Государственный университет – Высшая школа экономики

Факультет Бизнес-Информатики

Кафедра Основ информатики
и прикладного программного обеспечения

C#

Объектно-ориентированный язык программирования

ОО проект:
Компьютерная игра. Итерации 1…4.

Проф. Забудский Е.И.

Москва 2008

Тема 9. Разработка компьютерных моделей
на основе объектно-ориентированной методологии:
практический пример

ОО проект:
Компьютерная игра Блэк джек (Blackjack). Итерация 1.

Выполнена только 1-я итерация.

Код в итерации 1 разрабатывается студентами самостоятельно

Итерации 2, 3 и 4 выполняются студентами самостоятельно

Дополнение 1 к этому проекту необходимо рассмотреть предварительно

Unified Modeling Language (UML) – популярный язык графического моделирования, используемый для представления объектно-ориентированных программ (**www.omg.org/uml**).

// **КОММЕНТАРИЙ**. На сайте http://www.firststeps.ru/ “Первые шаги” представлено много интересных обучающих материалов по различным интегрированным средам и языкам программирования, в том числе представлены C# и платформа .NET (step by step).

Содержание

Объектно-ориентированный проект:
игра Блэк джек (Blackjack)

Будут рассмотрены все этапы разработки реального проекта, от самого начала и до завершения готового проекта.

Первая итерация разработки игры и с помощью объектно-ориентированного анализа (ООА): определение основных функций системы, их реализация и испытание. Выполнение: **1)** начального анализа, **2)** проектирования и 3) программной реализации этой игры (этот пункт студентам предлагается выполнить самостоятельно).

Создание игры не будет проводиться в один прием. Процесс будет итеративным и постепенным.

1. Блэк джек (Blackjack) – цель рассмотрения игры

Блэк джек (**Blackjack**) – карточная игра, цель которой набрать больше очков, чем раздающий карты (**dealer**), но не более чем **21**.

Цель состоит в том, чтобы, применяя концепции объектно-ориентированного программирования, которые изучались на протяжении предыдущих занятий, написать на языке C# программную реализацию игры Блэк джек (**Blackjack**). Для упрощения некоторые особенности игры не учитываются.

2. Почему именно Блэк джек (Blackjack)?

Две причины использования игры Блэк джек (**Blackjack**) в качестве проекта объектно-ориентированного программирования:

- Почти все имеют представление о какой-нибудь карточной игре.
- Игра Блэк джек (**Blackjack**) хорошо вписывается в парадигму объектно-ориентированного программирования.

С такой предметной областью (доменом), как карточные игры, – знакомы все. Важнейшее условие успешного объектно-ориентированного анализа и проектирования — возможность консультироваться у специалиста (эксперта) в той предметной области, модель которой создается. А в течение одного, двух занятий едва ли можно стать специалистом в какой-нибудь незнакомой до этого области знаний.

Для реализации же карточной игры не требуется специалист в проблемной области. Собственный опыт – вот и все, что нужно для самостоятельного анализа и проектирования. Если возникнут какие либо затруднения, всегда можно взять колоду карт и разыграть сценарий, т.е. выполнить анализ и проектирование с ее помощью.

Другими словами, значительно проще использовать карточную игру, чем пытаться изучить незнакомую предметную область. Знание предметной области даёт возможность сконцентрироваться на действительной цели занятия – изучении применения принципов объектно-ориентированного программирования.

Карточные игры, и Блэк джек (Blackjack) в частности, очень хорошо вписываются в парадигму объектно-ориентированного программирования. Взаимодействия между раздающим карты, игроками, картами у них на руках и другими картами поможет увидеть, как в объектно-ориентированных системах реализуется взаимодействие между различными объектами.

3. Постановка задачи

Каждый новый проект начинайте с постановки задачи, или цели. В момент постановки задачи определяются наиболее общие характеристики создаваемой системы. Постановка задачи не должна охватывать все аспекты решаемой проблемы.

Постановка задачи (vision statement) определяет наиболее общие характеристики системы, которую вы собираетесь создать, и проблем, которые придется решать при этом.

Постановка задачи для системы "игра Блэк джек (Blackjack)":
игра Блэк джек (Blackjack) позволяет игрокам играть в нее по общепринятым правилам.

4. Необходимые условия

Перед началом анализа желательно определить необходимые условия. Игра Блэк джек (**Blackjack**) требует выполнения немногих условий. В первую очередь надо определить, каким образом пользователь будет взаимодействовать с системой.

Игра Блэк джек (**Blackjack**) позволяет пользователю взаимодействовать с системой: **1)** посредством командной строки или **2)** с помощью графического пользовательского интерфейса (**GUI**). Естественно, в начальных версиях игры взаимодействие с пользователем происходит только с помощью интерфейса командной строки.

Важно составить список всех необходимых требований перед тем как приступать к проектированию. В противном случае может оказаться, что в разработанном проекте выполнить необходимые требования будет невозможно.

5. Начальный анализ игры Блэк джек (Blackjack)

На лекции 12, разд. 3 «Введение в объектно-ориентированный анализ (ООА)» рассмотрены **ОО Анализ**, а также этапы итеративного процесса разработки. Процесс разработки игры **Блэк джек** будет итеративным, причем каждая итерация будет начинаться с анализа.

Лучше всего начать анализ с описанной ранее постановки задачи: игра Блэк джек (Blackjack) позволяет игрокам играть в нее по общепринятым правилам.

Используем постановку задачи для составления начального списка вопросов. Четко сформулированные вопросы помогут в дальнейшем анализе.

6. Правила игры Блэк джек (Blackjack)

6.1. Цель игрока – собрать на руках такое сочетание карт, чтобы сумма очков была больше, чем на руках у раздающего карты, но не превосходила **21**. Каждой карте соответствует определенное количество очков от **1** до **11**, причем туз может означать и **1** и **11** (в зависимости от того, какое значение дает преимущество). Карты до десятки "стоят" соответственно своему численному значению (двойка – 2, пятерка – 5 очков, etc.), а все картинные карты – по 10 очков. Масть для определения достоинства не имеет никакого значения.

Рассмотрим по очереди каждый из наиболее важных аспектов игры.

6.2. Ставки

Прежде чем раздающий карты их раздаст, каждый игрок должен сделать ставку. Ограничим ставку 25 или 50 рублями за игру.

6.3. Раздача карт

После того как все игроки сделали ставки, раздающий (**dealer**) должен раздать карты. **Начиная с первого игрока**, раздающий карты (**дилер**) дает каждому игроку по одной карте лицом кверху так, что раздающему достается последняя карта.

Затем процесс повторяется с единственным отличием – раздающий карты кладет свою карту лицом вниз. Перевернутая карта раздающего называется скрытой картой (**hole card**).

Раздача заканчивается, когда каждому игроку (включая раздающего карты) **роздано по две карты**. После завершения раздачи начинается игра.

6.4. Игра

Ход игры зависит от открытой карты раздающего (**dealer**). Если открытая карта раздающего – десятка

(приносит **10** очков) или туз (**11** очков), раздающий карты должен проверить свою скрытую карту и если обе карты дают ему в сумме **21** очко (такая сумма называется естественной или Блэк джек [**Blackjack**]), игра автоматически оканчивается и играющие переходят к расчету. Если же скрытая карта не приносит 21 очко, игра продолжается дальше в обычном порядке.

В случае, когда открытая карта раздающего — не десятка и не туз, в игру вступает следующий игрок. Если у него **21** очко, то есть Блэк джек (**Blackjack**), игра переходит к следующему игроку. Если же у игрока сумма очков меньше **21**, возможны два варианта: **1)** взять еще карту или **2)** остановиться.

6.4.1. Взять еще (hit) — если игрок не доволен картами, которые у него на руках, он может взять еще карту. Игрок может брать карты до тех пор, пока: **1)** не наберет больше, чем **21** очко (перебор, то есть проигрыш, банкротство, разорение) или **2)** не остановится.

