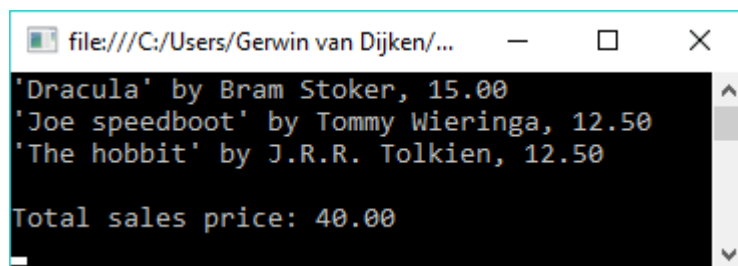


## Assignment 1 – Bookshop

A bookshop sells books. For each book, the title, author and price is recorded. The retailer regularly wants an overview of the stock and also of the total sales price (of the entire stock).

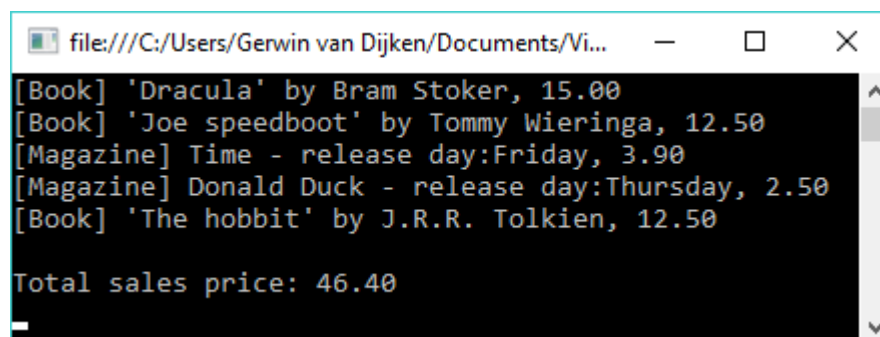
- Create a class Book (with fields, a constructor with parameters, and a Print method) and a class BookStore (containing a list, an Add method and a PrintCompleteStock method). Books can only be added with the Add method.
- In de Start method, create a bookstore and add some books to it (using the Add method). You may create these books with hardcoded values (titles, ...), you don't have to ask information from the user. After adding all books, print the complete stock of the bookstore.



```
file:///C:/Users/Gerwin van Dijken/...  
'Dracula' by Bram Stoker, 15.00  
'Joe speedboot' by Tommy Wieringa, 12.50  
'The hobbit' by J.R.R. Tolkien, 12.50  
Total sales price: 40.00
```

After some time, the retailer decides to sell also magazines (besides books). For each magazine, the title, price and day of release is recorded. This means that a book and a magazine have some fields in common.

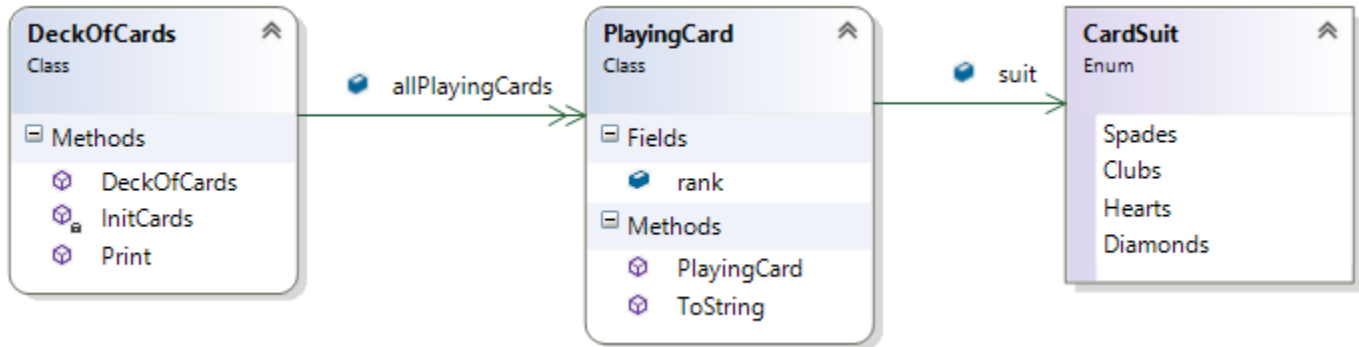
- Add a class Magazine, and make use of *inheritance*.
- Adjust class BookStore so not only books but also magazines can be stored. Adjust method Add so not only a book but also a magazine can be added (*use only one Add method!*).
- In de Start method, create some magazines, and add these to the bookstore. Print the complete stock again, where both books and magazines are displayed. Also print the total price.



```
file:///C:/Users/Gerwin van Dijken/Documents/Vi...  
[Book] 'Dracula' by Bram Stoker, 15.00  
[Book] 'Joe speedboot' by Tommy Wieringa, 12.50  
[Magazine] Time - release day:Friday, 3.90  
[Magazine] Donald Duck - release day:Thursday, 2.50  
[Book] 'The hobbit' by J.R.R. Tolkien, 12.50  
Total sales price: 46.40
```

## Assignment 2 – Deck of cards

In this assignment, we are going to implement (the start of) a card game. The idea is to establish a 'deck of cards' containing 52 cards (4 x 13). A card's suit is either Spades, Clubs, Hearts or Diamonds. The class diagram is provided below:



- Examine this class diagram carefully before implementing the various classes (and fields/methods). All cards must be placed in a list. The `ToString()` method of `PlayingCard` class is an 'override' method.
- Add a method 'Shuffle' to class `DeckOfCards`. Implement a (simple) shuffle-algorithm to shuffle the cards.

