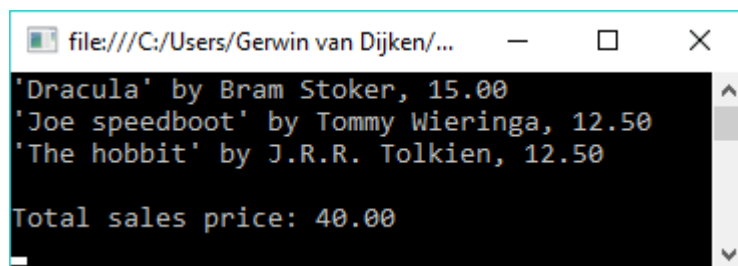


Opdracht 1 - Boekhandel

Een boekhandel verkoopt boeken. Van elk boek wordt de titel, auteur en prijs bijgehouden. De winkelier wil regelmatig een overzicht hebben van alle boeken en ook van de totale verkoopprijs (van de gehele voorraad).

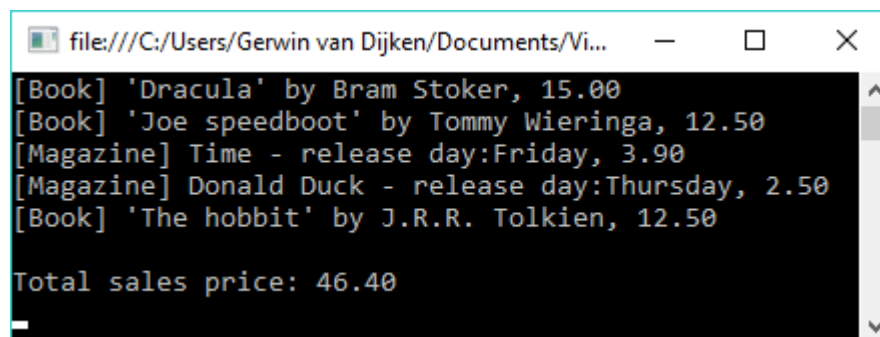
- Maak een class Boek (met velden, een constructor met parameters, en een Print-methode) en een class BoekHandel (met een lijst, en een Voegtoe methode en een PrintVoorraad methode). Boeken kunnen alleen via de Voegtoe-methode worden toegevoegd.
- Maak in de Start-methode een boekhandel aan en voeg een aantal boeken toe (via Voegtoe). Deze boeken mogen hardcoded worden aangemaakt, dus je hoeft geen informatie te vragen aan de gebruiker. Na het toevoegen van de boeken, print de complete voorraad van de boekhandel.



```
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
```

Na enige tijd besluit de winkelier behalve boeken ook tijdschriften te verkopen. Van elk tijdschrift wordt de titel, prijs en dag van uitgifte bijgehouden. Dit komt dus deels overeen met de informatie dat bijgehouden wordt voor boeken.

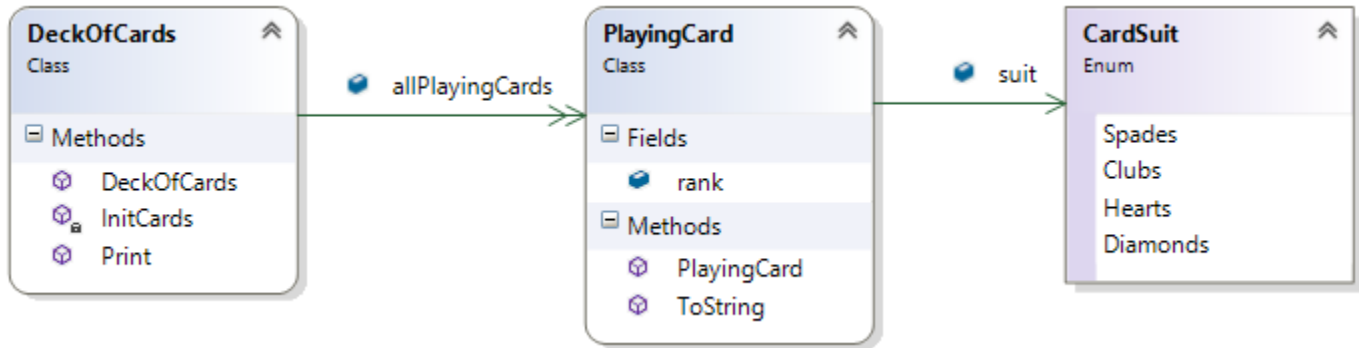
- Voeg een class Tijdschrift toe, zodanig dat er gebruik wordt gemaakt van *overerving* (inheritance).
- Wijzig class Boekhandel zodat niet alleen boeken maar ook tijdschriften kunnen worden opgeslagen. Wijzig methode Voegtoe zodanig aan dat er niet alleen een boek, maar ook een tijdschrift kan worden toegevoegd (let op: *gebruik maar één Voegtoe-methode!*).
- Maak in de Start-methode een aantal tijdschriften aan, en voeg deze toe aan de boekhandel. Print nogmaals de complete voorraad, waarbij zowel boeken als tijdschriften getoond worden. Toon ook de totaalprijs.



```
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
```