6.4.2. Остановка (stand) — если игрок доволен картами на руках, он может остановиться и не брать дополнительных карт.

После того как игрок обанкротится или остановится, игра переходит к следующему игроку. Этот процесс повторяется до тех пор, пока не сыграют все игроки. Когда сыграют все игроки, раздающий карты разыгрывает имеющиеся у него карты на руках. После того как сыграет раздающий карты, играющие переходят к подсчету очков и выплате выигрышей.

7. Расчет

После того как сыграет раздающий карты (или когда у него Блэк джек [**Blackjack**]), игроки рассчитываются. Игроки, которые обанкротились (набрали больше чем **21** очко – перебор), теряют свои ставки. Игроки, у которых на руках такие карты, что очков меньше, чем у раздающего карты, также теряют свои ставки. Только игроки, набравшие очков больше, чем раздающий карты, выигрывают суммы, равные своим ставкам. Наконец, считается, что игроки, набравшие столько же очков, сколько и раздающий карты, сыграли "вничью" и в дележе выигрыша участия не принимают. Таким образом, игрокам, сыгравшим "вничью", никаких выплат не производится. После этого ставки делятся поровну между участниками дележа.

Если у игрока комбинация Блэк джек (**Blackjack**), а у раздающего карты – нет, то игроку выплачивается выигрыш в размере три вторых его ставки. Например, если ставка составляет 100 рублей, то выплачивается 150 рублей (100*3/2).

8. Дополнительные сведения

Нужно упомянуть еще несколько важных подробностей, относящихся к правилам игры Блэк джек (**Blackjack**).

Колода – Блэк джек (**Blackjack**) играется четырьмя стандартными колодами по **52** карты. Эти четыре колоды складываются в одну большую стопку карт.

Количество игроков – в Блэк джек (**Blackjack**) могут играть от одного до семи игроков.

Удваивание ставок – после того как игрок получил две карты, он может удвоить ставку. Если игрок принял такое решение, он удваивает свою ставку, получает еще одну карту и заканчивает свой ход.

Страховка – если первая (открытая) карта раздающего карты – туз, игрок может сделать **страховую** ставку. Величина страховой ставки равна половине первоначальной ставки. Если скрытая карта раздающего приносит ему **21** очко, игрок остается при своих (*т.е. заканчивает игру без прибыли и убытка*). Если же скрытая карта не приносит раздающему **21** очко, игрок теряет страховую ставку.

Разбиение пары – говорят, что игрок имеет пару, если после раздачи две начальные карты имеют одинаковое достоинство (например две десятки). Если игрок имеет пару, то он может разделить карты на руках на две новые руки. Если игрок разбивает пару, раздающий карты раздает по одной дополнительной карте в каждую руку, а игрок должен сделать еще одну ставку, равную начальной, на дополнительную руку. Игрок может разбивать пары (кроме пары из двух тузов), которые образовались в результате предыдущего разбиения пар. Заметим также, что **21** очко на одной из рук, появившихся в результате разбиения пар, не рассматривается как Блэк джек (Blackjack). После разбиения пары игрок может по очереди для каждой руки брать еще карты или останавливаться.

9. Объектно-ориентированный анализ игры Блэк джек (Blackjack). Идентификация действующих лиц

(см. Материалы к Лекции 12 и 13, разд. 3, с. 33,сл.)

Есть два довода в пользу анализа. Посредством анализа создаем:

1) модель использования: описание того, как пользователь может использовать систему;

2) модель предметной области: определение основных понятий, используемых при описании системы.

Первый шаг создания модели использования – **определение действующих лиц**, которые будут использовать систему. Как следует из предыдущего раздела, в игре Блэк джек (**Blackjack**) два действующих лица – **игрок**(и) (**player - человек**) и **раздающий карты (dealer - компьютер)**.

Определив действующих лиц, ответили на вопрос "**Кто будет использовать систему?**". Посредством модели использования, можно определить - **Как действующие лица будут использовать систему?**

9.1. Создание предварительного списка различных способов использования (прецедентов)

Для эффективного построения модели использования нужно перечислить все возможные действия каждого действующего лица.

А. Игроки могут:

1. делать ставки;
2. брать дополнительные карты;
3. не брать дополнительные карты (останавливаться);
4. разориться, или обанкротиться, в результате перебора (т.е. набрать больше **21** очка);
5. получать Блэк джек (**Blackjack**);
6. делать страховую ставку;
7. разбивать пары;
8. удваивать ставку;
9. продолжать играть дальше;
10. прекратить игру.

Б. Раздающий карты может:

1. раздавать карты;
2. заканчивать игру;
3. брать дополнительные карты;
4. не брать дополнительные карты (останавливаться);
5. разориться, или обанкротиться, в результате перебора (т.е. набрать больше **21** очка);
6. получить Блэк джек (**Blackjack**).

В приведенном списке рассмотрены не все случаи, но для начала этого хватит.

9.2. Планирование итераций (от лат. *iteratio – повторение*)

На лекции 12, разд. 3 «Введение в объектно-ориентированный анализ (ООА)» знакомились с итеративным процессом разработки. На начальном этапе итеративного процесса разработки игры создадим базовую (предельно простую) реализацию игры Блэк джек (**Blackjack**). Каждая последующая итерация будет добавлять к базовой реализации игры дополнительные функциональные возможности.

Такой подход обеспечивает быстрые, ощутимые результаты. Еще одно преимущество такого подхода – устранение возможных проблем по мере их появления, а не в конце разработки. Таким образом, итерационный подход предотвращает лавинообразное появление проблем в конце разработки.

Самое важное в итерационной разработке – начать с тщательного планирования итераций. При выявлении дополнительных фактов план итераций придется переработать; однако начальный эскиз задает направление проектирования, цели и даже даст нечто более важное – чувство удовлетворения по мере достижения целей.

Обычно итерации планируются, исходя из важности того или иного варианта использования программы. **При работе с заказчиком лучше всего позволить заказчику определять важность каждого варианта использования**. В случае с Блэк джек (**Blackjack**), необходимо определять важность каждого варианта использования, исходя из функциональности готовой игры. **Выберем варианты использования программы, абсолютно необходимые для готовой игры, и реализуем их в первую очередь.** Другие варианты использования программы могут быть реализованы на более поздних итерациях. Такой подход позволит как можно скорее создать работающую реализацию игры. Проект Блэк джек (**Blackjack**) будет завершен за четыре основных итерации.

9.2.1. Итерация 1: Готовая базовая игра (в материале лекции реализована 1-я итерация)

Выполнив первую итерацию, создадим готовую базовую игру. В итерации 1 реализуются следующие, совершенно необходимые, возможности.

А. Возможности, используемые игроком:

1. брать дополнительные карты;
2. не брать дополнительные карты (останавливаться);
3. разориться, или обанкротиться, в результате перебора (т.е. набрать больше **21** очка).

Б. Возможности, используемые раздающим карты (**dealer**):

1. сдавать карты;
2. брать дополнительные карты;
3. не брать дополнительные карты (останавливаться);
4. разориться, или обанкротиться в результате перебора (т.е. набрать больше 21 очка).

В конце итерации 1 получили игру, в которую можно играть с помощью интерфейса командной строки. В игре будет двое участников: раздающий карты (**компьютер**) и один игрок (**человек**). Раздающий карты будет сдавать карты, причем он будет позволять каждому игроку (в данном случае одному) брать дополнительные карты, пока игрок не решит остановиться или не обанкротится. После того как сыграют все игроки, игра завершится.

9.2.2. Итерация 2: Правила (эту итерацию студентам предлагается выполнить самостоятельно)

Вторая итерация добавляет к игре правила. Итерация 2 реализует следующие возможности использования.

