

Oefening: GUI Advanced

In deze oefening doorlopen we een aantal belangrijke aspecten van object oriëntatie.

Deel1

Maak een nieuw C# Winforms project aan. Voeg een tweede project toe van het type Class Library. In de Winforms applicatie voeg je een referentie toe naar de Class Library. Het is de bedoeling om alle logica in de Class Library te plaatsen.

Deel2

Maak een klasse Account met volgende eigenschappen:

- IBAN
- Balance
- Creationdate
- Interest

Overschrijf de toString methode zodat deze alle eigenschappen retournt als string.

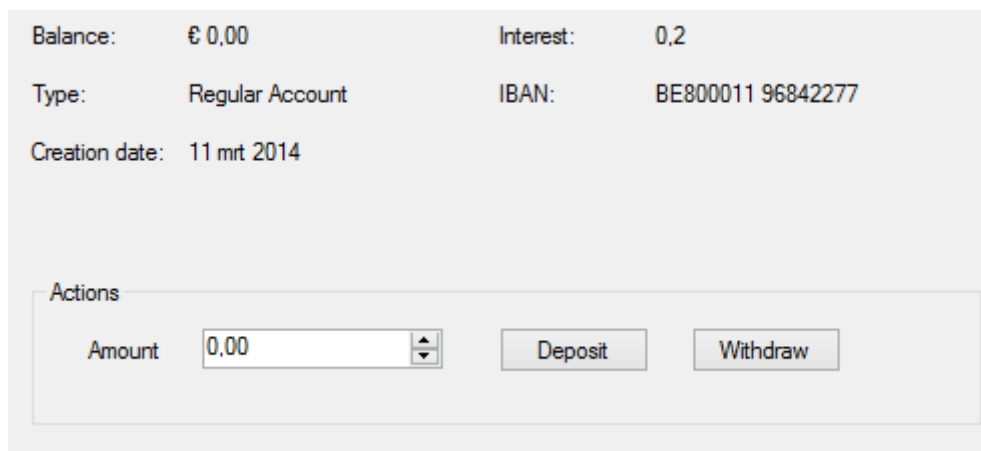
Voorzie:

- een constructor met parameters voor alle eigenschappen van de klasse
- een methode deposit, om geld bij te storten
- een methode withdraw, om geld af te halen

Zorg er voor dat het saldo van de klasse enkel via de methodes storten en afhalen kan aangepast worden.

GUI:

Maak volgend formulier:



Balance:	€ 0,00	Interest:	0,2
Type:	Regular Account	IBAN:	BE800011 96842277
Creation date:	11 mrt 2014		

Actions

Amount:

Bij de start van de applicatie maak je automatisch een nieuwe bankrekening. Voor de invoer van amount kan je een numericUpDown control gebruiken. Voor deze numericUpDown control zet je DecimalPlaces op 2 en maximum op 100000.

Deel3

Een rekening mag maximaal 1000 euro onder nul gaan. Pas de methode afhalen aan zodat dit wordt gecontroleerd. Indien het gevraagde bedrag niet kan afgehaald worden gooi je een Exception.

Controleer dat er een positief bedrag wordt gestort/afgehaald.

GUI:

Toon een messagebox indien de storting niet geldig is.

Deel4: Overerving

Maak een klasse RegularAccount (zichtrekening) en een klasse SavingsAccount (spaarrekening). Beide erven over van de klasse Account. Er zijn enkele verschillen tussen een zichtrekening en een spaarrekening. Zo krijgt men bij een spaarrekening een extra intrest: de getrouwheidspremie. Bij een zichtrekening kan men dan weer een betaalkaart gebruiken.

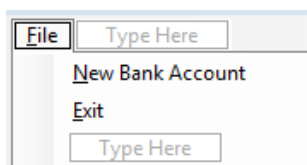
Voeg aan de klasse SavingsAccount een extra property toe: LoyaltyBonus (type Decimal). Aan de klasse RegularAccount voeg je een lijst van strings toe. Elke string in de lijst is een code die verwijst naar een betaalkaart.