Opdracht 2 – Pak kaarten

We gaan in deze opdracht een (begin van een) kaartspel implementeren. De bedoeling is dat er een 'pak kaarten' (deck) beschikbaar komt met 52 speelkaarten (4 x 13). Een kaart is van type Schoppen, Klaveren, Harten of Ruiten (Spades, Clubs, Hearts, Diamonds). Hieronder vind je de classdiagram:



- Kijk goed naar deze classdiagram om de verschillende classes (en velden/methoden) te implementeren. Alle kaarten moeten in een lijst geplaatst worden; gebruik hierbij een geneste loop (4 x 13). De `ToString()`-methode van class `PlayingCard` is een 'override'-methode.
- Voeg een methode 'Shuffle' toe aan het pak speelkaarten (`DeckOfCards`) waarmee alle kaarten geschud kunnen worden. Bedenk zelf een (eenvoudige) manier om de kaarten te schudden.

Het hoofdprogramma ziet er als volgt uit:

```

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

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

De bijbehorende uitvoer zie je (deels) hiernaast:



Kaarten liggen op volgorde

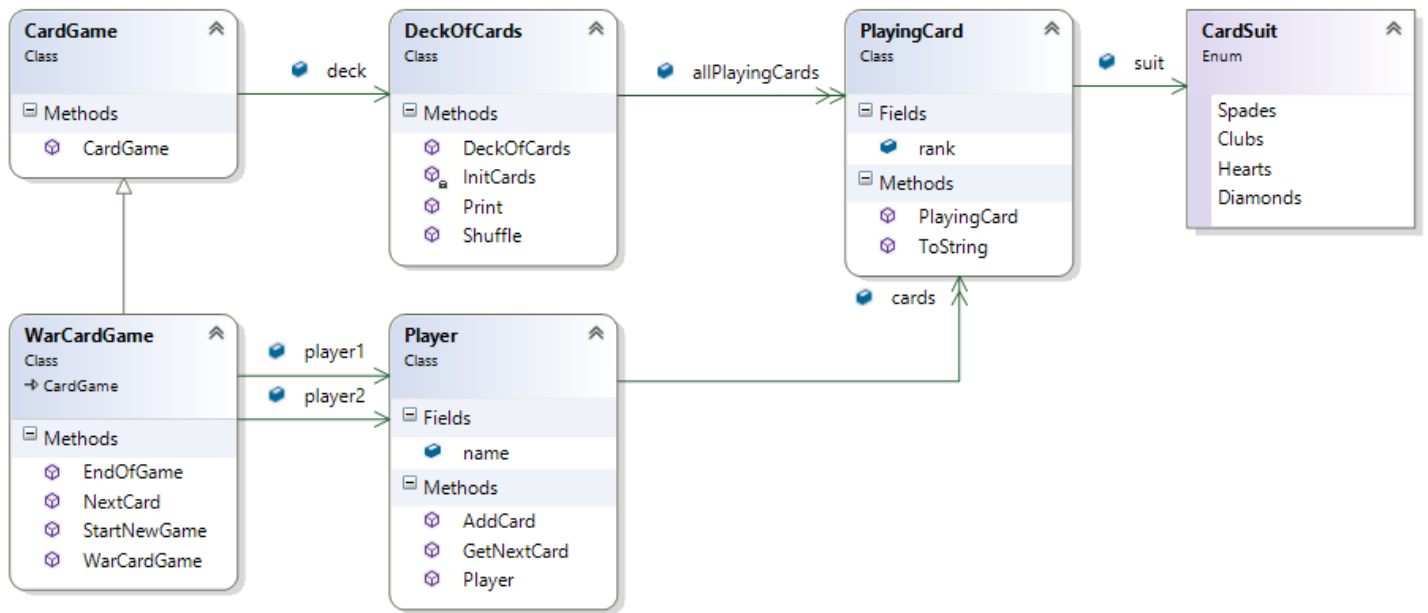
Kaarten zijn geschud

Opdracht 3 – Kaartspel 'Oorlogje'

In de vorige opdracht hebben we een pak speelkaarten gemaakt, waarbij de kaarten geschud kunnen worden. Dit kunnen we nu gaan gebruiken voor een kaartspel; het spel dat we gaan implementeren heet 'Oorlogje', zie: [https://nl.wikipedia.org/wiki/Oorlogje_\(kaartspel\)](https://nl.wikipedia.org/wiki/Oorlogje_(kaartspel)).

- a) Kopieer class DeckOfCards (van de vorige opdracht) naar deze 3^{de} opdracht.
- b) Maak een `class Player` aan met een 'name' (string). Elke speler heeft ook een aantal kaarten gedurende het spel; voeg een lijst met kaarten ('cards') toe aan class Player.
- c) Voeg de volgende methode toe aan class Player: `public void AddCard(PlayingCard card)`. Via deze methode kan een speelkaart aan de speler gegeven worden (*achteraan de lijst*).
