Opgave 4: Klooster

Ontwerp op basis van onderstaande tekst een aantal klassen die met elkaar samenwerken om een paterorde te modelleren van paters die bidden, nadenken en elkaar inspireren. We helpen je een handje, door dingen die zeker een klasse moeten worden met Hoofdletter te spellen. State variables en methods herken je doordat ze *in cursief* staan. Methods worden bovendien van haakjes() voorzien.

Je kunt steeds de werking testen van de klassen die je reeds hebt, door in Program rechtstreeks objecten van je gemaakte klassen aan te maken met de new operator, door er methods op uit te voeren, en door ze af te beelden via de toString() method.

Later kun je hiermee een Windows-GUI ontwerpen die als game interface fungeert van een simulator van een paterorde waar gedachten circuleren.

Beschrijving van het te modelleren systeem

We proberen te bestuderen hoe *Gedachte*'s ontstaan, veranderen en zich verspreiden in een groep Pater's, door deze als klassen te modelleren. Daarna kunnen we met deze klassen als bouwstenen bijvoorbeeld een game bouwen om een paterorde en de evolutie van hun *Gedachte*'s te simuleren.

Een Pater heeft een *naam*, een *Persoonlijkheid* en een aantal *Gedachte*'s (maximum 20) in zijn hoofd.

Zijn *Persoonlijkheid* is iets dat we met twee getallen van 0 tot 99 voorstellen. Enerzijds *goedheid*: 0 = pure evil, 99 = absolutely good. Anderzijds *creativiteit*: 0 = lawful, 99 = chaotic. Dit mogen gerust ints zijn. Waarden 0 t/m 33 stellen op de ene as "evil" en op de andere as "lawful" voor. 34 t/m 66 stellen op beide assen "neutral" voor. 67 t/m 99 stellen "good" resp. "chaotic" voor. De *toString()* method moet iets als "chaotic good" of "lawful evil" kunnen afbeelden aan de hand van beide opgeslagen Persoonlijkheid-waarden.

Merk op dat evil niet betekent dat de Pater *Gedachte*n van criminele aard heeft. Wel zal hij *Gedachte*n op een meer egoistische manier interpreteren, en de good pater zal eerder een altruistische invalshoek hanteren. Maar de gedachte "gij zult niet doden" zal ook door een evil *Persoonlijkheid* uit eigenbelang geëerbiedigd worden.

Een *Gedachte* is iets dat in het hoofd van een Pater zit. Een *Gedachte* slaat altijd op een *concept*, zoals manier van omgaan met elkaar, wraak, welke daden slecht of goed zijn, ... We stellen een *concept* gewoon voor als een getal, van 1 tot en met 9 (er zijn dus 9 mogelijke concepten). Bijvoorbeeld: concept 1 zou kunnen zijn "hoe sta ik tegenover wraak?", concept 2 kan zijn "wanneer gaat een ouder zijn kind straffen?", ...

Tevens heeft de Pater een bepaalde mening over dat *concept*. De Gedachte in het hoofd van een Pater stelt die mening eveneens voor in de vorm van een *Persoonlijkheid*, net zoals de Pater zelf ook al algemeen een *Persoonlijkheid* (good, evil, lawful, chaotic) heeft. Je kunt de *Persoonlijkheid*-klasse dus herbruiken in *Gedachte*.

Dit betekent dat een *Gedachte* in het hoofd van de Pater in conflict kan zijn met zijn *Persoonlijkheid*. Het zou kunnen dat die erg vergevingsgezind is (lawful good *Persoonlijkheid*), maar wel omtrent het concept "wraak" de mening heeft dat je soms wraak mag nemen als je onrecht aangedaan is (wat eigenlijk eerder chaotic neutral is). Of een chaotische Pater zou zijn kind normaal misschien speelser straffen en een good Pater zou zijn kind misschien ook al eens belonen wanneer het iets goeds doet, maar het kan best zijn dat hij hierover toch een andere mening heeft, doordat de *Gedachte* een *Persoonlijkheid* heeft die verschilt van de algemene *Persoonlijkheid* van de Pater.