Via constructor chaining zorg je er voor dat de juiste parameters van de klasse RegularAccount en SavingsAccount worden doorgegeven aan de constructor van de klasse Account.

Overschrijf ook de toString methode voor RegularAccount en SavingsAccount zodat deze alle eigenschappen toont.

GUI:

Voeg volgend menu toe aan het formulier:



Via File > New Bank Account open je een tweede formulier. Hiervoor kan je de methode showDialog gebruiken.

<div><p>Type</p><p><input checked="" type="radio"/> Normal account</p><p><input type="radio"/> SavingsAccount</p><p>IBAN: <input type="text"/></p><p>Credit Cards</p><div><div>IbCreditCards</div><div><input type="text"/></div><div>Add</div></div><div>Create</div></div>	<ul style="list-style-type: none">• Intresten zijn vast<ul style="list-style-type: none">○ Gewone rekening 0,2%○ Spaarrekening 1,5% en getrouwheidspremie 0,5%• Saldo is bij creatie steeds 0.• Creatiedatum is vandaag.• Indien savings account geselecteerd<ul style="list-style-type: none">○ Disable credit cards groupbox• Bewaar de credit cards een listbox.• Create<ul style="list-style-type: none">○ Sluit venster○ Vul data van het hoofdformulier in
--	---

Deel5: Abstract

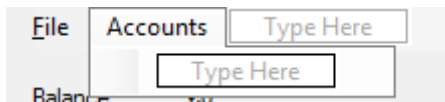
Het is niet wenselijk om een object van de klasse Rekening aan te maken. Maak gebruik van het sleutelwoord abstract om van de klasse rekening een abstracte klasse te maken. Voeg een abstracte property AccountType (type string) toe. Override deze property in SavingsAccount en RegularAccount. De property heeft enkel een get die het accounttype als string zal returnen.

Deel6: Polymorfisme

In het hoofdformulier maak je een list van rekeningen. Noem deze lijst `accounts`. Deze lijst kan zowel spaarrekeningen als zichtrekeningen bevatten. We hebben nu een variabele `account` die de rekening op het scherm toont en een variabele `accounts` die alle rekeningen bevat.

Wijzig de *New Bank Account* logica zodat deze gebruik maakt van de `accounts` list.

Tenslotte willen we ook kunnen wisselen tussen rekeningen. Voeg een nieuw hoofdmenu item toe waar je alle rekeningen toont.



Gebruik de `accounts` variabele om de rekeningen aan het menu toe te voegen.

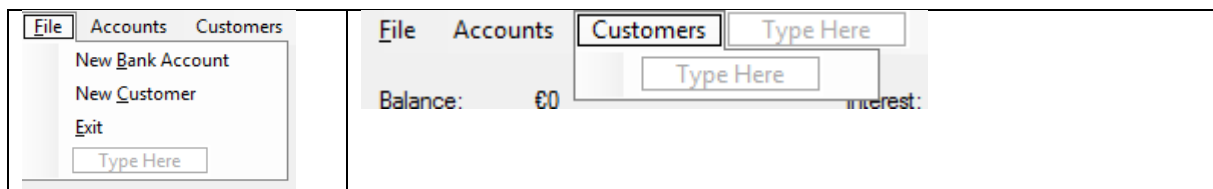
Deel7: Compositie

Maak een klasse `Customer`. Een klant heeft een:

- `FirstName`
- `LastName`

Voorzie een constructor met parameters en overschrijf de `ToString` methode. Voeg een extra property toe aan de klasse `Account`: `Customer`. Via deze property weet je wie de eigenaar is van een rekening. Pas de constructor van `Account`, `SavingsAccount` en `RegularAccount` aan zodat je een klant kan koppelen aan een rekening. Pas ook de `ToString` methode aan.

Pas het hoofdprogramma aan. Voeg een lijst met klanten toe aan het hoofdformulier. Deze lijst kan je beheren via het menu:



- Maak een nieuw formulier om een klant toe te voegen.
- Maak een nieuw formulier om een klant te wijzigen.
- Zorg er ook voor dat de gebruiker een klant kan kiezen bij het aanmaken van een rekening.