AD – Oefeningen Herhaling OO

Doelstellingen

- Zelf een klasse met properties en methodes kunnen ontwerpen en nieuwe instanties van deze klasse aanmaken en gebruiken
- Een (eenvoudige) overervingsrelatie implementeren
- Het verschil tussen concrete- en abstracte klassen kennen en het juiste type kunnen gebruiken in een specifieke situatie
- Het concept method-overriding kennen en kunnen toepassen
- Het concept polymorfie begrijpen en kunnen toepassen in een concrete situatie

Set-up

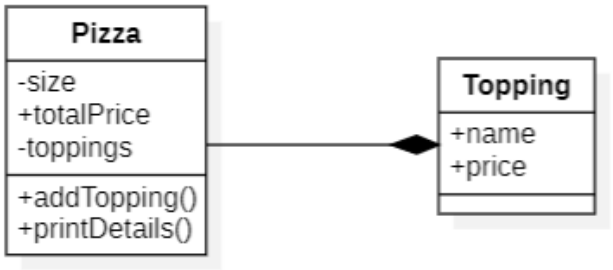
Open Visual Studio en maak een nieuwe solution aan. Voeg per opdracht een nieuw project toe aan deze solution, met als naam de naam van de opdracht.

Classes en objecten

ConsolePizzeria

Maak een eenvoudige console-applicatie voor een pizzeria. Implementeer hiervoor de volgende classes:

- Pizza
  - Size: **enum** met de waarden *Small*, *Medium* en *Large*
  - Totaalprijs: custom **property** die totaalprijs van de pizza berekent (zie later)
  - Toppings: lijst met toppings
- Topping
  - Naam
  - Prijs



Toelichting bij class Pizza:

- AddTopping(): deze **methode** voegt een meegegeven topping toe aan de interne lijst met toppings voor deze pizza
- totalPrice: een custom **property** die de totaalprijs van de pizza berekent en terugstuurt. De prijs wordt als volgt berekend:
  - de basisprijs van een pizza is €11. Maak hiervoor een **constante**.
  - De prijs van de pizza is afhankelijk van de grootte van de pizza:

        - Voor een kleine pizza wordt 20% korting op de basisprijs toegekend
        - Voor een grote pizza wordt 20% supplement aangerekend
        - Voor een medium pizza wordt de basisprijs toegepast
    - Bij deze prijs wordt de som van de prijzen van de toppings opgeteld
- PrintDetails(): deze **methode** schrijft de gegevens van de pizza naar de console. Zorg ervoor dat de uitvoer er als volgt uitziet:

```
Size: Medium
Toppings:
    * Mushrooms
    * Bacon
    * Olives
Total price: 15,15 euro
```

Main-methode:

Test je classes door een Main-methode te maken. Maak hierin een nieuwe pizza aan en voeg een aantal toppings toe. Print vervolgens de details van de pizza naar de console.

ConsoleCards

In deze opgave maak je de klassen voor een eenvoudig kaartspelletje hoger-lager. Maak hiervoor onderstaande klassen:

Klasse Card:

- Een kaart heeft een property *Value* en *Suit*. De *value* stelt de **numerieke** waarde van de kaart voor (2 t.e.m. 10, aas = 1, boer = 11, dame = 12, heer = 13). Voor de *Suit* maak je een **enum** met de waarden Clubs (klaveren), Hearts (harten), Spades (schoppen) en Diamonds (ruiten).
- Een constructor met twee parameters: één voor de *Value* en één voor de *Suit*.
- Een methode IsHigherThan die controleert of de huidige kaart een hogere waarde heeft dan een meegegeven kaart
- Een override van de ToString()-methode, die een string returnt die de kaart duidelijk weergeeft (bv.: 12 Hearts wordt weergegeven als “harten dame”).

