TED UNIVERSITY

BlackJack Game

Design Report

Buğra Kaan Aydın

BlackJack Game

30 April 2020

Table of Contents

About the Game

Materials of the Game

Black jack is a famous card game which is played with a 52 cards deck of 4 groups consisting of 13 cards. Groups are:

- Diamond
- Club
- Heart
- Spade

Each group has the one of each of the following cards:

Cards = {ACE, 2, 3, …, 10, JACK, QUEEN, KING}

Each card has a value. The cards with a number have the value of the number on themselves. Royal cards {JACK, QUEEN, KING} have the value 10. ACE has two values: 1 and 11. A player that owns an ACE card may use it for either of the values.

Actors of the Game

There two types of the actors in the game:

- Dealer: only one per game.
- Player: minimum two per game, there is no maximum limit.

Mechanism and the Aim of the Game

The main aim of the game is to have the maximum score after all of the rounds. For this aim, in each turn the aim for a player is to reach 21 or be the player reaching closest to 21

without exceeding it because the player(s) that satisfy this rule gets the bet of that round. If there is more than one player with the same score that satisfies this rule, then the bet is distributed equally among these players.

The game starts with a shuffled deck. Then the first round starts. Each player is given two cards just after every round starts. Then, the first player's turn starts. Each player takes as many cards as they want one by one unless they exceed 21 with their current hand.

A player that exceeds 21 loses that round and can not take any other cards. A player that stops i.e. chooses not to take any other cards, keeps his/her cards until the end of the turn.

At the end of each turn, if any exists, the player that did not lose a round opens their hand, the one(s) with highest score wins that round and gets his/her part of the bet.

After all of the rounds end, the scores of the players are compared and the one with the highest score wins the game. If there are multiple winners, then that is a draw of those players.

Design of the Program

High Level Design

The program should consist of one Graphical User Interface(GUI), and a Background System. The GUI will be created using Swing and will have the components and containers in the low level design part.

Low Level Design

GUI

External Packages: jawax.swing

Internal Packages: gui

Contents of gui package:

- Classes
    - MainFrame extends JFrame
        - Layout : CardLayout
        - It will only have the panels below as its properties.
    - MainMenu, HowToPlayMenu, SettingsMenu, GamePanel extend JPanel
        - MainMenu : will provide a transition to other menu and game panels, will use GridBagLayout
        - HowToPlayMenu : will provide instructions and information about the game, and will use BorderLayout.
        - SettingsMenu : will provide settings of the game, mute and private mode, and will use BoxLayout.
        - GamePanel : will coordinate the game, will use GridBagLayout.
            - Labels for the usernames, scores, card images, state of the game, round and current turn, and buttons for the users to make their moves in their turn.

Background System

External packages: No external package.

Internal packages: start, actors, materials.

Contents of the start package:

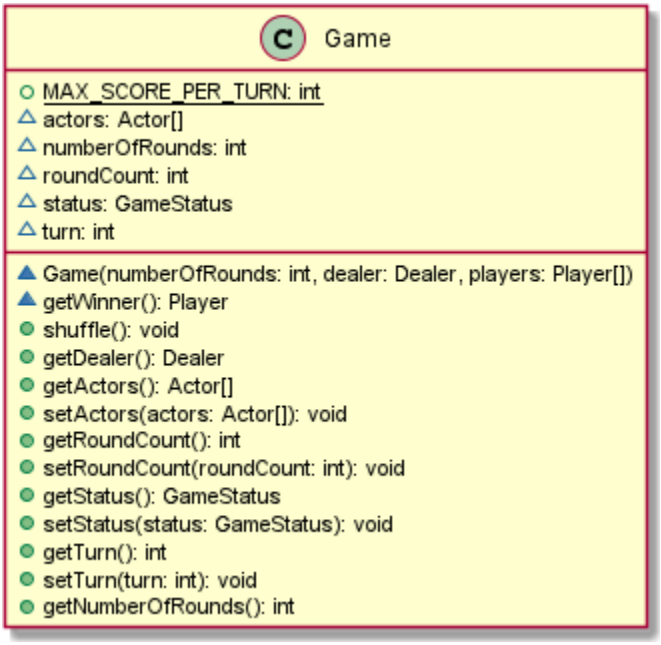
- Enumerations
    - GameStatus{PLAYING, FINISHED}: possible states of the game.
        - Class(Enum) Diagram:

- Classes
    - Game
        - This class will be the main class to coordinate everything in the game. Briefly, it will keep the information about the actors(dealer and players) and cards in the game, shuffle the deck, and control the turns of the players.
        - Properties:
            - Actor[] actors: will keep the dealer(0-th index) and players(1-st index to n-th index),
            - int numberOfRounds: will keep how many rounds will be in the game.
            - int roundCount: will keep how many rounds have already passed.

- Status status : will keep the state of the game.
- int turn: will keep the index of the player playing.

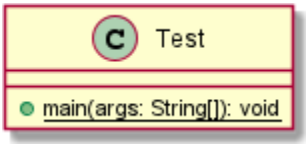
- Methods:
  - Game(Actor[], int): The only constructor the class will have.
  - Person getWinner() : if the finished, returns the highest score player(s), null otherwise.
  - void shuffle() : shuffles the deck in the dealer.
  - Encapsulation methods(getters and setters) for every property.

- Class Diagram:

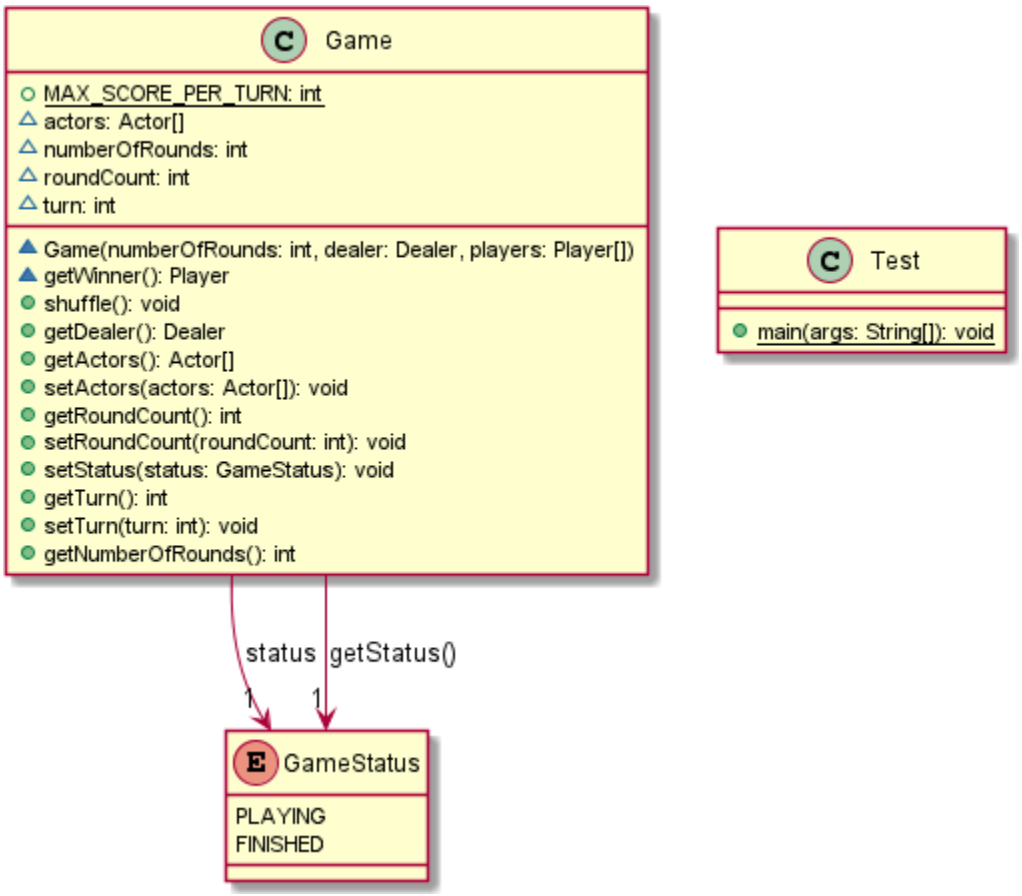


- Test
  - This class will only have a main method and will be used for the console application. It will simply create a game. It will be used only in development, not in deployment versions.

