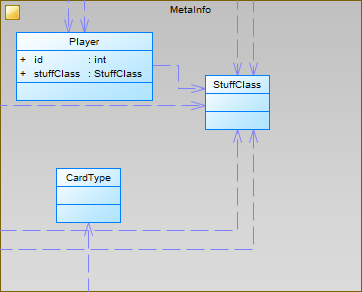
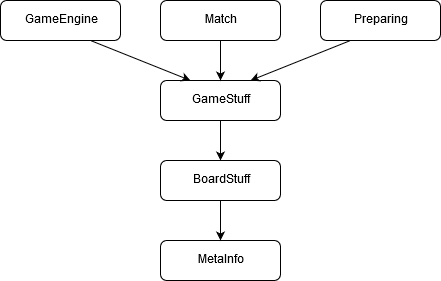
# Описание архитектуры игры «Faculty wars»

Архитектура игры «Faculty wars» должна обеспечивать следующие возможности:

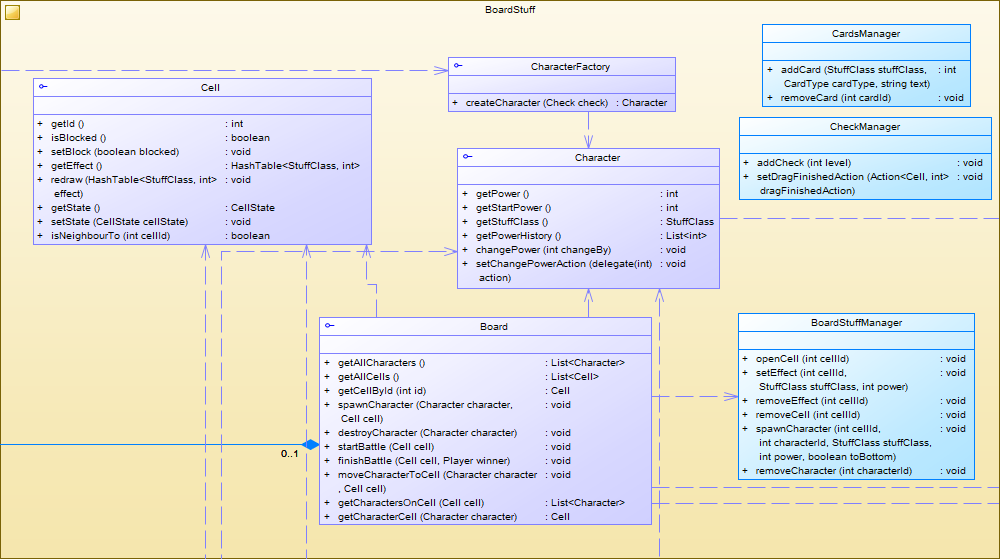
* Простота добавления новых карт
* Максимальное количество игровых механик, доступных для использования в карте
* Возможность разрабатывать отдельно игровую логику, взаимодействие с игроком, игровые компоненты (карты), отображение на экран.

Общая схема разработанной архитектуры изображена на схеме ниже.



MetaInfo – архитектурная область, в которой содержатся базовые элементы игры, которые практически никогда не меняются и доступ к которым может осуществляться из любой части кода. Включает в себя:

* Player, структура – хранит базовую информацию об игроке.
* StuffClass, перечисление – хранит все классы персонажей игры. На данный момент их три: IASA, FICT, FPM.
* CardType, перечисление – хранит все типы карт игры. На данный момент их четыре: золотая, серебряная, нейтральная и играющаяся в любой момент.



\* - на диаграмме изображены только основные элементы

BoardStuff – архитектурная область, отвечающая за вывод на экран клеток поля, персонажей, фишек и карт, за отрисовку анимаций и обработку действий пользователя.

API:

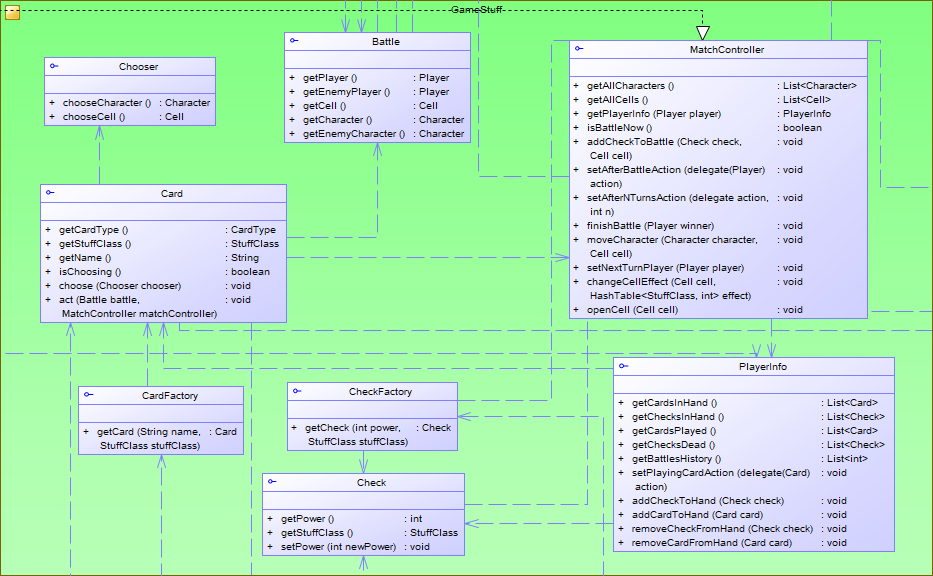
* Board, интерфейс – предоставляет полный контроль над ситуацией на доске, без учета каких-либо правил игры. Примеры методов: заспаунить персонажа на клетке поля, уничтожить персонажа, открыть клетку поля, отрисовать эффект на клетке поля, переместить персонажа на другую клетку.
* Cell, интерфейс – предоставляет информацию о клетке поля. Не позволяет выполнять никаких действий, кроме установки блокировки на клетку – все действия с клеткой осуществляются через интерфейс Board.
* Character, интерфейс – предоставляет информацию о персонаже на поле.

Данная архитектурная область разделена на несколько слоёв.

Слой 1 – непосредственная отрисовка игровых объектов (карт, персонажей, клеток поля) и низкоуровневая обработка действий пользователя. Включает в себя в основном скрипты префабов игровых объектов. Классы: CardClickHandler, CheckDragHandler.

Слой 2 – низкоуровневое управление игровыми объектами. К этому слою относятся классы, которые отвечают за работу панели с картами (содержит методы добавления и удаления карты из панели), панели с фишками, игрового поля (имеет методы изменения текстуры клетки, появления и удаления персонажа и т. п.). Классы: CardsManager, CheckManager, BoardStuffManager.

Слой 3 – относительно высокоуровневое управление компонентами на экране. Классы этого слоя позволяют выполнять такие действия, как спаун персонажа на клетке, получение списка всех персонажей на поле, отрисовка определенного эффекта на клетке поля, и т. п. Реализуют интерфейсы архитектурной области BoardStuff, которые предоставляют API для остальных компонентов игры. Классы: BoardController, PlayerController (реализует интерфейс доступа к картам и фишкам игрока из архитектурной области GameStuff), CellImpl, CharacterImpl.



GameStuff – архитектурная область, содержащая компоненты игровой логики: карты и фишки. Для реализации карт необходим доступ к полю игры и к панелям карт и фишек игрока. Для этого в области присутствуют три интерфейса, реализованные в других областях:

* MatchController – предоставляет доступ ко всем компонентам игрового поля. Карты работают не напрямую с интерфейсом Board, описанном в предыдущем разделе (поскольку он не учитывает правила игры), а с данным интерфейсом, содержащим высокоуровневые методы (завершить битву, установить игрока, который будет ходить следующим и т. п.).
* PlayerInfo – предоставляет доступ к панели карт игрока и к панели фишек игрока. Реализован в архитектурной области BoardStuff классом PlayerController.
* Battle – хранит информацию о текущей битве: информацию об игроке и о противнике, о персонажах игрока и противника.
* Chooser – позволяет осуществлять выбор персонажа на поле, клетки поля или фишки перед выполнением эффекта карты.

*\*\* отступление о том, как реализованы битвы*

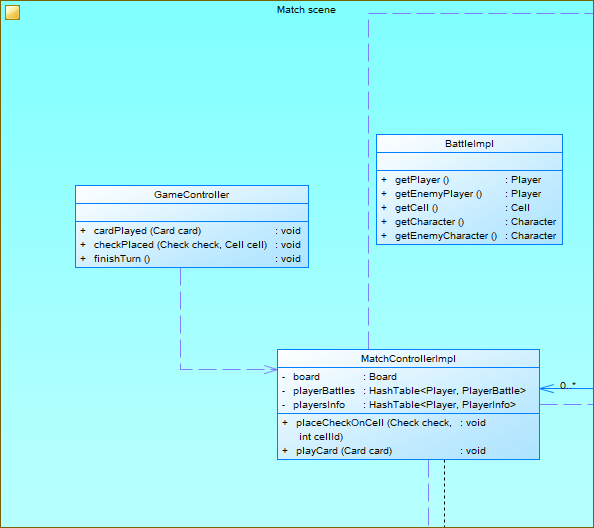
*Когда на клетке поля оказываются хотя бы одна пара персонажей, принадлежащих разным игрокам, для каждой пары персонажей создается два объекта Battle – для каждого игрока свой. При разыгрывании игроком карты, в нее будет передаваться соответствующий объект Battle. Несколько примеров:*