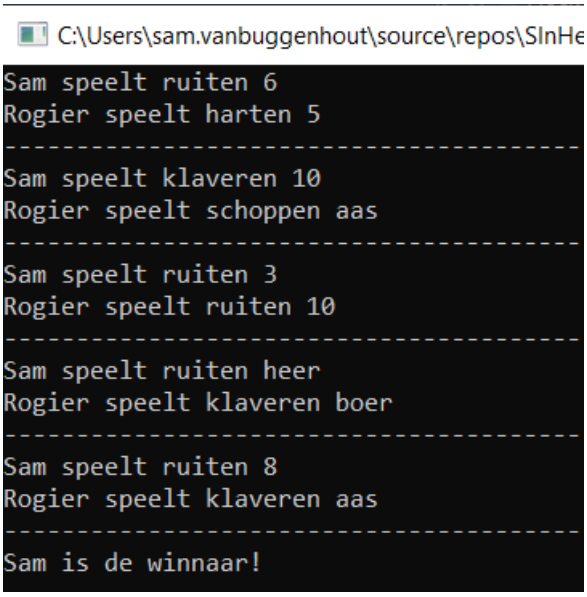
Klasse Deck:

- Bevat een (interne) lijst met kaarten. Deze mogen niet rechtstreeks aangesproken worden buiten de klasse!
- Een constructor die de lijst met kaarten opvult met alle mogelijkheden (**tip:** overloop alle waarden van 1 t.e.m. 13. Maak voor elke waarde van elke Suit een Card-object aan en voeg dit toe aan de lijst met kaarten).
- Een methode Shuffle() die de interne lijst met kaarten schudt in een willekeurige volgorde (opgelet: je schudt de interne lijst, je returnt dus géén nieuwe lijst!). **Tip:** je kunt hiervoor gebruik maken van het Fisher-Yates algoritme)
- Een methode DrawCard die de laatste kaart uit de lijst verwijdert en terugstuurt als return-waarde

Klasse Player:

- Heeft twee properties *Name* en *Score* die publiek toegankelijk zijn en één interne lijst met kaarten die niet van buitenaf kan aangesproken worden
- Voorzie tevens een property *HasCards* die aangeeft of de speler nog kaarten heeft of niet
- Een methode *TakeCard* die een meegegeven kaart toevoegt aan zijn interne lijst met kaarten (m.a.w. de speler krijgt een kaart).
- Een methode *PlayCard* die een **willekeurige** kaart uit de lijst met kaarten van de speler verwijdert en terugstuurt als return-waarde.

Controleer je oplossing met het programma dat meegegeven werd als voorbeeldcode. De uitvoer zou er ongeveer als volgt moeten uitzien:


C:\Users\sam.vanbuggenhout\source\repos\SlnHe
Sam speelt ruiten 6
Rogier speelt harten 5

Sam speelt klaveren 10
Rogier speelt schoppen aas

Sam speelt ruiten 3
Rogier speelt ruiten 10

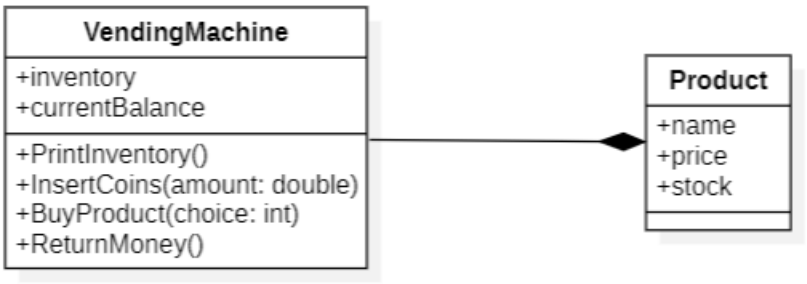
Sam speelt ruiten heer
Rogier speelt klaveren boer

Sam speelt ruiten 8
Rogier speelt klaveren aas

Sam is de winnaar!

ConsoleVendingMachine

Voor deze opdracht simuleer je een eenvoudige drankautomaat. Maak hiervoor twee classes *VendingMachine* en *Product*, zoals hieronder beschreven:



Class Product:

- Name: de naam van het product
- Price: de prijs van het product
- Stock: het aantal producten van dit type in de machine
- Voorzie tevens een **constructor** waarmee je deze drie class members kunt initialiseren

Class VendingMachine:

- Inventory: een **array** met plaats voor 5 producten
- Current Balance: het totale bedrag dat op dit moment in de machine geworpen is
- Voorzie een **default-constructor** (zonder parameters), waarin je de array met producten opvult met 5 producten naar keuze.
- PrintInventory: deze methode schrijft de huidige inventaris met producten naar de console, voorafgegaan door een volgnummer (van 1 t.e.m. 5). Indien het product op de huidige positie niet meer voorradig is, schrijf je de tekst “UITVERKOCHT” naar de console bij dit nummer.

```
DRANKAUTOMAAT
=============
1. Fanta - 1,75 euro
2. Coca Cola - 1,75 euro
3. UITVERKOCHT
4. Water (plat) - 1,50 euro
5. Water (bruis) - 1,50 euro
```

- InsertCoins: met deze methode kan het huidige bedrag in de machine verhoogd worden. De methode controleert tevens of het meegegeven bedrag positief is.
- BuyProduct: deze methode kan gebruikt worden om een product uit de machine te kopen.
    - De methode ontvangt als **parameter** de positie van het gekozen element (nummer van 1 tot 5). De methode gooit een **Exception** indien de invoer ongeldig is.
    - Vervolgens controleert de methode of het gekozen product nog voorradig is. Toon een gepaste foutmelding indien het product reeds uitverkocht is.
    - Nadien controleert de methode of er reeds voldoende geld in de machine werd ingeworpen. Indien dat niet het geval is, toon je opnieuw een gepaste foutmelding.
    - Als het product op voorraad is en er voldoende geld in de machine zit, toon je een boodschap dat het gekozen product afgeleverd wordt. Hierdoor wordt de voorraad (stock) van het product met één verlaagd. Het bedrag in de machine wordt tevens

verminderd met de prijs van het aangekochte product.
Zorg ervoor dat nadien het resterende bedrag wordt weergegeven.

```
Uw Coca Cola komt eraan!
Resterend saldo: 0,25 euro
```

- ReturnMoney: deze methode geeft het huidige bedrag dat op dat moment in de machine zit terug aan de gebruiker (schrijf een boodschap naar de console met het bedrag dat teruggegeven wordt en zet het huidige bedrag op 0).

```
Je krijgt 0,25 euro terug
```

Main-methode:

Schrijf een Main-methode om je classes te testen. Maak een nieuwe instantie van de class VendingMachine aan, welke je gebruikt in de rest van de Main-methode.

Begin met het tonen van de producten in de automaat, gevolgd door een keuzemenu met de volgende opties:

```
DRANKAUTOMAAT
=============
1. Fanta - 1,75 euro
2. Coca Cola - 1,75 euro
3. Fruitsap - 1,60 euro
4. Water (plat) - 1,50 euro
5. Water (bruis) - 1,50 euro

a) geld inwerpen
b) product kiezen
c) geld terug
x) afsluiten
Maak uw keuze:
```

a) Geld inwerpen: via deze keuze kan de gebruiker geld in de machine werpen

```
DRANKAUTOMAAT
=============
1. Fanta - 1,75 euro
2. Coca Cola - 1,75 euro
3. Fruitsap - 1,60 euro
4. Water (plat) - 1,50 euro
5. Water (bruis) - 1,50 euro

a) geld inwerpen
b) product kiezen
c) geld terug
x) afsluiten
Maak uw keuze: a
Bedrag: 2
```

b) Product kiezen: de gebruiker vult het getal in dat bij de keuze hoort

```
DRANKAUTOMAAT
=============
1. Fanta - 1,75 euro
2. Coca Cola - 1,75 euro
3. Fruitsap - 1,60 euro
4. Water (plat) - 1,50 euro
5. Water (bruis) - 1,50 euro

a) geld inwerpen
b) product kiezen
c) geld terug
x) afsluiten
Maak uw keuze: b
Keuze: 2
Uw Coca Cola komt eraan!
Resterend saldo: 0,25 euro
```

c) Geld terug: de gebruiker kan met deze optie het resterende bedrag in de automaat terugvragen

```
DRANKAUTOMAAT
=============
1. Fanta - 1,75 euro
2. Coca Cola - 1,75 euro
3. Fruitsap - 1,60 euro
4. Water (plat) - 1,50 euro
5. Water (bruis) - 1,50 euro

a) geld inwerpen
b) product kiezen
c) geld terug
x) afsluiten
Maak uw keuze: c
Je krijgt 0,25 euro terug
```

Herhaal dit totdat de gebruiker de optie ‘x’ kiest.

