Проект № 1: Blackjack Solitaire

Данный проект направлен на закрепление знаний и навыков по следующим темам:

--- *принципы программирования (в частности, DRY, «наименьшее удивление», «минимальное зацепление» и «максимальная связность»);*

--- *реализация объектно-ориентированного подхода на основе заявленной спецификации;*

--- *разработка архитектуры приложения с применением UML.*

Конечный результат

Слушатель создаст карточную игру Blackjack Solitaire в консольном варианте (без графики).

Рекомендуем сначала потренироваться с игрой в ее онлайн-версии:

https://www.solitairenetwork.com/solitaire/blackjack-square-solitaire-game.html

Особое внимание следует уделить правилам подсчета ОЧКОВ и БАЛЛОВ за проведенный раунд (да-да, БАЛЛЫ и ОЧКИ --- разные понятия. Все нужные пояснения даны ниже).

Максимально возможное кол-во ОЧКОВ в одном ряду или колонке карт равно **21**. Если просуммированные ОЧКИ в каком-то ряду или колонке окажутся равны **22** и более, такая ситуация называется «перебор» и тогда все очки в этом ряду (или колонке) сгорают, т.е. обнуляются.

В нашей игре карты выкладываются в 4 ряда и 5 колонок. Карты одна за другой берутся из однократно перетасованной колоды и кладутся в любую свободную ячейку на столе. Однажды выложенную карту передвигать нельзя. Есть 4 «мусорные» ячейки, куда можно выбросить текущую карту, если по каким-то причинам она мешает; итого можно избавиться максимум от 4 карт. Как только все 16 «рабочих» ячеек на столе будут заполнены, раунд игры заканчивается и код вычисляет игровые БАЛЛЫ.

Как подсчитывать очки и баллы

У каждой карты есть присущая ей ценность в ОЧКАХ, причем масть роли не играет. Если карта без картинки, ее ценность (ОЧКИ) определяется числом на лицевой стороне (т.е., от 2 до 10 ОЧКОВ). Джокера в колоде нет. Валеты, дамы и короли оцениваются одинаково, по 10 ОЧКОВ. Блэкджек возможен только в крайней правой или крайней левой колонке.

Сама сложная карта — это туз. Его ценность может быть равна 11 или 1, и выбирается она так, чтобы получить как можно больше ОЧКОВ без перебора в том ряду и/или колонке, где находится этот туз; возможна даже ситуация, когда один и тот же туз может «стОить» по-разному, скажем, по горизонтали 1, а по вертикали 11, или наоборот. Главное, подобрать такое его значение, чтобы получить как можно больше ОЧКОВ --- но без перебора!

В архиве с заданием есть PNG-файлы, где показаны конкретные сыгранные комбинации карт и правильно подсчитанные баллы; эти файлы можно использовать для тестирования ваших алгоритмов. Напр., файл **C_2bj_67.png** означает, что в этой комбинации есть два блэкджека, а число игровых баллов равно 67.

ОЧКИ	**БАЛЛЫ**	**Критерий**
ситуация «блэкджек»	10	**ОБЕ** карты в колонке дали в сумме 21 очко
21	7	Весь ряд / колонка дает 21 очко
20	5	Весь ряд / колонка дает 20 очков
19	4	Весь ряд / колонка дает 19 очков
18	3	Весь ряд / колонка дает 18 очков
17	2	Весь ряд / колонка дает 17 очков
16 и менее	1	Весь ряд / колонка дает 16 очков или менее
ситуация «перебор»	0	Весь ряд / колонка дает 22 очка или более

Что именно должна делать программа:

1. Играется один-единственный раунд. В результате этого раунда будут заполнены все 16 «рабочих» ячеек на столе (это произойдет обязательно), и от 0 до 4 «мусорных» ячеек (а это уже как сам игрок решит). Затем вычисляются БАЛЛЫ, их значение выводится в консоль, после чего программа завершает работу.

2. Раунд начинается с тасования колоды, затем выдается первая карта. «Рабочие» ячейки стола расположены и пронумерованы как показано ниже (схема дана примерная, выравнивание не требуется; главное --- сохранить идею этой компоновки):

1	2	3	4	5
6	7	8	9	10
	11	12	13	
	14	15	16	

   «Мусорные» ячейки имеют номера 17, 18, 19 и 20. На схеме выше их нет, да и показывать их на столе не обязательно.

   Программа должна каждый раз сообщать игроку, какая именно выпала очередная карта, а игрок должен указать номер ячейки, куда он хочет эту карту положить.