1. *На клетке оказываются два персонажа, принадлежащих разным игрокам – начинается битва. Создается два объекта Battle – для первого персонажа и для второго (соответственно в них по-разному устанавливаются поля «мой персонаж» и «персонаж противника»). Когда первый игрок разыгрывает карту, в её метод act передается объект Battle его персонажа. При разыгрывании карты вторым игроком, в неё будет передаваться второй объект Battle.*
2. *На клетке находилось два персонажа игрока1, и игрок2 выложил на клетку своего персонажа – начинается битва. Поскольку теперь на клетке есть две пары противоборствующих персонажей, создается четыре объекта Battle – для каждой пары, для каждого игрока. Когда игрок1 разыгрывает карту, её метод act вызывается дважды. В первый раз туда передастся объект Battle первой пары от его лица, во второй раз – объект второй пары. Аналогично при разыгрывании карты игроком2.*
3. *На клетке четыре персонажа – по два от каждого игрока. Создано восемь объектов Battle. Каждая карта будет срабатывать четыре раза.*

API:

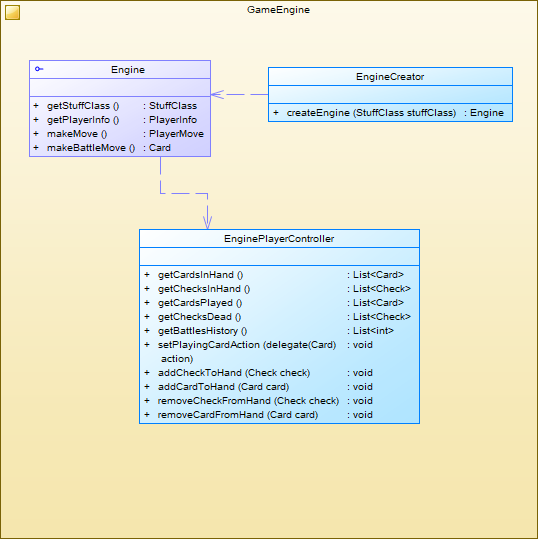
* Card, абстрактный класс – хранит информацию о карте, а также определяет два абстрактных метода. Act – выполнение эффекта карты, принимает объекты Battle и MatchController. Choose – осуществление выбора перед выполнением эффекта карты (некоторые карты требуют от игрока осуществить выбор персонажа или клетки поля. В данном классе осуществляется этот выбор с помощью интерфейса Chooser).
* Check, интерфейс – хранит информацию о фишке
* CardFactory, интерфейс – фабрика карт
* CheckFactory, интерфейс – фабрика фишек

Каждая карта игры реализована в виде отдельного класса-наследника Card. Фишки реализованы с помощью одного класса CheckImpl.



Match – архитектурная область, содержащая высокоуровневую информацию об игре, такую как реализация игровой логики и управления игрой.

* MatchControllerImpl, класс – реализует игровую логику, выполняет роль модели. В этом классе описаны правила игры. Используя методы интерфейса Board, класс выводит актуальную информацию на экран. Следит за состоянием битв, поля и игры в целом. Реализует интерфейс MatchController, таким образом предоставляя доступ картам для управления ситуацией на поле, в пределах правил игры. Ничего не знает об игроке за компьютером, оба игрока для данного класса полностью равноправны.
* GameController, класс – главный и наиболее высокоуровневый класс игры. Выполняет функции контроллера. Принимает действия игрока, считанные и первично обработанные более низкоуровневыми компонентами, и вызывает соответствующие методы модели. Выполняет инициализацию бота-оппонента и связывает его с моделью. Следит за очередностью ходов игрока и бота.
* BattleImpl, класс – хранит информацию о битве. Реализация интерфейса Battle, описанного в предыдущем разделе.
* PlayerMove, интерфейс – хранит информацию о ходе оппонента-бота.



GameEngine – архитектурная область, отвечающая за реализацию оппонента-бота.

API:

* Engine, интерфейс – основной интерфейс области. Через него осуществляется получение ходов бота, обычных и боевых.
* EngineCreator, статический класс – фабрика объектов Engine
* EnginePlayerController – реализация интерфейса PlayerInfo (область GameStuff) для бота. Хранит информацию о картах и фишках в руке бота.