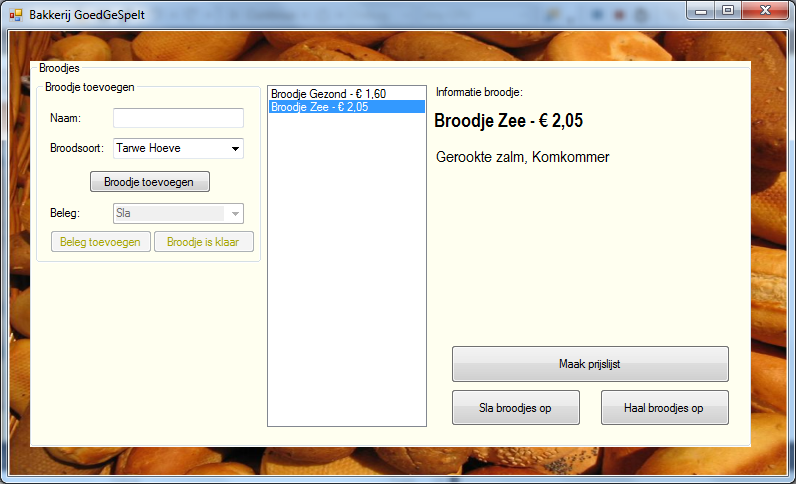
# 0421Maatwerk S – S2 Opdracht Bakkerij GoedGespelt (Files / Streams)

*Inge van Engeland 2014, Achtergrondinformatie Merel Veracx 2016*

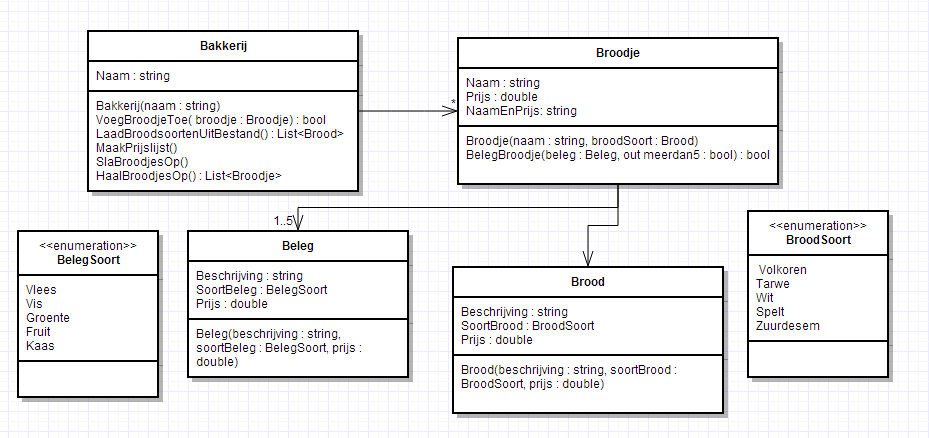
Bakkerij GoedGespelt verkoopt belegde broodjes aan bedrijven. Hiervoor willen ze graag een applicatie waarin ze deze broodjes kunnen samenstellen en prijslijsten kunnen uitdraaien. Deze applicatie ga je in deze opdracht maken.

De interface ziet er als volgt uit:



**Korte uitleg:** Broodjes kunnen worden toegevoegd. Hierbij geef je het broodje een naam en je kiest een broodsoort uit. Als het broodje toegevoegd is kun je het gaan beleggen. Je kiest beleg (tot een maximum van 5, en beleg mag niet dubbel voorkomen) en geeft vervolgens aan dat het broodje klaar is. Het broodje verschijnt in de lijst en als je op het broodje klikt zal rechts gedetailleerde informatie over het broodje getoond worden. Vervolgens kan er een prijslijst gegenereerd worden (als txt bestand) en kun je de broodjes opslaan. Ook is het mogelijk de eerder opgeslagen broodjes weer op te halen.

Op de volgende pagina vind je het klassendiagram voor de applicatie.



In bovenstaand klassendiagram zie je hoe de applicatie geprogrammeerd dient te worden.