А. Возможности, используемые игроком:

1. получить **Блэк джек (Blackjack)**.

Б. Возможности, используемые раздающим карты (**dealer**):

1. вести игру (определять победителей, проигравших и сыгравших вничью);
2. получать **Блэк джек (Blackjack)**.

В конце Итерации 2 будут работать все возможности, реализованные в Итерации 1. Кроме того, программа определит и укажет, когда игрок: а) получит Блэк джек (Blackjack), б) разорится (обанкротится), в) остановится, г) выиграет, д) проиграет или е) сыграет вничью.

9.2.3. Итерация 3: Ставки (эту итерацию студентам предлагается выполнить самостоятельно)

В Итерации 3 добавится возможность: **1)** делать ставки перед игрой и **2)** удваивать их во время игры. Итерация 3 реализует **следующие варианты использования**.

А. Возможности, используемые игроком:

1. делать ставки;
2. удваивать ставки.

Б. Возможности, используемые раздающим карты (**dealer**):

1. вести игру (принимать ставки).

В конце Итерации 3 будут работать все возможности, реализованные в Итерациях 2 и 1. Кроме того, программа позволит делать и удваивать ставки.

9.2.4. Итерация 4: Пользовательский интерфейс (эту итерацию студентам предлагается выполнить самостоятельно)

В Итерации 4 будет усовершенствован пользовательский интерфейс командной строки и будет создан графический пользовательский интерфейс на основе шаблона Модель – Вид – Контроллер (см. Лекция 15, разд. 2, с. 4…23). Итерация 4 реализует следующие варианты использования.

А. Возможности, используемые игроком:

1. принять решение играть снова;
2. завершить игру.

С целью упрощения проекта страховая ставка и разбиение пар опущены. Эти возможности оставлены в качестве самостоятельного упражнения. Блэк джек (**Blackjack**) — игра с большим количеством различных вариаций. Часто страхование не позволяется, а разбиение пар чрезмерно усложняет систему.

10. Итерация 1: Готовая базовая игра

Анализ игры Блэк джек (**Blackjack**)

10.1. Возможности реализуемые на Итерации 1. 9

А. Возможности, используемые игроком (**player - human**):

1. брать дополнительные карты;
2. не брать дополнительные карты (останавливаться);
3. разориться, или обанкротиться, в результате перебора (т.е. набрать **больше 21** очка).

Б. Возможности, используемые раздающим карты (**dealer - computer**):

1. сдавать карты;
2. брать дополнительные карты;
3. не брать дополнительные карты (останавливаться);
4. разориться, или обанкротиться, в результате перебора (т.е. набрать **больше 21** очка).

После того, как определен набор используемых вариантов, можно приступать к созданию начальной модели предметной области и проектированию.

10.2. Уточнение используемых возможностей

Начнем с раздающего карты (**dealer**), поскольку игра начинается с раздачи карт. Сначала нужно описать все возможности использования обычным языком (прежде чем писать на языке C#).

Раздающий карты раздает каждому игроку, начиная с первого и заканчивая собой, **по одной карте лицом вверх**. Затем раздающий повторяет этот процесс, но свою карту кладет **лицом вниз**. После того как раздающий карты раздаст всем игрокам, включая и себя, по две карты, раздача заканчивается и начинается игра.

- **Б.1. Раздача карт (dealer - computer):**

1. раздающий карты раздает каждому игроку, включая и себя, по одной карте лицом **вверх**;
2. раздающий карты раздает каждому игроку, кроме себя, вторую карту лицом **вверх**;
3. раздающий карты сдает **себе** карту **лицом вниз** (скрытая карта).

- Предварительные условия:

• новая игра.

- Постусловия, т.е. выходные условия:

• все игроки и раздающий карты имеют на руках по две карты.

Далее следуют варианты использования "**игрок берет еще карты**" (**Player Hits**), и "**игрок останавливается**" (**Player Stands**). Начнем с варианта использования "игрок берет еще карты" (Player Hits).

Игрок решает, что карты на руках его не устраивают. Он еще не разорился, и решает брать дополнительные карты. Если игрок не разорился, он может выбирать, то ли ему снова брать карту, то ли остановиться. Если игрок разорится, игра переходит к следующему игроку.

- **А.1. Игрок берет еще одну карту (player - human):**

1. игрок решает, что карты на руках его не устраивают;
2. игрок просит раздающего карты дать ему еще одну карту;
3. игрок может брать дополнительные карты либо остановиться, если общее количество очков карт на руках меньше или равно 21.

- Предварительные условия:

• игрок имеет на руках карты, причем общее количество очков карт меньше или равно **21**.

- Постусловия, т.е. выходные условия:

• к картам на руках у данного игрока добавляется новая карта.

- Альтернативный вариант — игрок разоряется:

• новая карта приводит к тому, что сумма очков карт на руках у игрока превышает **21**. Игрок разоряется (проигрывает). Наступает очередь следующего игрока или раздающего карты.

Рассмотрим простой случай использования – "Игрок останавливается". Игрок решает, что он доволен картами на руках и останавливается.

- **А.2. Игрок останавливается (player - human):**

1. игрок решает, что он вполне доволен картами на руках и останавливается.

- Предварительные условия:

• сумма очков карт на руках у игрока меньше или равна **21 (<=21)**.

- Постусловия, т.е. выходные условия:

• право очередного хода передается следующему игроку.

Теперь становится ясно, что разорение игрока или раздающего карты не является используемой возможностью, поскольку это следствие других действий. Раздающий карты и игрок никогда не смогут совершить действие разорения; однако они могут выбирать: брать дополнительные карты или остановиться.

После того как мы удалим такие варианты использования, как разорения – А.3 и Б.4, нам останется разобрать только варианты использования "раздающий карты берет еще карты" **(Dealer-computer Hits)** и "раздающий карты останавливается" **(Dealer-computer Stands)**. Начнем с возможности:

> Раздающий карты должен брать дополнительную карту, **если сумма очков карт на руках у него меньше 17 (< 17)**. Если раздающий не разорится после того, как возьмет карту, а сумма очков карт на руках у него все еще меньше 17, он должен взять еще одну карту. Раздающий карты должен **остановиться**, если сумма очков карт на руках у него больше или равна 17. Когда раздающий **разорится** или **остановится**, игра завершается.

- **Б.2. Раздающий карты берет еще карты (Dealer /computer/ Hits):**

1. раздающий карты должен брать дополнительную карту, если сумма очков карт на руках у него меньше 17 **(< 17)**;
2. к картам на руках у раздающего добавляется новая карта;

3. **если у раздающего сумма очков карт на руках равна 17 (=17), он должен взять еще одну карту.**

- Предварительные условия:
- сумма очков карт на руках у раздающего равна 17 **(= 17)**.

- Постусловия, т.е. выходные условия:
- на руках у раздающего карты появляется новая карта;
- игра заканчивается.

- Альтернативный вариант – раздающий карты разоряется:
- Добавление новой карты к имеющимся на руках приводит к тому, что сумма очков карт на руках у раздающего превышает **21 (> 21)**. Раздающий карты разоряется (проигрывает).

- Альтернативный вариант – раздающий карты останавливается:
- После добавления новой карты сумма очков карт на руках у раздающего больше или равна **17 (>= 17)**. Он останавливается.

Случай "Раздающий карты останавливается", как и случай "Игрок останавливается" довольно прост.

Сумма очков карт на руках у раздающего больше или равна 17 **(>= 17)** и он останавливается.

- **Б.3. Раздающий карты останавливается (Dealer-computer Hits):**
- сумма очков карт на руках у раздающего больше или равна **17 (>= 17)**.

- Предварительные условия:
- сумма очков карт на руках у раздающего больше или равна **17 (>= 17)**.

- Постусловия, т.е. выходные условия:
- игра заканчивается.

