawnB, PawnC и PawnD. Играчът задава команди за движение на царя и на пешките, като целта е или царят да стигне до row 0 на дъската и да победи, или да бъде притиснат от пешките, така че да няма повече нито един възможен ход. Царят печели играта и в случай, че нито една от пешките няма възможен ход по игралното поле. Играчът движи фигурите и на царя, и на пешките чрез команди в конзолата.

**Д**аденият ни код беше рефакторирен, преименуван, променен и допълнен, като в резултат целият код е организиран в King Survival Solution. В него имаме обособени три папки – Engine, GamePlay Classes, Interfaces и отделно файловете KingSurvival.Tests и StartGame.cs. Диаграмата на проекта изглежда по следния начин.



**Файлове:**

**KingSurvival.Tests.** – Съдържа unit тестове на кода, разделени логически в отделни файлове – Pawn Tests, King Tests, Direction Tests, Board Tests.

**Games**.cs – тук се съдържа Main.cs класът на проекта и от него се стартира Engin-a.

Папка **Engine**:

**KingSurvivalEngine**.cs – В метода Run() се прави смяната на ходовете на двете страни – царят и пешките и се извършва разпознаване на зададените команди за движение и изпълнението им чрез метода ProcessCommand(). Тук се прави и проверка за край на играта и кой е победителят и се дават упътвания на конзолата за случващото се в процеса на игра.

**Constants**.cs – съдържа константите в играта – първоначалните позиции на фигурите и техния брой.

**Console.Renderer**.cs – наследява интерфейса IRenderer. Служи за изчистване и изрисуване на игралната дъска след всеки ход.

**Command.cs** – наследява интерфейса ICommand и превръща подадения от потребителя вход в команда за движение.

**PieceFactory.cs –** тук се създават фигурите и им се задават координати. Наследява интерфейса IPieceFactory.

**Папка Formatters** – в нея има форматиращи методи, наследяващи IFormatter.

Папка **GamePlay** **Classes**:

**Board**.cs – Класът представя игралната дъска, представляваща матрица от бели и черни полета.

**Direction.cs –** Задава и ъпдейтва посоките на движение на фигурите.

**King**.cs – Класът на царя, в който се съдържат конструктор за King и се имплементира метода Move(), който изпълнява движението на царя и брояч за направените ходове.

**Pawn**.cs – Класът на пешките, в който се съдържат конструктор за пешка и се прави проверка за валидността на зададения символ на пешката.

**Piece**.cs – наследява интерфейса IPiece и съдържа конструктор за създаване на фигурите.

**Validations**.cs – класът служи за правене на проверки по отношение на командите – дали са правилно подадени и могат ли да бъдат изпълнени.

Папка **Interfaces**:

**ICommand.cs –** свързва се с командите за движение.

**IDirection.cs –** задава координатите при движението.

**IFormatter.cs –** използва се за правилното форматиране на символите.

**IPiece.cs –** представя движението на фигурите по дъската.

**IPieceFactory.cs –** свързва се със създаването на фигурите.

**IRenderer**.cs – съдържа метода Render() и се отнася до прерисуването на дъската след всеки ход.

Списък с промените в хронологичен ред:

* разделяне на първоначалния код в няколко основни класа
* почистване на Engine-а от проверките за валидни ходове
* почистване на метода Move в класa King
* почистване на метода Move за Pawn
* добавяне на константи
* преименуване на променливите в Engine
* отделяне на функционалността за Pawn в нов клас (Pawn)
* преименуване на променливи в Engine, Pawn и King
* отделяне на функционалността за игралното поле в Board
* имплементиране на интерфейса IMovable
* организиране на файловете в папки по namespace-и
* отделяне на функционалността за попълване на игралното поле в метод
* промяна поведението на Board класа от статично на динамично (instantiated)
* добавяне на свойство за масива от символи в Board и имплементиране на защита от подаването му чрез референцията му
* добавяне на клас диаграма
* имплементиране на движението на царя с необходимите проверки за полетата и визуализирането
* имплементиране на свойство, което позволява масива за игралното поле да се достъпва с индекси
* имплементиране на движението на пешките с необходимите проверки за полетата и визуализирането им
* имплементиране на константи за символите, текстовете и размерите
* отделяне на функционалността за посоките на движение и подаването на командите за посока в отделни класове
* оптимизиране на класа за визуализиране на играта
* добавяне на абстрактен клас за игрална фигура
* имплементиране на метод, който проверява, дали дадено поле е заето от фигура
* почистване на повтарящ се код за метода Move чрез наследяване на базовия метод
* добавяне на методи за проверка за победител и изпълнение на последващи инструкции
* презаписване на проверката за еднаквост в класа Direction
* добавяне на проверки за валидни посоки на движение
* добавяне на интерфейса IDirection
* добавяне на интерфейса IPiece
* добавяне на интерфейса ICommand
* общо промени по определенията за достъп
* отделяне на функционалността за принтиране на съобщения в отделен метод
* освобождаване от зависимостта на методите за писането от конзолата
* отделяне на всички проверки за валидност в отделен клас
* имплементиране на AbstractFactory за фигурите
* имплементиране на Singleton за визуализирането на играта
* документиране на класовете и методите
* добавяне на тестове
* имплементиране на Bridge между различните по размер изобразявания на игралното поле

***Линк към GitHub на проекта*:** <https://github.com/TeamKingSurvival1/KingSurvival/>

**Участници в проекта:**

Александър Димитров – **a.dimitrov**

Иван Бангиев – **bangiev**

Крум Тюкенов – **Overon**

Даниела Пенкова – **Regina05**

Мирослав Цаков – **miroslav.tsakov**

Ангел Великов – **angel\_rw**

Росица Бойкова – **rosica.boykova**