**Achtergrondinformatie bij deze opdracht:**

Demo project: <https://portal.fhict.nl/IS/S2/Lesmateriaal%20Maatwerk%20stroom/Demo_Files.zip>

• BinaryFormatter formatter

• formatter.Serialize(bestand,dier)

• ...

Algemene informatie over txt en xml kun je terugvinden in project

<https://git.fhict.nl/se2-opdrachten/referentie_uitwerkingen/tree/master/Streams%20en%20exceptions/converteren>

**Stap 1:** Open het startproject dat bij deze opdracht hoort. Hierin vind je alleen het form-design.Zet bovenstaand klassendiagram om in code. Tip: implementeer de methodes nog niet, maar zorg dat je programmacode wel compileert (door bijv. null / true / false te retourneren bij de methodes)

**Stap 2:** In de class Bakkerij ga je de “VoegBroodjeToe” methode programmeren. Deze methode voegt een nieuw broodje toe aan het assortiment. De naam van het broodje moet echter uniek zijn. Als dat niet het geval is, dan geeft de methode “false” terug en wordt het broodje niet toegevoegd. Anders wordt het broodje wel toegevoegd en geeft de methode “true” terug.

**Stap 3:** Vervolgens gaan we de methode “LaadBroodsoortenUitBestand” programmeren, ook in de Bakkerij class. In het meegeleverde bestand “BroodSoortenLijst.bin” zijn alle broodsoorten van Bakkerij GoedGespelt opgeslagen. De methode leest dit bestand en zorgt dat er een List<Brood> teruggegeven wordt met daarin alle broodsoorten zoals ze in het bin-bestand staan.

**Tip:** Serializable is **niet** slim.

* De velden in klasses Brood en BroodSoort moeten qua naamgeving **exact** overeenkomen.
* Schrijf een hoofdletter verkeerd, bv. beschrijving ipv Beschrijving en dit veld wordt niet ingelezen.
* Kies een ander type, en je krijgt een ArgumentException bij het Deserializeren. Bv. float Bedrag ipv double Bedrag.
* Foreach(Broodje b in list) formatter.Serialize(fs, b); is **niet hetzelfde** als formatter.Serialize(fs, list);

**Stap 4:** De volgende te programmeren methode is “MaakPrijslijst”, nog altijd in de Bakkerij class. Deze methode zorgt ervoor dat alle broodjes uit de List<Broodje> van de Bakkerij class worden opgeslagen in een tekstbestand. Gebruik hierbij de property NaamEnPrijs in de class Broodje die ervoor zorgt dat de naam en de prijs van het broodje netjes achter elkaar gezet worden (met een streepje ertussen). Pas indien nodig de property nog aan om dit mogelijk te maken.

**Stap 5:** De methode “SlaBroodjesOp” slaat alle broodjes op in een binair bestand genaamd “broodjes.bin”. Programmeer deze methode.

**Stap 6:** De methode “HaalBroodjesOp” haalt alle broodjes op uit het bestand “broodjes.bin”, mits dit bestand bestaat natuurlijk. De methode geeft een List<Broodje> terug.

**Stap 7:** Nu gaan we naar de class Broodje. Implementeer de methode “BelegBroodje”. Deze methode geeft een bool terug die aangeeft of het beleg al een keer voorkomt, maar heeft ook nog een out-parameter van het type bool die aangeeft of een broodje meer dan 5x beleg gekregen heeft. Programmeer de methode zo dat er op beide uitzonderingen gecontroleerd wordt. En zorg dat als het beleg toegevoegd mag worden aan het broodje dit ook gebeurt.

De classes zijn nu helemaal geprogrammeerd. Nu de userinterface nog!