10.3. Моделирование возможностей использования

В итерации 1 случаи использования довольно просты. В действительности модели использования возможностей могут быть сложнее; рассматриваем упрощенный вариант.

Взаимодействия между раздающим карты и игроками более интересны. На **рис. 1** показана последовательность событий, происходящих при использовании возможности "**раздача карт**". На **рис. 2** показана последовательность событий, происходящих при использовании возможности "**игрок берет еще карты**".

10.4. Моделирование предметной области

Определив варианты использования в качестве основы для построения модели предметной области, можно выделить семь отдельных объектов:

1. **BlackjackGame** (игра Блэк джек [**Blackjack**]),
2. **Dealer** (раздающий карты),
3. **Player** (игрок),
4. **Card** (карта),
5. **Deck** (колода карт),
6. **DeckPile** (стопка карт),
7. **Hand** (карты на руках).

На **рис. 3** изображена получившаяся модель предметной области.

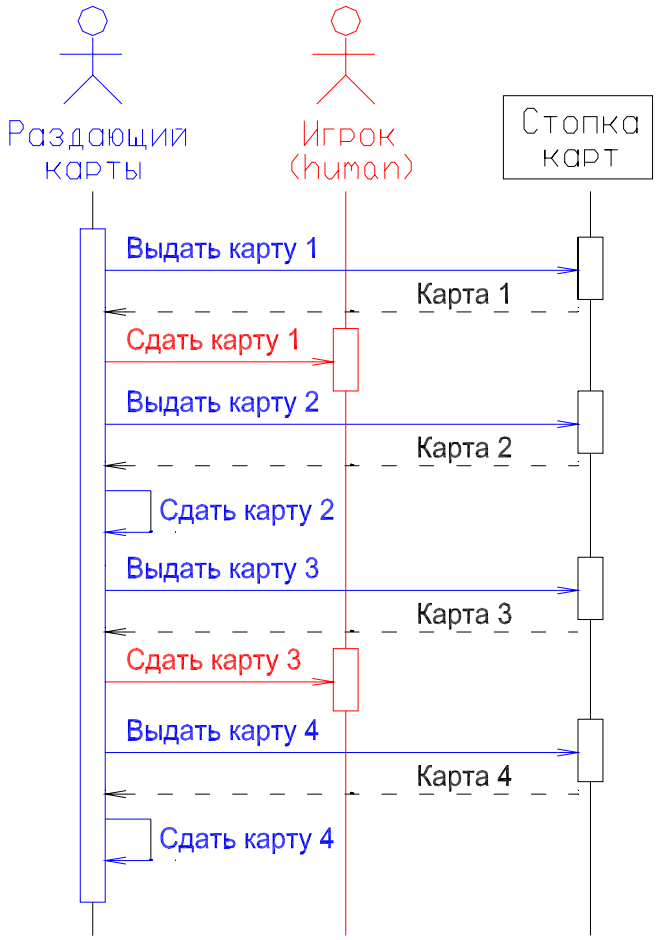


Рис. 1. Диаграмма последовательностей событий при использовании возможности "раздача карт"

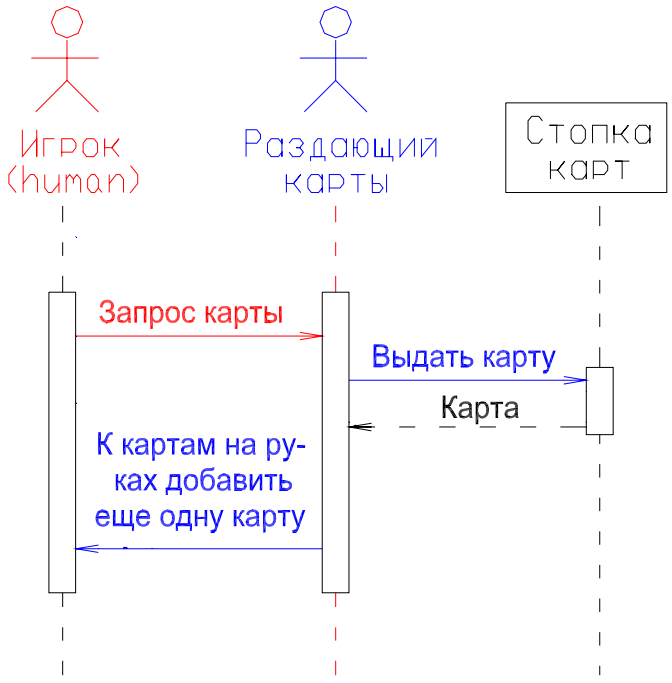


Рис. 2. Диаграмма последовательностей событий при использовании возможности "игрок берет еще одну карту"

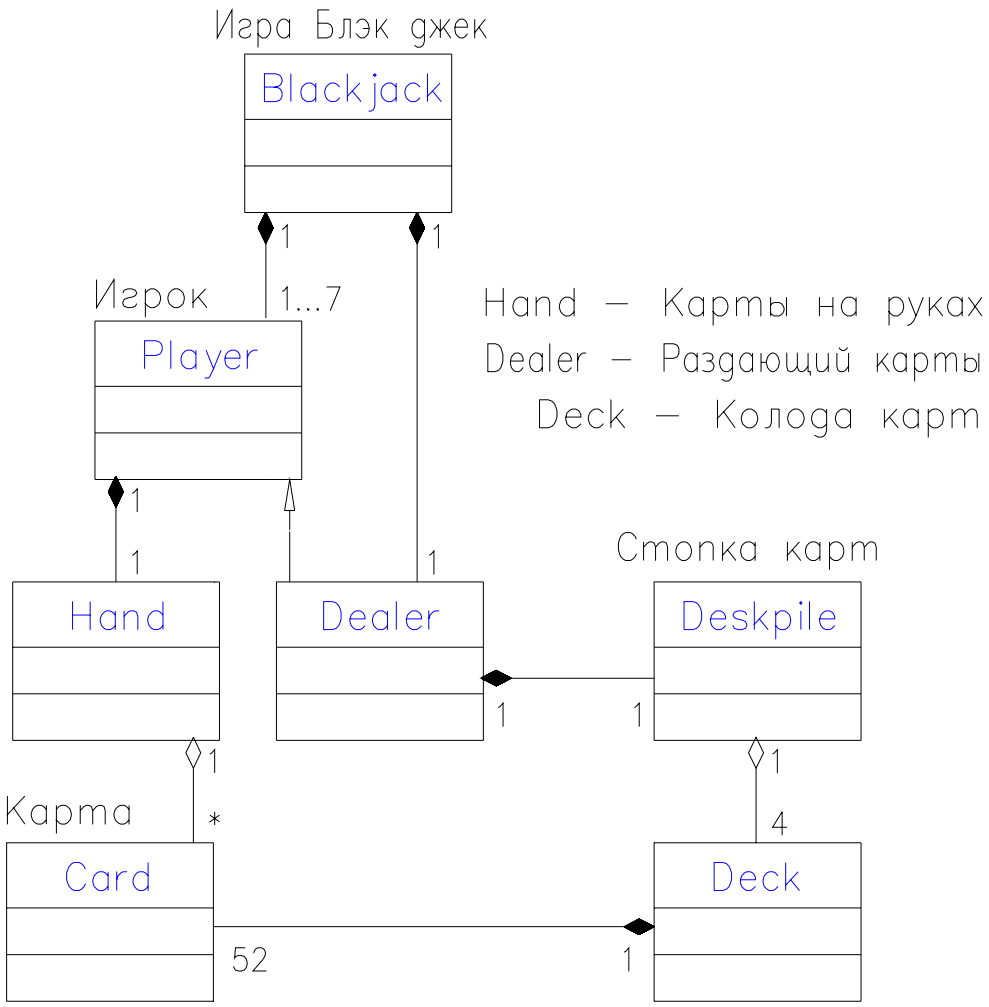


Рис. 3. Модель предметной области игры "Blackjack"

11. Объектно-ориентированное проектирование игры Блэк джек (Blackjack)

(см. Материалы к Лекции 14)

Применив объектно-ориентированное проектирование к результатам выполненного ранее анализа, получим набор основных классов, их функций и описание их взаимодействия и способов получения данных. После этого можно приступить к очередному этапу проектирования и реализации.