Overerving

ConsolePayroll

We maken een eenvoudige applicatie waarmee we de loonlijst van alle werknemers van een bedrijf naar de console kunnen printen.

Een bedrijf heeft zowel werknemers die vast in dienst zijn (en dus een maandsalaris krijgen) als werknemers die per uur betaald worden.

Van elke werknemer houden we de volgende gegevens bij:

- Naam
- Personeelsnummer
- Datum van indiensttreding
- Anciënniteit (= custom property; het aantal jaar dat een werknemer in dienst is, naar beneden afgerond;)

Daarnaast wordt voor elk type specifiek de volgende gegevens bijgehouden:

- Vaste werknemer (FullTimeEmployee):
  - Jaarloon
- Werknemers die per uur betaald worden (HourlyEmployee):
  - Aantal gewerkte uren
  - Loon per uur

Opmerking: alle gegevens dienen bij creatie meegegeven te worden!

Zorg ervoor dat voor elke werknemer het loon kan berekend worden via de methode *CalculateSalary()*. Het loon wordt voor elk type werknemer echter op een andere manier berekend:

- Vaste werknemer:
  - Salaris = jaarloon / 12
- Werknemers die per uur betaald worden:
  - Salaris = uurloon x aantal gewerkte uren

Main-methode:

Test je classes uit door in de main-methode een lijst van werknemers aan te maken. In de lijst moeten zowel vaste werknemers als werknemers die per uur betaald worden zitten.

Print vervolgens een overzicht naar de console dat er als volgt uitziet:

```
EMPLOYEES
=========
101|John Doe|23 years|5000,00 euro
102|Jane Smith|18 years|5416,67 euro
103|Bob Johnson|22 years|5833,33 euro
104|Alice Brown|17 years|6250,00 euro
105|Eve Wilson|21 years|6666,67 euro
106|Michael Smith|20 years|2052,00 euro
107|Sara Johnson|23 years|2604,00 euro
108|David Lee|22 years|1890,00 euro
109|Emily Wilson|21 years|2805,00 euro
110|Daniel Brown|5 years|2385,00 euro
```

Opmerking: zorg ervoor dat de bedragen tot op **twee cijfers na de komma** weergegeven worden.

Uitbreidingen:

- Zorg ervoor dat voor het loon van een vaste werknemer het loon mede bepaald wordt door de anciënniteit.
    - Bij creatie krijgt de werknemer een beginsalaris via de constructor (in plaats van het “volledige jaarsalaris”)
    - Bij het maandloon wordt 1,7% per gewerkt jaar opgeteld
- Probeer de tabel met gegevens op een mooie manier naar de console te schrijven. Je kan hiervoor experimenteren met een library, zoals **Spectre Console**. Neem zeker eens een kijkje in de documentatie in de rubriek “Table”.

Emp. No	Name	Seniority	Salary
101	John Doe	23 years	5000,00 euro
102	Jane Smith	18 years	5416,67 euro
103	Bob Johnson	22 years	5833,33 euro
104	Alice Brown	17 years	6250,00 euro
105	Eve Wilson	21 years	6666,67 euro
106	Michael Smith	20 years	2052,00 euro
107	Sara Johnson	23 years	2604,00 euro
108	David Lee	22 years	1890,00 euro
109	Emily Wilson	21 years	2805,00 euro
110	Daniel Brown	5 years	2385,00 euro

ConsoleMediaPlayer

Voor deze opdracht maak je een eenvoudige applicatie die een mediaplayer simuleert. Er zijn twee soorten mediaplayers: een **music player** en een **video player**.

Beide types mediaplayers delen de volgende data en functionaliteiten:

- Current media: de locatie van het bestand dat door de mediaplayer wordt afgespeeld (bv.: “my song.mp3”)
- Volume: het huidige volume (getal, 0 – 100)
- IncreaseVolume: hiermee wordt het huidige volume met één verhoogd (opgelet: niet hoger dan 100!). Daarnaast wordt het huidige volume opnieuw weergegeven.
- DecreaseVolume: hiermee wordt het huidige volume met één verlaagd (opgelet: niet lager dan 0!). Toon tevens het nieuwe volume in de console.