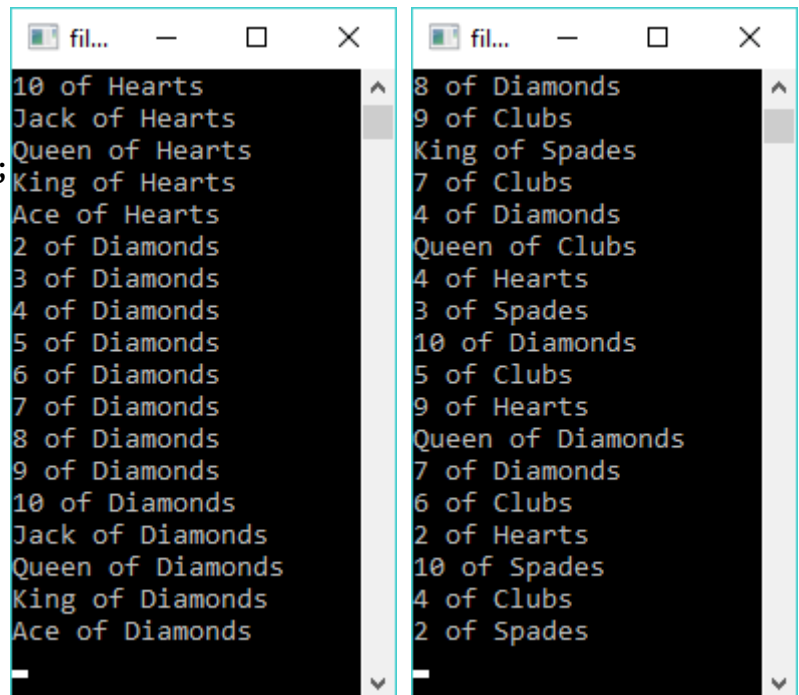
The Main program is given below:

```

void Start(string[] args)
{
    DeckOfCards deck = new DeckOfCards();
    deck.Print();

    deck.Shuffle();
    deck.Print();
}
  
```

The output is shown (partially) to the right:



Kaarten liggen op volgorde

Kaarten zijn geschud

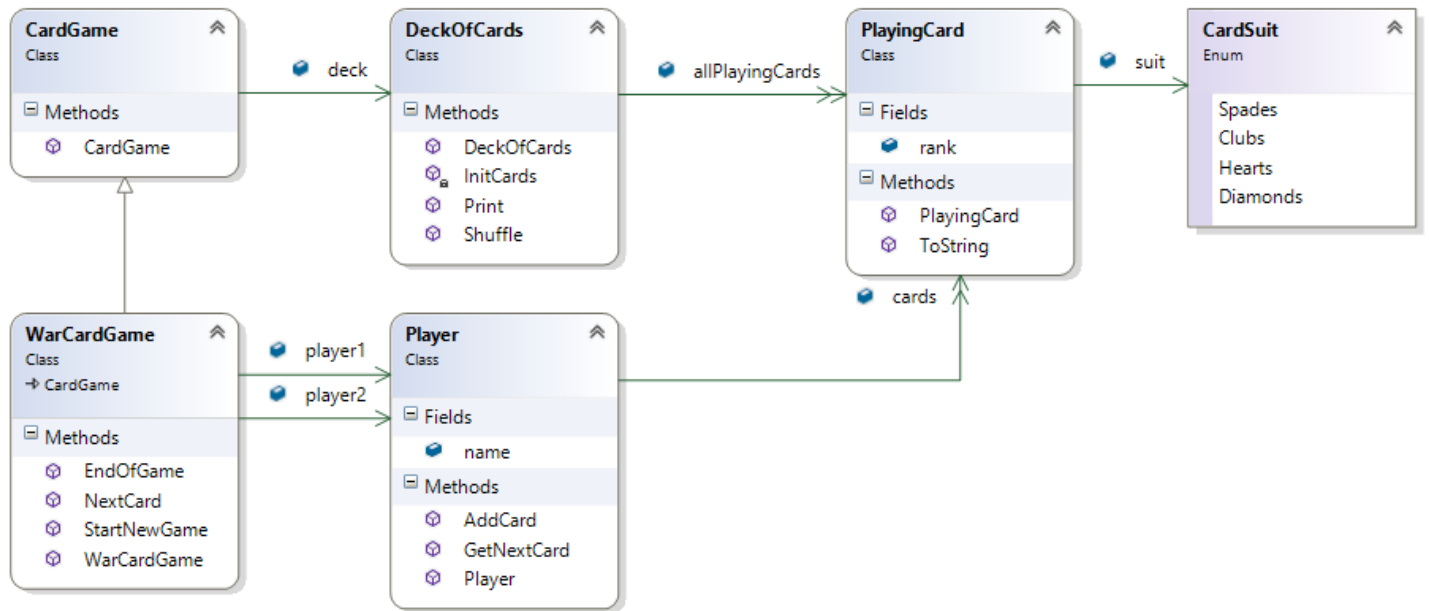
## Assignment 3 – Card game ‘War’