11.1 Карточки CRC (Class Responsibility Collaboration)

Карточки **CRC** помогают установить задачи объекта исходя из его назначения.

При моделировании предметной области мы определили стартовый перечень объектов. Пользуясь этим перечнем, нужно с помощью карточек CRC определить: **1) обязанности объектов** и **2) их взаимодействия.**

На рис. 4 – 10 показаны возможные результаты сеанса работы с карточками **CRC**.

11.2. Интерфейс командной строки

В лекции 15, разд. 2 "Объектно-ориентированный подход к программированию пользовательского интерфейса" был рассмотрен шаблон проектирования **MVC** (**Model View Controller – модель/вид/контроллер**). При разработке пользовательского интерфейса игры Блэк джек (**Blackjack**) используется шаблон проектирования **MVC**. Следовательно, чтобы пользователь мог вводить команды и видеть, как протекает игра, нужно к объекту **Player** (Игрок) добавить механизм наблюдения, а также объект **Console** (Консоль).

Поскольку в программе есть только одна консоль (**Console**), естественно применить шаблон одноэлементного [одноточечного] множества (**Singleton**). При этом надежность программы, так как ответственность за назнпчение и отмену доступа к экземпляру возлагается неоредственно на сам объект. Это гарантирует, что будет создан только один экземпляр, и кроме того гарантируется, что будет всего лишь одна точка доступа к экземляру.

Игра Блэк джек (**Blackjack**)	
Назначение	**Объекты, обеспечивающие выполнение назначения**
Создает игроков	Игрок (**Player**)
Создает карты на руках	Карты на руках (**Hand**)
Создает раздающего карты	Раздающий карты (**Dealer**)
Создает стопку карт	Колода (**Deck**), стопка карт (**Deckpile**)
Соединяет всех игроков, раздающего карты и консоль	Игрок (**Player**), раздающий карты (**Dealer**), консоль (**Console**), стопка карт (**Deckpile**)
Начинает игру	Раздающий карты (**Dealer**)

Рис. 4. Карточка CRC игры Блэк джек (**Blackjack**)

Колода (Deck)	
Добавляет себя к стопке карт (**Deckpile**)	Стопка карт (**Deckpile**)
Строит **52** карты	Карта (**Card**)

Рис. 5. Карточка CRC для объекта **Deck** (колода карт)

Карта (Card)	
Хранит масть	Масть (**Suit**)
Хранит ранг	Ранг (**Rank**)
Отображает себя	Ранг (**Rank**), масть (**Suit**), консоль (**Console**)
Хранит и переключает состояние: **лицом вверх или вниз**	

Рис. 6. Карточка CRC для объекта **Card** (карта)

Игрок (Player)	
Хранит карты на руках	Карты на руках (**Hand**)
К картам на руках добавляет новые	Карты на руках (**Hand**), карта (**Card**)
Принимает решение: **брать карту или остановиться**	Карты на руках (**Hand**)
Сообщает раздающему карты: **игру сделал**	Раздающий карты (**Dealer**)
Обновляет наблюдателей	Наблюдатель игрока (**Player Listener**)
Отображает себя	Карты на руках (**Hand**), консоль (**Console**)
Обнаруживает обанкротившиеся карты на руках	Карты на руках (**Hand**)

Рис. 7. Карточка CRC для объекта **Player** (игрок)

Раздающий карты (Dealer) **наследует от игрока (Player)**	
Передает ход **игроку**	Игрок (**Player**)
Передает очередь следующему игроку (**или себе**)	Игроки (**Player**), включая **и себя**
Отслеживает игроков в ходе игры	Игрок (**Player**)
Начинает новую игру	
Сдает карты	Стопка карт (**Deckpile**), карта (**Card**), игрок (**Player**) **и сам**
Решает брать еще или остановиться	
Сообщает игрокам, когда они могут играть	Игрок (**Player**) и сам

Рис. 8. Карточка CRC для объекта **Dealer** (раздающий карты)

Карты на руках (Hand)	
Хранит карты	Карта (**Card**)
Добавляет карты себе	Карта (**Card**)
Устанавливает себя в начальное состояние	
Переворачивает все карты	Карта (**Card**)
Отображает себя	Карта (**Card**), консоль (**Console**)
Вычисляет свою сумму очков	Карта (**Card**), ранг (**Rank**)
Обнаруживает разорение	Карта (**Card**)

Рис. 9. Карточка CRC для объекта **Hand** (карты на руках)

Стопка карт (Deckpile)	
Принимает карты на хранение	Карта (**Card**)
Тасует карты	Карта (**Card**)
Сдает карты	Карта (**Card**)
Собирает карты	Карта (**Card**)
Устанавливает себя в начальное состояние	Карта (**Card**)

Рис. 10. Карточка CRC для объекта **Deckpile** (стопка карт)

11.3. Объектная модель игры Блэк джек (Blackjack). *В целом*, девять классов и два интерфейса образуют полную модель классов игры Блэк джек (Blackjack). Эта модель изображена на **рис. 11.**