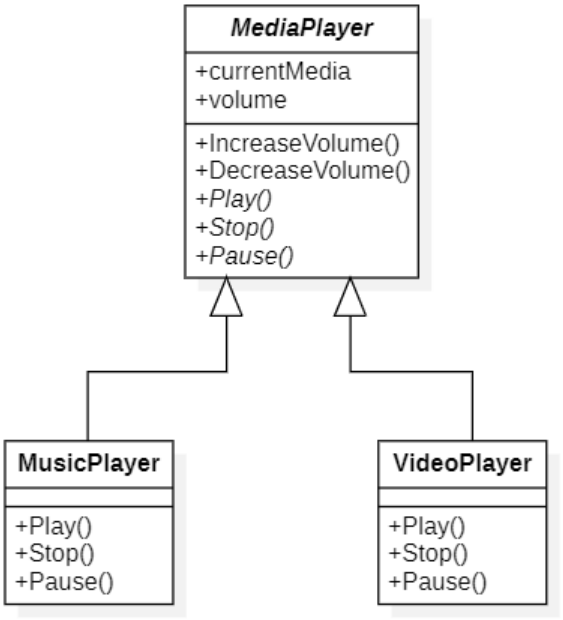
Beide types ondersteunen ook de methodes *Play, Pause* en *Stop*. De logica om een muziekbestand af te spelen verschilt echter van de logica om een videobestand af te spelen. Hetzelfde geldt voor de andere twee methodes. Om de dingen eenvoudig te houden, printen we telkens een andere boodschap naar de console:

```
EXAMPLE MUSIC PLAYER
====================
Playing music: my song.mp3
New volume: 51
Pausing music: my song.mp3
New volume: 50
Stopping music: my song.mp3


EXAMPLE VIDEO PLAYER
====================
Playing video: my video.mp4
New volume: 51
Pausing video: my video.mp4
New volume: 50
Stopping video: my video.mp4
```

Opmerkingen:

- Zorg ervoor dat het niet mogelijk is om een instantie van een generieke MediaPlayer te maken.
- Bij het aanmaken van een nieuwe MusicPlayer/VideoPlayer moet het standaardvolume ingesteld worden op 50.
- Baseer je op onderstaand UML-diagram:



ConsoleBank

In deze opgave maak je een applicatie waarmee je verschillende soorten bankrekeningen simuleert. De applicatie werkt met twee types bankrekeningen: een savings-account (spaarrekening) en een checking-account (zichtrekening). Beide rekeningen hebben gelijkaardige functionaliteiten, maar bevatten ook logica die specifiek is voor het type rekening. Hieronder vind je een overzicht van de data en functionaliteiten van beide types:

Savings-account:

- Huidig saldo op de bankrekening
- Rekeningnummer (in tekst-formaat, bv.: “BE12 34567 89”)
- Maandelijkse interest die de eigenaar ontvangt
- Geld afhalen en storten (**opgelet**: de rekening mag niet negatief gaan! Throw een **Exception** indien het saldo op de rekening ontoereikend is.)
- Huidige maand afsluiten: de interest wordt berekend en aan het huidige saldo toegevoegd

Checking-account:

- Huidig saldo op de bankrekening
- Rekeningnummer (in tekst-formaat, bv.: “BE12 34567 89”)
- Aantal gratis transacties (constante waarde, 100 per periode)
- Prijs per transactie (constante waarde, €0,10 per transactie)
- Aantal uitgevoerde transacties sinds vorige periode
- Geld afhalen en storten (**opgelet**: de rekening mag niet negatief gaan! Throw een **Exception** indien het saldo op de rekening ontoereikend is.). Voor elke afhaling/storting wordt het aantal transacties met één verhoogd.
- Huidige maand afsluiten: voor elke betalende transactie wordt €0,10 aangerekend. Dit bedrag wordt van het saldo van de rekening gehaald.