Nu we het gehad hebben over de Pater en wat er in zijn hoofd omgaat, kunnen we ons misschien de vraag stellen waar die *Gedachte*'s vandaan komen. Een *Gedachte* ontstaat in het hoofd van de Pater en komt erbij bovenop de reeds aanwezige gedachten door te *bid()* 'en. Wanneer we nu (bv vanuit Program) een Pater die we gemaakt hebben laten *bid*() 'en, dan zal die eerst aan de *Inspiratie* vragen: *inspireerMij()* - dat bepaalt over welk concept het *Gedacht* dat in de Pater opkomt zal gaan. Het alzo ontstane *Gedacht* neemt als *Persoonlijkheid* de *Persoonlijkheid* van de *bid*() 'ende Pater over.

Wat is nu die *Inspiratie*? Dit is iets abstracts; er bestaat er maar 1 van en je Program moet het gewoon aanmaken met behulp van de new constructor, helemaal bij de start. Wanneer je je door *Inspiratie* laat inspireren (*inspireerMij()*), zal die simpelweg elke keer een ander conceptnummer (van 1 tot en met 9) returnen; de eerste oproep 1, de volgende 2 enzovoort, en na 9 terug 1.

Elke Pater heeft *Inspiratie* nodig want anders kan hij niet bidden, want de *Inspiratie* verschaft hem een concept voor de nieuwe *Gedachte* die na het bidden in zijn hoofd bijgekomen is. De constructor van Pater verwacht dus een instantie van *Inspiratie* te ontvangen zodat de Pater die kan bijhouden en gebruiken wanneer nodig.

Nu je terug bezig bent met constructors voor Pater: elke constructor hoort een *naam* te ontvangen zodat we geen Paters kunnen aanmaken die geen naam hebben. Zorg dat er een constructor is die een *Persoonlijkheid*-object ontvangt, maar ook een zonder. In dat geval wordt de *Persoonlijkheid* random toegekend. Maak het jezelf gemakkelijk door dit over te laten aan de constructor van de *Persoonlijkheid*-klasse: de lege constructor maakt gewoon een *Persoonlijkheid* met random values voor *goedheid* (good-evil) en *creativiteit* (lawful-chaotic).

De Pater zal zijn *Gedachte*'s ook willen verspreiden door te *spreek()* 'en. Bij het spreken ontstaat Woord. Het Woord is de verklanking van zijn *Gedachte*'s. Je kunt een *Gedachte* dus *verwoord()* 'en. Welke *Gedachte* gaat hij uitkiezen om uit te spreken? Neem gewoon elke keer hij spreekt de volgende, en begin opnieuw bij de eerste *Gedachte* als je voorbij het huidige aantal *Gedachte*n zit. Indien de Pater geen *Gedachte*n in zijn hoofd heeft, dan wordt een Exception gethrowd: Pater heeft niets te zeggen.

In feite houdt Woord gewoon dezelfde info bij die nodig is om een *Gedacht* op te slaan, maar daarnaast ook een waarde *begeestering.* In feite is dit gewoon nog een *Persoonlijkheid*-object, waar je een kopie van de *Persoonlijkheid* van de Pater opslaat op het moment dat hij sprak. Maw: de *Persoonlijkheid* van de Pater schijnt door in zijn Woord'en, zelfs als die qua opvatting iets totaal anders zeggen (vergelijk het met iets vertellen zonder overtuiging). *verwoord*() verwacht dus die *Persoonlijkheid* te ontvangen als parameter om die als waarde voor *begeestering* te nemen.

Een Pater kan ook *luister() '*en naar Woord. In dat geval wordt - via een gepaste constructor voor *Gedachte* - een *Gedachte* gemaakt die aan het lijstje *Gedachte*'s in zijn hoofd toegevoegd wordt. De *Persoonlijkheid* van de *Gedachte* wordt niet klakkeloos overgenomen. Het gemiddelde wordt genomen van de scores (*goedheid*, *creativiteit*) van de 3: de *Gedachte*, de *begeestering* en de *Persoonlijkheid* van de *luister*() 'ende pater.

Het gevaar bestaat dat, door te veel bidden en luisteren, het hoofd van de Pater op den duur vol zit. Dan zal hij een Exception throwen: hoofd zit vol.