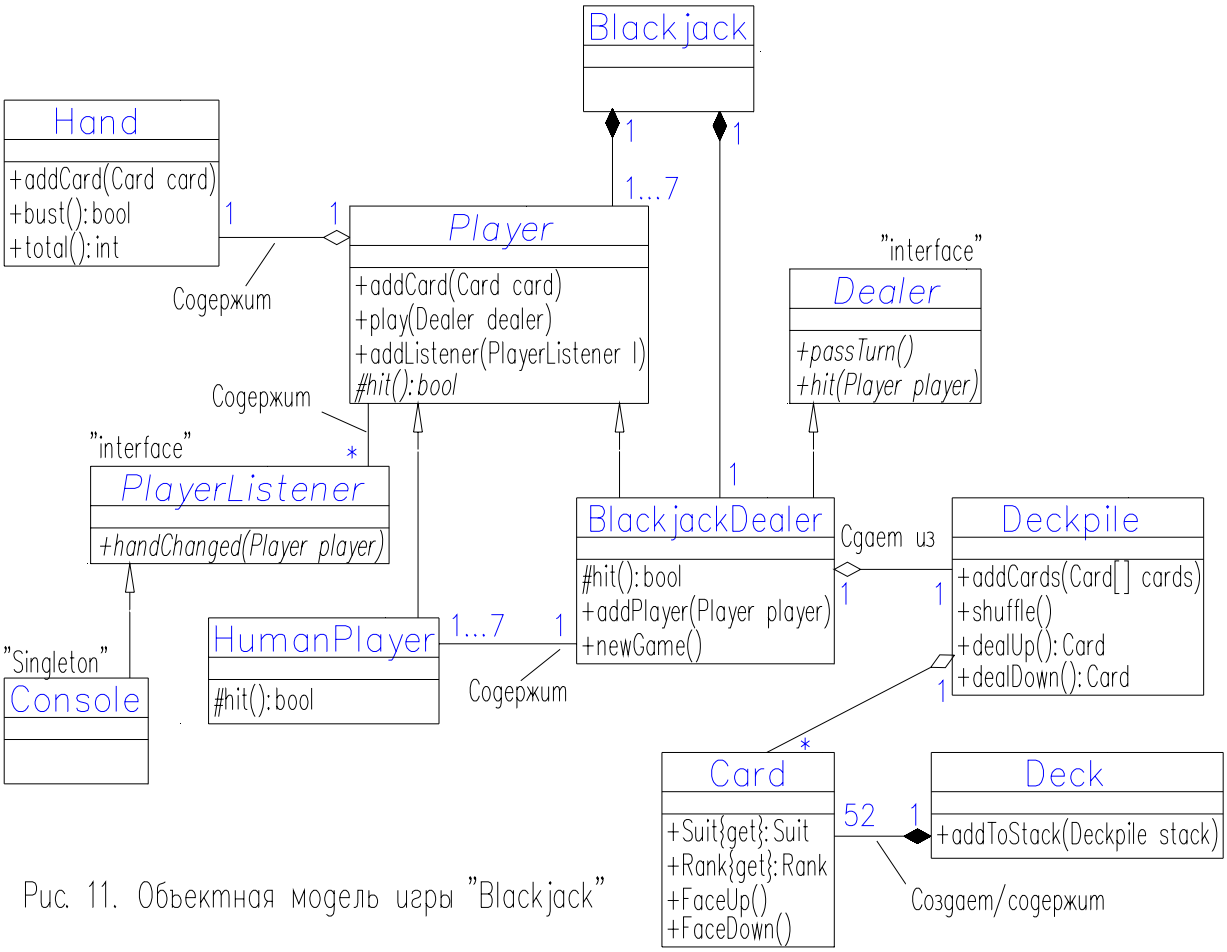


Рис. 11. Объектная модель игры "Blackjack"

Далее приводится описание реализации этой модели.

12. Реализация игры Блэк джек (Blackjack)

На **рис. 11** приведена **основная часть** реализации объектной модели (**см. рис. 13**).

12.0. Листинге 0. Card.cs. Класс Card

Класс **Card** (листинге 0) реализован почти так же, как класс **Card**, описанный в **Практ. зан. 3, Дополн. 1, раздел 3 "Шаблоны продвинутого проектирования"**. Класс **Rank** был незначительно изменен. В листинге 1 представлена новая реализация класса **Rank**.

Константы в классе **Rank** усовершенствованы и содержат числовые значения карт. В том классе сделаны и другие изменения. Теперь этот класс содержит общий не поддающийся изменению (**readonly**) список (**ArrayList**). Этот список заменил поддающийся изменениям массив, который использовался в **Практ. зан. 3, Дополн. 1, раздел 3 "Шаблоны продвинутого проектирования"**. Использование неподдающегося изменению (**readonly**) списка (**ArrayList**) предотвращает случайное изменение списка (**ArrayList**) перечисления.

12.1a. Листинг 1a. Rank.cs. Класс Rank

12.1b. Листинг 1b. Suit.cs. Класс Suit

12.2. Класс Deck и Deckpile

Класс **Deck** претерпел значительные изменения и сильно отличается от класса **Deck**, описанного в **Практ. зан. 3, Дополн. 1, раздел 3 "Шаблоны продвинутого проектирования"**. В листинге 2 представлена новая реализация этого класса.

Листинг 2. Deck.cs. Класс Deck

Объект класса **Deck** имеет методы, позволяющие создавать объекты класса **Card** и добавлять себя в стопку карт (**Deckpile**). Класс **Deckpile** представлен в листинге 3.

12.3. Класс Deckpile

Класс **Deckpile** имеет методы, позволяющие: **1)** тасовать карты (объекты **Card**), **2)** раздавать их и **3)** добавлять себе новые карты. В отличие от первоначальной версии, класс **Deckpile** содержит ссылки на все созданные им карты. Благодаря этому он без труда может извлечь все карты (объекты **Card**) и восстановить свое начальное состояние. Хотя такая модель не полностью соответствует тому, что происходит в реальном мире, она значительно упрощает управление картами.

Важно понять значение изменений, приведенных в листинге. Оба класса **Deck** и **Deckpile** реализуют только возможности, которые абсолютно необходимы в игре. Они не предоставляют дополнительных функций просто так, "на всякий случай, а вдруг когда нибудь они нам понадобятся". Невозможно предвидеть будущее, поэтому не стоит добавлять дополнительные возможности до тех пор, пока вы не убедитесь в том, что они действительно нужны и будут использоваться.

Попытки предусмотреть в программе все "а что если..." часто выдают программиста, **плохо** знакомого с объектно-ориентированным подходом.

Листинг 3. Deckpile.cs. Класс Deckpile

Классы Player (листинг 5) **и HumanPlayer** (листинг 6)

Класс **Player** основывается на классе **Hand**. Класс **Hand** представлен в листинге 4.

12.4. Листинг 4. Hand.cs. Класс Hand

Объект класса **Hand** может: **1)** добавлять себе карты, **2)** восстанавливать свое начальное состояние, **3)** переворачивать свои карты, **4)** подсчитывать очки и **5)** представлять себя в виде строки

(**String**). Обратите внимание на то, что класс **Hand** считает, что туз приносит только **11** очков. В последующих итерациях будет добавлена возможность учитывать туз как **11** или **1** очко.

12.5. Листинг 5. Player.cs. Абстрактный класс Player

При объектно-ориентированном подходе игроки (объекты **Player**) сообщают раздающему карты (объекту **Dealer**) о том, что они сыграли.

12.6. Листинг 6. HumanPlayer.cs (играющий человек). Производный класс HumanPlayer: Player

В абстрактном классе **Player** определены поведения и свойства, общие для игроков (классы player) и раздающих карты (классы **Dealer**). Эти поведения включают манипуляции с картами на руках у игроков, отслеживание наблюдателей (**PlayerListener)** и действия, выполняемые игроком, когда наступает его очередь.

В классе **Player** определен абстрактный метод **public bool hit()**. В течение игры объект базового класса **Player** может вызывать этот метод, чтобы определить, брать еще карты или остановиться. Производные классы могут реализовать этот метод для того, чтобы определить свое собственное поведение. Например, **HumanPlayer** спрашивает у пользователя, когда брать еще карты, а когда остановиться. После того как игрок (**Player**) сделал свою игру, он сообщает об этом раздающему карты (**Dealer**), вызывая метод **passTurn()** раздающего карты (**Dealer**) (передать). Получив это сообщение, раздающий карты (**Dealer**) сообщает следующему игроку (**Player**), что тот может играть.

12.7. Листинг 7. Dealer.cs. interface Dealer. Раздающий карты

Раздающий карты (**Dealer**) – это интерфейс, в котором определены дополнительные методы, используемые раздающим карты (**Dealer**).

12.8. Листинг 8. BlackjackDealer.cs . Производный класс BlackjackDealer: Player, Dealer

Класс **BlackjackDealer** является наследником класса **Player**, поскольку раздающий карты (**Dealer**) — это тоже игрок (**player**). Кроме поведений игроков, предусмотренных в классе **Player**, раздающий карты: **1)** взаимодействует с игроками, **2)** раздает им карты, а также **3)** информирует игрока о том, что пришла его очередь играть. Когда игрок (**Player**) вызывает метод **passTurn()** раздающего карты (объект **Dealer**), это означает, что в игру вступает следующий игрок. **Рис. 12** демонстрирует взаимодействие между **раздающим карты** и **игроками**.

BlackjackDealer применяет метод **stopPlay()** для прекращения игры. У раздающего карты также есть метод **hit()**, который возвращает **true (истина), если у раздающего меньше 17 очков (< 17)** и **false (ложь) — если больше или равно 17 (>= 17)**.

12.9. Листинг 9. Blackjack.cs . Класс Blackjack

Класс **Blackjack** создает: **1)** раздающего карты (класс **Dealer**), **2)** карты на руках (класс **Hands**), **3)** стопку карт (класс **Deckpile**), **4)** колоды (классы **Deck**) и **5)** соединяет их вместе. Новая игра начинается после того, как раздающий карты (класс **Dealer**) получает команду начать новую игру.