- Methods
    - main(String[] args): the method to start the console application.
- Class Diagram:



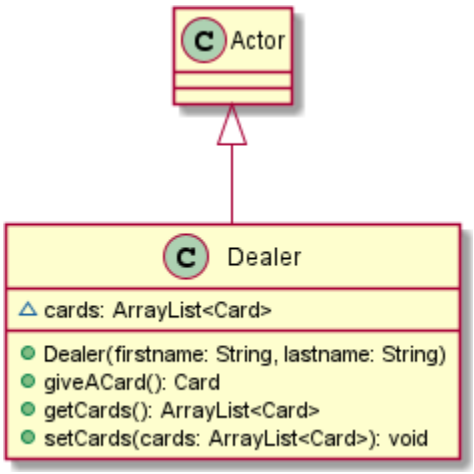
- Class (Package) Diagram - with relations:



Contents of the actors package:

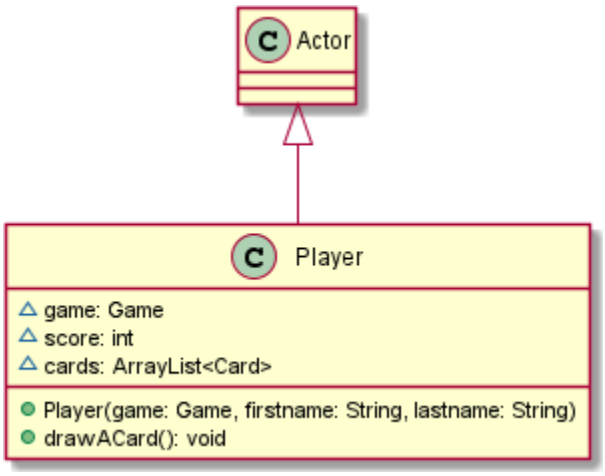
- Classes
    - Abstract Actor
        - This class will simulate the actors. It will not be instantiated.
        - Properties:
            - String firstname, lastname : personal information
        - Methods:
            - Encapsulation methods(getters and setters) for every property.
        - Class Diagram:

- Dealer
  - This class will simulate the dealer of the game. There will be only one dealer per game. Therefore, this class will have only one instance.
  - Properties:
    - ArrayList<Card> deck : the deck of cards to play.
  - Methods:
    - giveACard() : Gives a card.
    - Encapsulation methods.
  - Class Diagram:

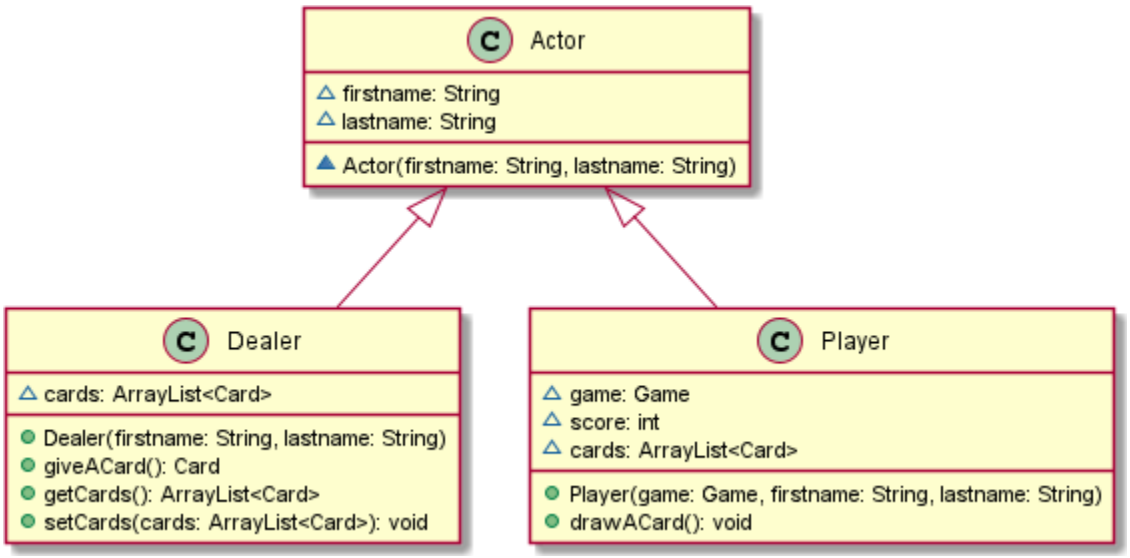


- Player
  - This class will simulate the players of the game.
  - Properties:
    - String nickname: an alias for the users chosen by themselves.
    - int score: total score of the player in the rounds so far.
    - ArrayList<Card> hand : the cards in the player's hand.

- Methods:
  - void drawACard() : person wants a card from the dealer.
  - Encapsulation methods(getters and setters) for every property.
- Class Diagram:



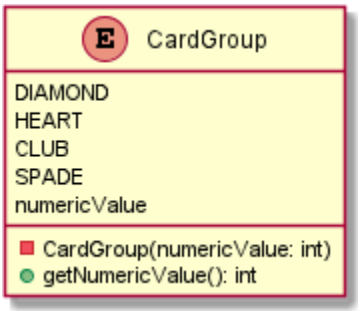
- Class (Package) Diagram - with relations:



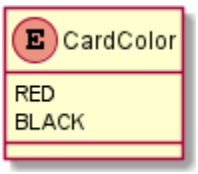
Contents of the materials package:

- Enumerations
  - Group
    - {DIAMOND(1), HEART(2), CLUB(3), SPADE(4); int numericValue;

      Group(int numericValue); getNumericValue} : possible groups of a card.
    - Class(Enum) Diagram:



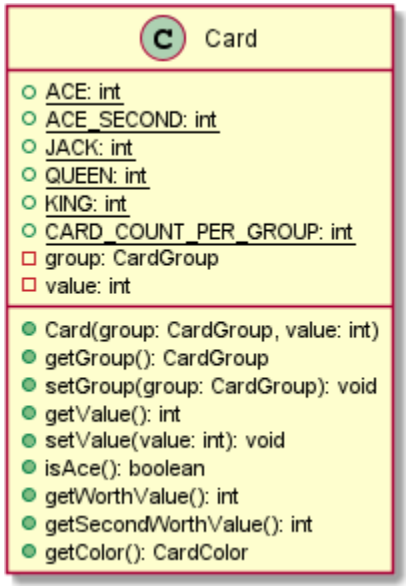
- CardColor{RED,BLACK} : possible colors of a card.
  - Class(Enum) Diagram:



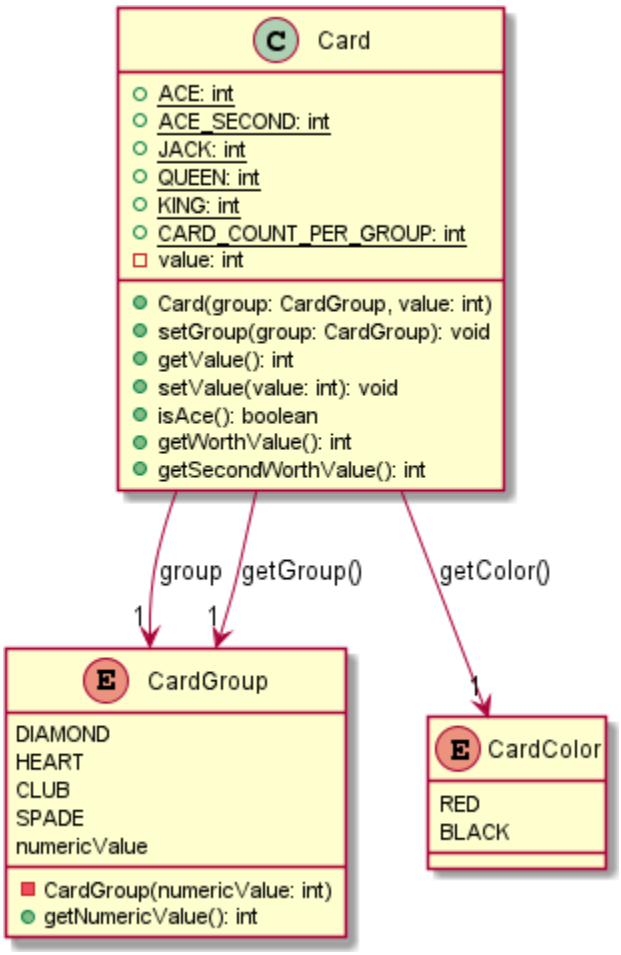
- Classes
  - Card
    - This class will simulate individual cards of the game.
    - Properties:
      - static final int ACE=1, ACE_SECOND=11, JACK=11, QUEEN=12, KING = 13
      - Group group : will keep the group of a card.
      - int value : will keep value on the card.(For JACK = 11, QUEEN = 12, KING = 13).
    - Methods:
      - Encapsulation methods(getters and setters) for every property.
      - int getWorthValue(): gets the worth value of every card. See Materials of the Game for more information. This method returns 1 for Ace.
      - int getSecondWorthValue(): gets the second worth value of the card. If the card is ACE, returns 11, otherwise same with getWorthValue() method.

- CardColor getColor(): gets the color according to the group of the card. Returns RED for HEART and DIAMOND groups, otherwise BLACK.

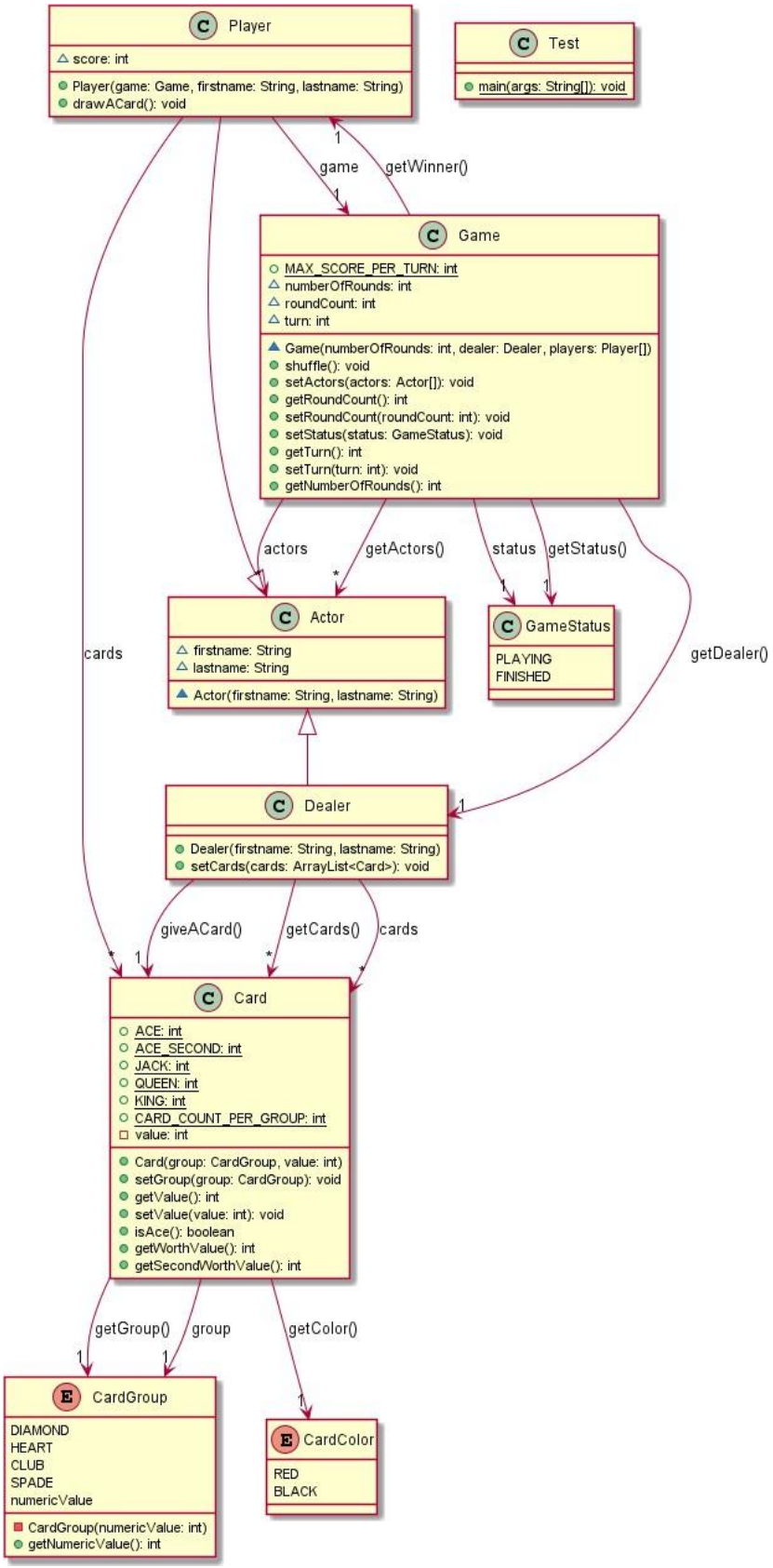
- Class Diagram:



- Class (Package) Diagram - with relations:



Relations between all classes



APPENDIX A

Here you can find the images that will be used in the game.

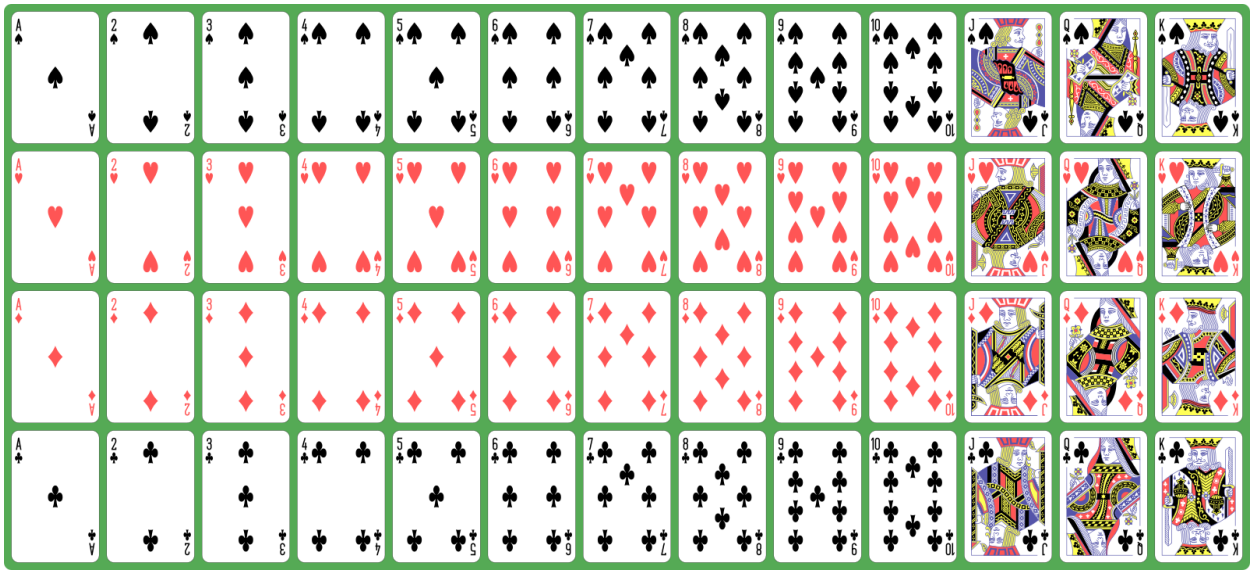


Image 1: Cards