Om dit te vermijden moet hij af en toe nadenken - *denkNa()*. Dan zal hij de *Gedachten* in zijn hoofd op orde zetten. Alle *Gedachten* die op hetzelfde *concept* slaan moeten 1 *Gedacht* worden. Dit kan op 3 manieren. Ofwel wordt de *Gedachte* behouden waarvan de *goedheid*-score meest aansluit bij de *Persoonlijkheid* van de Pater. Ofwel deze waar de *creativiteit* het meest overeen komt. Ofwel krijgt het nieuw *Gedacht* als *Persoonlijkheid* het gemiddelde van alle waardes voor *goedheid* resp. *creativiteit*. Welke van die drie manieren gebruik je? Neem elke keer een andere, niet random maar eerst manier 1, volgende keer manier 2, volgende keer manier 3, dan weer manier 1 enzovoort.

Nadenken heeft nog een ander effect. De *Persoonlijkheid* van de Pater zal evolueren in de richting van de *Gedachte*n die in zijn hoofd zitten. Elke keer dat nagedacht wordt moet, na ordenen van de *Gedachten*, de *Persoonlijkheid* opgeschoven worden met een tiende van het verschil tussen zijn waarde en de gemiddelde waarde van de *Gedachte*n in zijn hoofd.

Hints:

- indien een nieuw Gedacht (door te bidden of luisteren) leidt tot een vol hoofd in een method van de Pater, probeer hem dan eerst te laten nadenken; zit het nog steeds te vol, throw dan pas een Exception
- gemiddelde berekenen van *Persoonlijkheid* van de Pater, *Persoonlijkheid* van het *Gedacht* en *begeestering* (ook een Persoonlijkheid-object): implementeer dit als een method genaamd *combineer()* die 2 parameters van het type Persoonlijkheid verwacht (nl *begeestering* en *Persoonlijkheid* van de oorspronkelijke Gedachte)
- nadenken implementeren: steek eerst de huidige *Gedachte*s in een andere variabele, en maak een nieuwe lijst *Gedachte*s aan die aanvankelijk leeg is. Loop over elk mogelijk *concept*, en per *concept* over alle *Gedachte*s over dat *concept.* Hou per *concept* 1 *Gedacht* over (op 1 van de 3 manieren, de eerste keer manier 1, volgende keer dat hij nadenkt manier 2, dan manier 3, dan terug 1 enz.)
- een private method *voegGedachtToe*() in Pater zal van pas komen
- maak een method *veranderPersoonlijkheid()* die je op het einde van *denkNa()* oproept
- laat elke method - bidden, spreken, luisteren, nadenken - output genereren met Console.WriteLine()
- zorg dat elke klasse mooi als string uitgebeeld kan worden dmv de *toString()* method (behalve *Inspiratie*).
- na genoeg nadenken en *Gedachte*n uitwisselen zal je zien dat de *Persoonlijkheid* van de Paters soms evolueert

Voorbeelden van output: ToString() en methods die zelf ToString() oproepen

Pater ToString(): Pater Jonas, chaotic good, 7 gedachten aan zijn hoofd.

bid(): Jonas bidt en komt tot een gedachte: chaotic good gedachte over concept 4

spreek(): Patrick spreekt over het volgende: chaotic good woorden over concept 2, uitgesproken met een lawful neutral ondertoon.

luister():
Roger luistert naar de volgende woorden: chaotic good woorden over concept 2, uitgesproken met een lawful neutral ondertoon.
Roger denkt er dit van: chaotic good gedachte over concept 2

denkNa():
Roger gaat alles even overdenken: 4 gedachten aan zijn hoofd momenteel.
Roger heeft voor het nadenken deze persoonlijkheid: neutral neutral
Roger heeft na nadenken deze persoonlijkheid: neutral neutral
Roger heeft de gedachten op een rijtje gezet: 2 resterende gedachten.

voegGedachtToe() mislukt door een te vol hoofd:
Roger hoofd zit vol, gedacht is niet opgeslagen. Hij gaat even nadenken.

Indien hoofd na nadenken nog steeds te vol zat:
Hoofd van Pater Roger zit vol en nadenken brengt geen soelaas.