In the previous assignment we have created a deck of cards, and the possibility to shuffle the cards. We can now use this in a card game; the card game we’re going to make is called ‘War’, see:

[https://en.wikipedia.org/wiki/War\\_\(card\\_game\)](https://en.wikipedia.org/wiki/War_(card_game)).

- a) Copy class DeckOfCards and PlayingCard (of the previous assignment) to this 3<sup>rd</sup> assignment.
- b) Create a `class Player` with member ‘name’ (string). Each player has some cards during the game; add a list of cards (‘cards’) to class Player.
- c) Add the following method to class Player: `public void AddCard(PlayingCard card)`. With this method a playing card can be given to the player (*added at the end of the list*).
- d) Add the following method to class Player: `public PlayingCard GetNextCard()`. With this method a playing card can be taken from a player (*return the first playing card from the list; make sure that this card is removed from the player’s list*).
- e) Create a `class CardGame` that has (as only member) a deck of cards.
- f) Create a `class WarCardGame` as derived class of class CardGame. This means that class WarCardGame inherits the deck of cards. Add a player1 and player2 to class WarCardGame; these 2 players are passed via de constructor.
- g) Add the following method to class WarCardGame: `public void StartNewGame()`. In this method the playing cards are shuffled, and all cards are divided evenly among the 2 players (*both players gets a card in turn, until all cards are divided*).
- h) Add the following method to class WarCardGame: `public bool EndOfGame()`. This method returns `true` if a player has no more cards, `false` otherwise.
- i) Add the following method to class WarCardGame: `public void NextCard()`. In this method one card is taken from both players. The player with the highest card (rank) gets both cards; if both cards have the same rank, then these 2 cards are ‘lost’ (*no one gets the cards*).  
*In this NextCard-method, use several WriteLine-statements to indicate which 2 cards are taken, and which player gets the cards (or to indicate that the cards are lost).*

All classes and their relationship can be seen in the classdiagram on the next page. Check if your classdiagram is the same.



With the next main program the card game can be played. The next page shows an example of the output.

```

void Start()
{
    Player player1 = new Player("John");
    Player player2 = new Player("Emma");

    // create game and play it
    WarCardGame war = new WarCardGame(player1, player2);
    PlayTheGame(war);
}

void PlayTheGame(WarCardGame war)
{
    war.StartNewGame();
    while (!war.EndOfGame())
    {
        war.NextCard();
    }

    // TODO: display who has won the game...
}

```

```
file:///C:/Users/Gerwin van Dijken/Document...  -  □  ×
[John] 7 of Spades - [Emma] Jack of Spades
Emma got the cards
[John] 6 of Hearts - [Emma] 4 of Spades
John got the cards
[John] 4 of Hearts - [Emma] 4 of Clubs
2 cards lost...
cards left: [John] 29x, [Emma] 5x
[John] 10 of Clubs - [Emma] 2 of Spades
John got the cards
[John] King of Clubs - [Emma] Queen of Diamonds
John got the cards
[John] Ace of Clubs - [Emma] Jack of Clubs
John got the cards
[John] 3 of Spades - [Emma] Jack of Spades
Emma got the cards
[John] 8 of Diamonds - [Emma] 7 of Spades
John got the cards
[John] 8 of Clubs - [Emma] 3 of Spades
John got the cards
[John] 10 of Hearts - [Emma] Jack of Spades
Emma got the cards
[John] 3 of Diamonds - [Emma] Jack of Spades
Emma got the cards
[John] 10 of Spades - [Emma] 10 of Hearts
2 cards lost...
cards left: [John] 30x, [Emma] 2x
[John] 7 of Clubs - [Emma] Jack of Spades
Emma got the cards
[John] King of Hearts - [Emma] 3 of Diamonds
John got the cards
[John] 2 of Diamonds - [Emma] Jack of Spades
Emma got the cards
[John] 9 of Spades - [Emma] 7 of Clubs
John got the cards
[John] Queen of Spades - [Emma] 2 of Diamonds
John got the cards
[John] King of Spades - [Emma] Jack of Spades
John got the cards
John has won!
```