12.10. Листинг 10. Console.cs . Производный класс Console: PlayerListener

Класс **Console** — одноэлементное [одноточечное] множество (шаблон проектирования **Singleton**). Этот класс: **1)** предоставляет доступ к командной строке; **2)** он также принимает информацию от игроков (классы **Player**) и **3)** распечатывает на экране состояние игроков при любом его изменении.

12.11. Листинг 11. PlayerListener.cs . interface PlayerListener

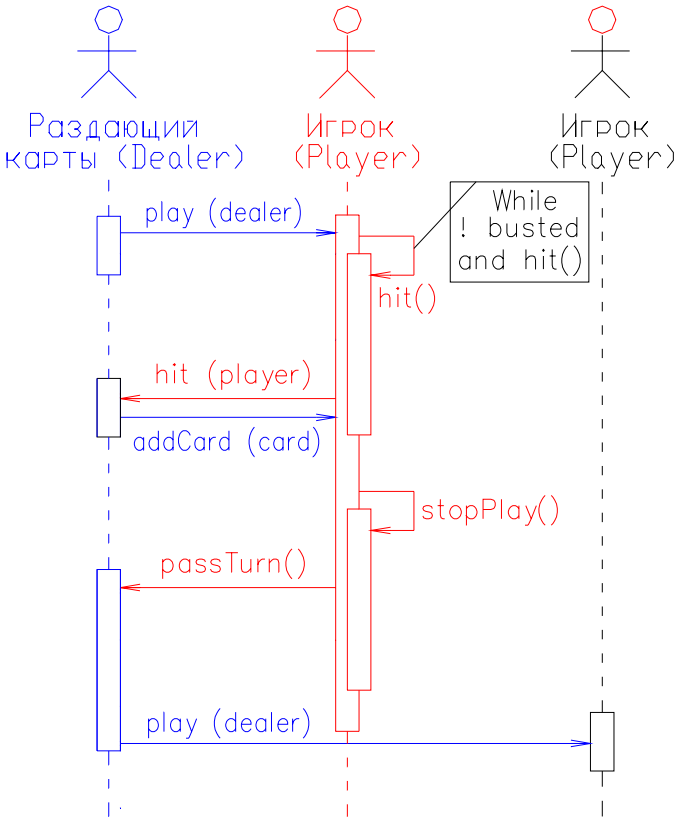


Рис. 12. Взаимодействие между раздающим карты и игроками

Результаты работы программы за сеанс игры

Human: 8♥
Dealer: 3♥
Human: 8♥ J♣
Dealer: 3♥ Hidden

[**H**]it or [**S**]tay

s

Dealer: 3♥ 2♥
Dealer: 3♥ 2♥ 6♥
Dealer: 3♥ 2♥ 6♥ 4♠
Dealer: 3♥ 2♥ 6♥ 4♠ 9♦

Резюме

Проанализировали, разработали и реализовали базовую игру Блэк джек (**Blackjack**) [только 1-я итерация, см. Содержание]. Благодаря использованию итерационного подхода **были быстро получены ощутимые результаты**. На следующих итерациях расширяются функциональные возможности игры Блэк джек (**Blackjack**) [эти итерации предлагается выполнить студентам самостоятельно].

Код, соответствующий объектной модели игры (см. рис. 11) не приводится. Студентам предлагается разработать его самостоятельно.

Основная цель данного материала проиллюстрировать ОО анализ и ОО проектирование.

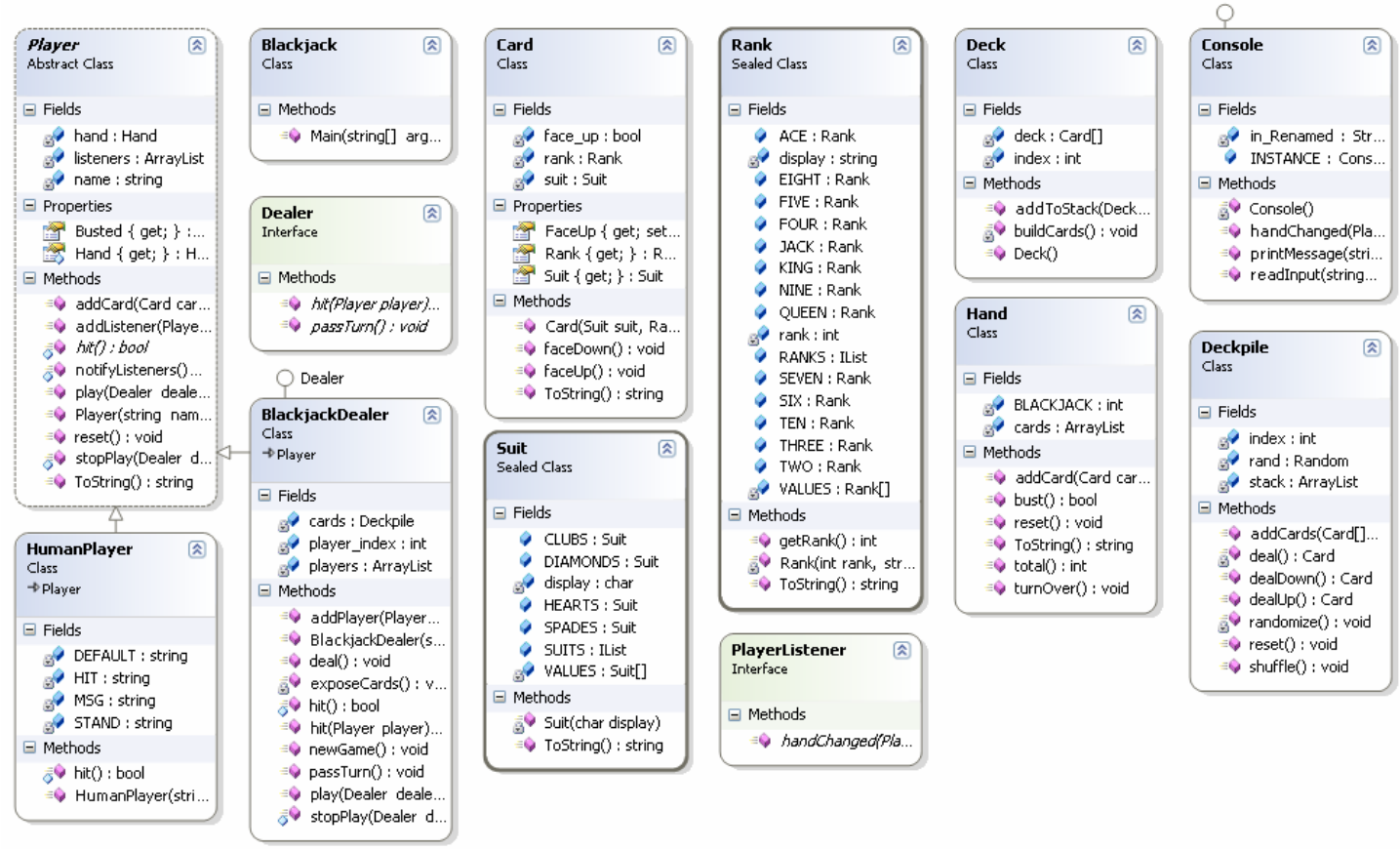


Рис. 13. Диаграмма классов (итерация 1)

Контрольные вопросы

1. Назовите два шаблона проектирования, с которыми вы сегодня познакомились. Где эти шаблоны использовались?
2. Найдите в исходных кодах один пример полиморфизма.
3. Найдите в исходных кодах один пример наследования.
4. Как объект класса **Deck** (колода карт) изолирует (инкапсулирует) свои карты?
5. Как объекты классов **BlackjackDealer** и **HumanPlayer** реализуют полиморфизм?

Ответы на вопросы

1. **PlayerListener** является примером шаблона наблюдателя.

Console – одноэлементное [одноточечное] множество. Оно реализует шаблон одноэлементного [одноточечного] множества.