- d) Voeg de volgende methode toe aan class Player: `public PlayingCard GetNextCard()`. Met deze methode kan een speelkaart van de speler opgevraagd worden (*geef de eerste speelkaart uit de lijst terug; zorg er voor dat deze kaart uit de lijst v/d speler verwijderd wordt*).
- e) Maak een `class CardGame` dat (als enige) een pak kaarten (DeckOfCards) bevat.
- f) Maak een `class WarCardGame` aan als afgeleide van class CardGame. Dit betekent dat class WarCardGame over een pak kaarten beschikt (*overerving*). Voeg een player1 en player2 toe aan class WarCardGame; deze 2 spelers worden via de constructor meegegeven.
- g) Voeg de volgende methode toe aan class WarCardGame: `public void StartNewGame()`. In deze methode wordt de stapel kaarten geschud, en worden de kaarten over de 2 speler verdeeld (om de beurt krijgen de 2 speler een kaart, totdat alle kaarten verdeeld zijn).
- h) Voeg de volgende methode toe aan class WarCardGame: `public bool EndOfGame()`. Deze methode geeft `true` terug als een speler geen kaarten meer heeft, anders `false`.
- i) Voeg de volgende methode toe aan class WarCardGame: `public void NextCard()`. In deze methode wordt bij beide spelers een kaart genomen. De speler met de hoogste kaart (rank) krijgt beide kaarten; als beide kaarten dezelfde rank hebben, dan zijn de 2 kaarten 'verloren' (niemand krijgt ze).
Gebruik in deze NextCard-methode verschillende WriteLine-statements om aan te geven welke 2 kaarten getrokken zijn, en wie de kaarten krijgt (of dat niemand ze krijgt).

De verschillende classes en hun onderlinge verhoudingen zie je in de classdiagram op de volgende pagina. Controleer of jouw classdiagram hetzelfde is.



Met het onderstaande hoofdprogramma kan het spel gespeeld worden. Op de volgende pagina zie je een voorbeeld van de uitvoer.

```

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!
```