3. Карта обозначается сочетанием ее ценности (цифрой или буквой) и масти (буквой):

   валет = **J** (от слова JACK), дама = **Q** (QUEEN), король = **K** (KING) и туз = **A** (ACE)

   черви = **H** (HEARTS), пики = **S** (SPADES), бубны = **D** (DIAMONDS), трефы = **C** (CLUBS)

   Например, **KH** означает «король червей», а **3S** --- «тройка пик». Соответственно, каждая «рабочая» ячейка на столе должна быть отмечена либо числом (ее порядковый номер, когда эта ячейка еще не заполнена), либо обозначением той карты, которая в этой ячейке уже хранится (на схеме ниже зеленым показаны свободные ячейки):

1	KH	3	4	5
2D	7	8	9	AS
	11	12	13	
	14	7C	16	

   (Предваряя вопросы: расцвечивать ничего не надо; пусть все будет белым на черном фоне).

4. Когда карта будет положена на стол или в мусор, новая карта выдается автоматически. После заполнения всех 16-ти «рабочих» ячеек раунд завершается.
5. Проверка допустимых значений:
   - Если игрок укажет номер уже заполненной «рабочей» или «мусорной» ячейки, программа должна сообщить об ошибке и пригласить игрока ввести иной номер ячейки для все той же карты. Прочие проверки --- по желанию.
6. Архитектура программы произвольная, но в ней как минимум должны быть следующие классы (плюс «главный» класс, упомянутый в п.7 ниже):
   - класс **Card**, который описывает модель игральной карты
   - класс **Deck**, который описывает модель игральной колоды
   - и класс **BlackjackSolitaire**, который описывает модель игры в целом

   Создание дополнительных классов приветствуется, а UML-диаграммы отлично помогут вам в разработке архитектуры приложения.
7. Главный класс уже готов; вот он:

```
public class BlackjackSolitaireRunner {
    public static void main(String[] args) {
        BlackjackSolitaire bjs = new BlackjackSolitaire();
        bjs.play();
    }
}
```

Более подробная разбивка действий программы:

- Выводим на экран исходное состояние стола.
- Случайным образом тасуем колоду.
- Выдаем карту.
- Просим игрока сделать ход, т.е. он должен ввести номер ячейки, а мы выданную карту туда положим --- но лишь после проверки, является ли номер допустимым! **Подсказка:** проверки лучше делать в отдельных методах, особенно если таких проверок много (отриц.число, буквы вместо числа, слишком большое число и т.д.).
- Выводим на экран текущее состояние (другими словами, после каждого хода игрока мы обновляем табличку).
- Продолжаем перечисленные шаги (выдать карту, получить от игрока номер ячейки, проверить этот номер, поместить карту, обновить табличку о состоянии стола), пока все 16 «рабочих» ячеек не окажутся заполнены.
- Теперь сообщаем игроку, что начинается подсчет баллов, и передаем информацию о текущем состоянии стола в ту функцию, которая вычисляет игровые баллы.
- Наконец, выводим в консоль итоговое число баллов и прощаемся с игроком.

Что именно сдавать на проверку:

КАК МИНИМУМ вот эти четыре файла:

```
Card.java
Deck.java
BlackjackSolitaire.java
BlackjackSolitaireRunner.java
```

Файлы с исходным кодом — **не проект целиком в среде NetBeans / Eclipse / IDEA и т.д., а только исходный код!** — надо упаковать в ZIP с именем (латинскими буквами): **firstname_lastname_bjs.zip** (напр.: **kolya_petrov_bjs.zip**).

ПРОЧИЕ ТРЕБОВАНИЯ и ПРИМЕЧАНИЯ:

Дата сдачи: в любой момент, но не позднее самого последнего дня курса.

Java-платформа: идеально 1.8; допустимо до 1.14 включительно (но не выше!)

Проект выполняется строго индивидуально; советоваться с коллегами можно, но учтите, что код будет пропущен через детектор плагиата.

Если код не скомпилируется у инструктора, проект получит **0 баллов** автоматом независимо от того, насколько гениальным является его алгоритм или сколько миллионов человеко-часов высокоинтеллектуального труда в него вбито. Весь проект без вычетов принесет 100 баллов. Бонусные баллы учитываются отдельно.

За отсутствие комментариев к публичным дата-типам и их компонентам будут сниматься баллы (-1 балл за каждый промах).

За несоблюдение конвенции по именованию: -1 балл за каждый промах.

Сгенерированная **javadoc**-документация на весь проект принесет добавочно 5 баллов, а комплект UML-диаграмм даст еще 10 бонусных баллов.

Если игра упакована не просто в ZIP-архив, а в JAR-файл и при этом успешно запускается стандартной командой (`java -jar kolya_petrov_bjs.jar`) на платформе 1.8, то мы добавим еще 10 баллов.

Приложение должно быть чисто консольным; графический интерфейс, картинки или расцвечивание --- все это не нужно и даже не принесет дополнительных баллов.

Приветствовать игрока, разъяснять ему правила не обязательно (хотя желательно, чтобы интерфейс игры был интуитивно понятным).

Троллить игрока можно, но в гомеопатических дозах; ирония допустима, сарказм ненаказуем, изящная шутка всегда уместна.