Tips:

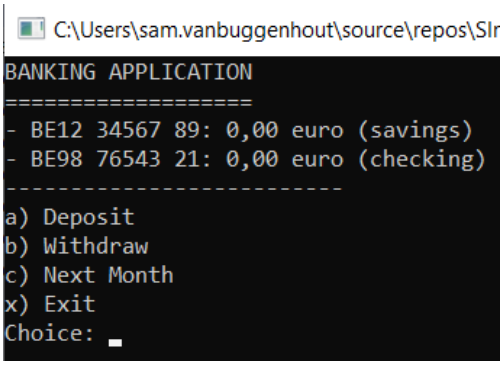
- Maak gebruik van abstracte klassen om duplicate code te vermijden

- Override de ToString()-methode van beide types, zodat de informatie van elke rekening duidelijk wordt weergegeven

Main-programma

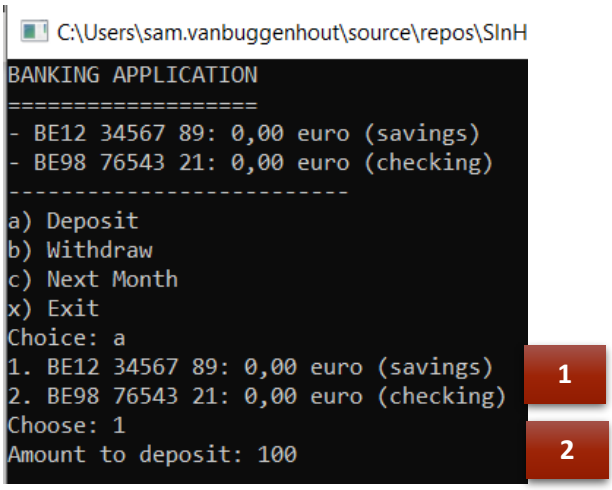
Maak ten slotte een demo-applicatie waarmee je de werking van de verschillende rekeningen illustreert.

Houd een lijst bij met een aantal rekeningen (minstens één van elk type). Print de informatie van de rekeningen naar de console met daaronder een menu met de verschillende functionaliteiten:



Deposit/Withdraw:

Laat de gebruiker eerst de rekening kiezen waar hij/zij het geld van wenst af te halen of geld op wenst te storten (1). Vervolgens vraag je het te storten/af te halen bedrag (2):

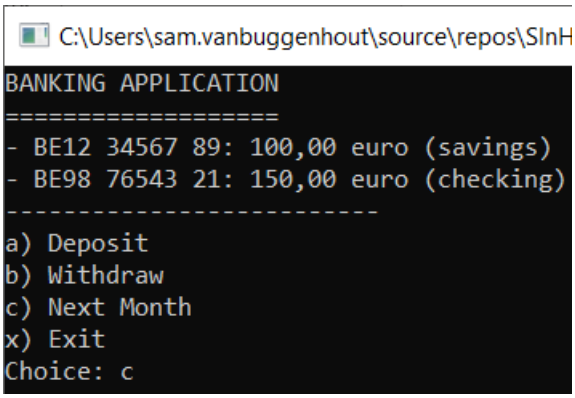


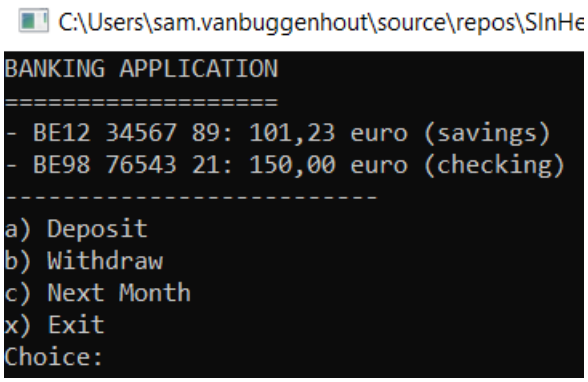
Het scherm wordt nadien opnieuw geüpdatet met de bijgewerkte saldo's.

Next Month:

Wanneer de gebruiker naar de volgende maand wenst te gaan, worden de interesten toegekend op de spaarrekeningen van de gebruiker en worden de kosten van zichtrekeningen verrekend.

Nadien wordt het scherm opnieuw weergegeven met de bijgewerkte saldo's:





De applicatie gaat door totdat de gebruiker de optie *Exit* kiest.