Rank (Ранг) **и Suit** (Масть) реализуют шаблон перечисления (с сохранением типа).

2. При обработке **HumanPlayer** класс **BlackjackDealer** рассматривает его как игрока (**Player**), используя при этом полиморфизм. Можно создать неодушевленных игроков и **BlackjackDealer** будет знать, как играть с ними в игру Блэк джек (**Blackjack**).

3. **Player** / BlackjackDealer / **HumanPlayer** является примером иерархии наследования.

4. Колода (**Deck**) полностью инкапсулирует хранимые в ней карты (экземпляры класса **Card**). Колода (**Deck**) не содержит каких-либо средств доступа или механизмов установки. Вместо этого колода (**Deck**) сама добавляет свои карты (экземпляры класса **Card**) в стопку карт (класс **Deckpile**).

5. **BlackjackDealer** и **HumanPlayer** используют полиморфизм, предоставляя свои собственные версии метода **hit()**. Когда метод **play()** обращается к методу **hit()**, то изменения в поведении метода **play()** определяются реализацией вызываемого метода **hit()**.

Задание студентам

1. Разработать код в соответствии с ОО анализом и проектированием, выполненными в разделах 9…12 (**ит-я 1**).
2. Выполнить ОО анализ, ОО проектирование и разработку кода на каждой из итераций **2, 3 и 4**. На итерации 4 при разработке **GUI** реализовать шаблон **MVC** (**М**одель – **В**ид – **К**онтроллер) (см. с.8, 9)

Литература

Базовый учебник

1. Мейер Б. Объектно-ориентированное конструирование программных систем. М.: Русская Редакция, 2005.

Основная

2. Буч Г., Якобсон А., Рамбо Дж. **UML**. С.-Петербург: Питер, 2006.
3. Гамма Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования. С.-Петербург: Питер, 2006.
4. Забудский Е.И. Учебно-методический комплекс дисциплины «Объектно-ориентированный анализ и программирование». М.: Кафедра ОИиППО ГУ-ВШЭ, 2007,
   Internet-ресурс – http://new.hse.ru/C7/C17/zabudskiy-e-i/default.aspx .
5. Кватрани Т. Визуальное моделирование с помощью Rational Rose 2002 и UML. М.: Вильямс, 2003.
6. Лафоре Р. Объектно-ориентированное программирование в C++. С.-Петербург: Питер, 2005.
7. Троелсен Э. C# и платформа **.**NET. С.-Петербург: Питер, 2006.
8. Синтес А. Освой самостоятельно объектно-ориентированное программирование за 21 день. Москва; С.-Петербург; Киев: Вильямс, 2002.

Дополнительная – Internet-ресурсы

9. Новые книги раздела **C#** – http://books.dore.ru/bs/f6sid16.html .
10. **C#** и ***.NET*** по шагам – http://www.firststeps.ru .
11. ***UML*** – язык графического моделирования – http://www.uml.org/ .
12. ***NUnit, JUnit*** – каркасы тестирования для испытания классов – http://www.junit.org , http://www.nunit.org .
13. Пакет объектного моделирования ***Rational Rose*** – http://www-306.ibm.com/software/rational/ .

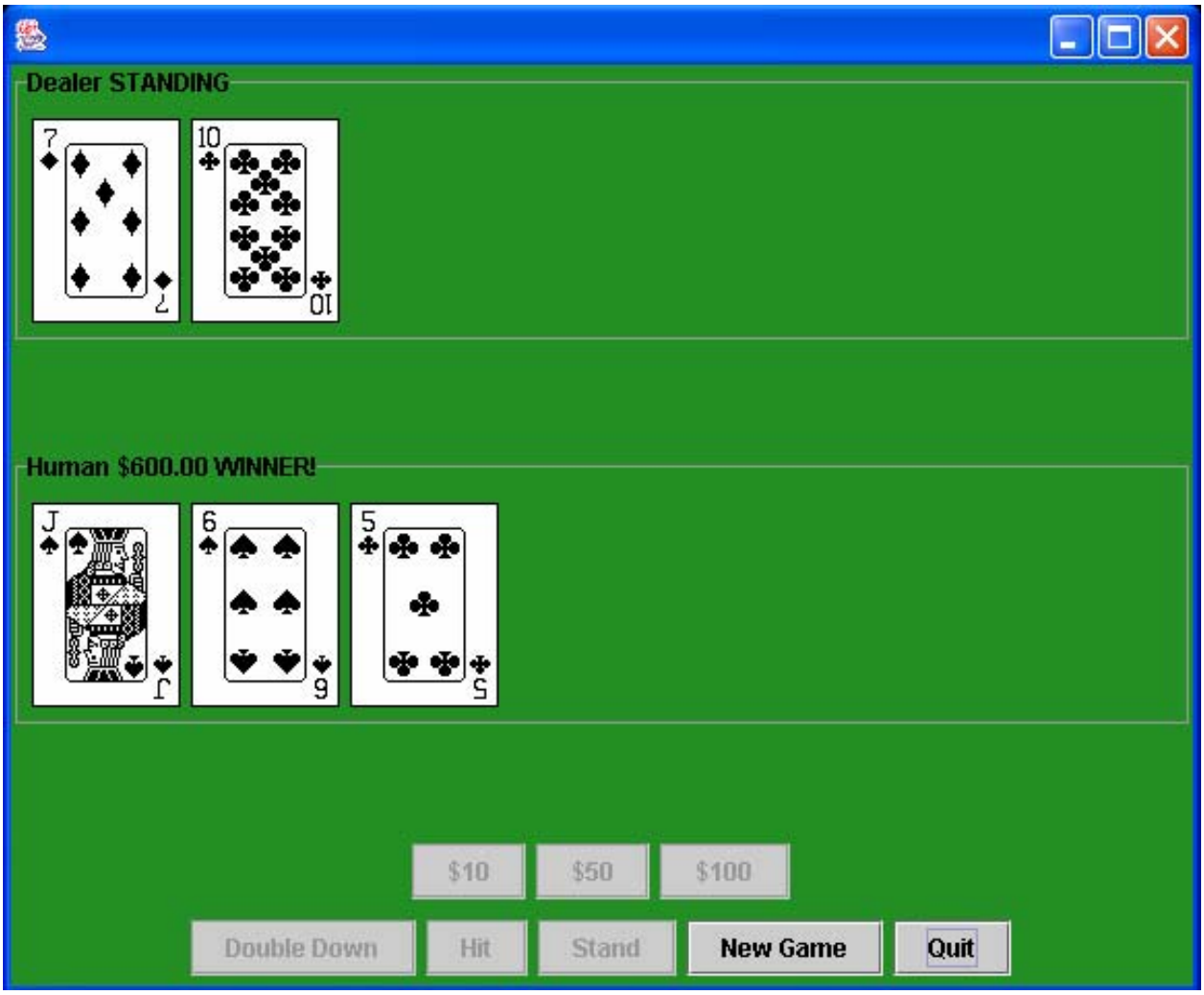
13*a*. Steve Burbeck "Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC)" – http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html .

13*b*. Информация о языке C# и платформе .NET – http://msdn2.microsoft.com/ru-ru/default.aspx .

13*c*. Информация о языке C# и платформе .NET – http://www.gotdotnet.com http://www.gotdotnet.ru

Дополнительная – книги

14. Мэтт Вайсфельд. Объектно-ориентированный подход: Java, **.**NET, C++. М.: КУДИЦ-ОБРАЗ, 2005.
15. Дж. Кьоу, М. Джеанини. Объектно-ориентированное программирование. С.-Петербург: Питер, 2005.
16. Уоткинз Д., Хаммонд М, Эйбрамз Б. Программирование на платформе .NET.: М.: Вильямс, 2003.



Возможная реализация GUI: результат выполнения итераций 2, 3 и 4 (см. с. 8, 9)