**Stap 8:** Zorg dat je globaal een Bakkerij object declareert. Deze wordt in de constructor geinitialiseerd.

**Stap 9:** Maak twee methodes aan:

* VulBroodSoortenComboBox(): Deze methode vult de combobox cbBroodSoorten met alle broodsoorten die in het bin-bestand staan. Hiervoor heb je in de class Bakkerij een methode gemaakt. Gebruik deze.
* VulBelegComboBox(): Deze methode vult de combobox cbBeleg. Gebruik daarvoor onderstaande code:  
   List<Beleg> belegsoorten = new List<Beleg>();

belegsoorten.Add(new Beleg("Sla", BelegSoort.Groente, 0.1));

belegsoorten.Add(new Beleg("Oude kaas", BelegSoort.Kaas, 0.15));

belegsoorten.Add(new Beleg("Jong belegen", BelegSoort.Kaas, 0.1));

belegsoorten.Add(new Beleg("Tomaat", BelegSoort.Groente, 0.05));

belegsoorten.Add(new Beleg("Komkommer", BelegSoort.Groente, 0.1));

belegsoorten.Add(new Beleg("Ham", BelegSoort.Vlees, 0.15));

belegsoorten.Add(new Beleg("Salami", BelegSoort.Vlees, 0.15));

belegsoorten.Add(new Beleg("Rookvlees", BelegSoort.Vlees, 0.2));

belegsoorten.Add(new Beleg("Tonijnsalade", BelegSoort.Vis, 0.3));

belegsoorten.Add(new Beleg("Brie", BelegSoort.Kaas, 0.25));

belegsoorten.Add(new Beleg("Appel", BelegSoort.Fruit, 0.15));

belegsoorten.Add(new Beleg("Gerookte zalm", BelegSoort.Vis, 0.45));

cbBeleg.DataSource = belegsoorten;

cbBeleg.DisplayMember = "Beschrijving";

Vervolgens zorg je ervoor dat deze twee methodes aangeroepen worden in de constructor van het form.

**Stap 10:** Dubbelklik op de knop “Broodje toevoegen”. Als er op deze knop geklikt wordt, wordt er een nieuw broodje aangemaakt met de gegeven naam en het gekozen broodsoort. Als de naam van dat broodje nog niet voorkomt wordt deze toegevoegd en worden de beleg-knoppen op enabled = true gezet. Omdat we met dit broodje verder gaan werken (toevoegen van beleg) is het handig dit broodje even ‘globaal’ te onthouden. Declareer een globale variabele “huidigBroodje” die gevuld wordt met het net aangemaakte broodje.

Als er geen broodsoort gekozen is of als de naam van het broodje al voorkomt dient er een passende melding te verschijnen.

**Stap 11:** Nu programmeren we de knop “Beleg toevoegen”. Deze zorgt ervoor dat het gekozen beleg wordt toegevoegd aan het broodje. Er dient een passende melding te verschijnen als het beleg al aanwezig is of als het broodje al 5x beleg heeft.

**Stap 12:** Programmeer de knop “Broodje is klaar”. Deze knop zorgt ervoor dat de listbox met broodjes opnieuw gevuld wordt en het zojuist gemaakte broodje daaraan toegevoegd wordt. In de listbox toon je de naam en prijs van het broodje (gebruik de property NaamEnPrijs).

**Stap 13:** Als er een broodje uit de lijst aangeklikt wordt verschijnt rechts informatie over dit broodje (zie ook het screenshot op de eerste pagina van deze opdracht). Zorg dat dit netjes getoond wordt, vergeet niet ook het aanwezige beleg van het broodje te tonen).

**Stap 14:** Het programma is nu bijna klaar, enkel de knoppen “Maak prijslijst”, “Sla broodjes op” en “Haal broodjes op” dienen nog geprogrammeerd te worden. Deze methodes zijn in de Bakkerij class al helemaal voorbereid. Zorg dat ook deze knoppen werken.

**Extra:** Probeer het programma nog te verbeteren, bijvoorbeeld:

* Zorg dat je bij de knoppen “Maak prijslijst”, “Sla broodjes op” en “Haal broodjes op” kunt kiezen waar je het bestand opslaat of welk bestand je opent.
* Zorg dat het beleg ook uit een file opgehaald wordt in plaats van dat dit uit te code gehaald wordt.