TINPRO02-3

Object georiënteerd programmeren 2

Inleveropdracht: het containerschip

Het containerschip. . .

Iedere in Rotterdam studerende student wordt geacht te weten dat de Rotterdamse haven jarenlang de grootste haven ter wereld was. Een aantal jaren geleden is Rotterdam echter van de troon gestoten. Een korte blik op Wikipedia leert dat in 2020 Rotterdam wereldwijd op de zesde plaats staat, achter Shanghai, Singapore, Kanton, Qingdao, Port Hedland en Tianjin.

De grootte van een haven wordt afgemeten aan het aantal ton goederen dat per jaar wordt overgeslagen (= verwerkt). De haven van Shanghai verwerkt ieder jaar 560 miljoen ton (= 560 miljard kilo!). Rotterdam slaat ieder jaar ruim 469 ton goederen over.

Een gedeelte van de goederen die door de Rotterdamse haven worden overgeslagen, zitten in containers. Iedere dag varen grote containerschepen de haven binnen, waarna de containers m.b.v. kranen worden gelost.

De opgave

In deze opgave ga je het lossen van één containerschip simuleren. Deze opgave is er vooral op gericht je te leren werken met inheritance, polymorfisme en threads. Het gebruik van inheritance en polymorfisme helpt je efficiënter te programmeren, waarbij het makkelijker wordt op een later moment je programma uit te breiden (zoals heel vaak het geval is in projecten).

Men loopt in computerprogramma's tegen problemen aan wanneer meerdere threads gebruik willen maken van dezelfde data. In deze opgave zitten dergelijke problemen verwerkt. Het is aan jou daar een correcte oplossing voor te vinden.

Voorwaarden

De voornaamste zaken, waarop je wordt beoordeeld zijn:

- Correct ontwerp van de classendiagram
- Correct ontwerp van de dynamische structuur van het programma (d.m.v. een sequence diagram)
- Correct gebruik van de concepten van inheritance
- Correct gebruik van polimorfisme
- Correct gebruik van Java Threads
- Correct gebruik van synchronized methodes
- Correct gebruik van notify en wait methodes
- Overzichtelijke programmeerstijl
- Correcte implementatie van het gehanteerde model

Let verder op, op volgende voorwaarden:

- Het is niet toegestaan om door een tool automatisch gegenereerde ontwerpen in te leveren: de ontwerpen moet je zelf gemaakt hebben **voordat** je de code gaat schrijven.
- Het werken met grafische ontwikkelomgevingen zoals Netbeans / IntelliJ is toegestaan, echter: je bent aanspreekbaar op ALLE programmeercode die je inlevert, of deze nu zelf geschreven is of gegenereerd door een tool!
- Met threads wordt op een beschaafde wijze omgegaan, d.w.z.: het gebruiken van deprecated en onveilige methodes zoals stop(), destroy(), enz. is verboden.
- Je bent vrij je programma uit te breiden met extra functionaliteit, gesteld dat de in deze opdracht gevraagde functionaliteit aanwezig is.

Geen g.u.i.

Daar deze opgave van zichzelf moeilijk genoeg is, hoeft een grafische gebruikersinterface niet geprogrammeerd te worden. Multithreading en correcte implementatie van het model staan in deze opgave centraal.

Het model

Het lossen van een containerschip wordt in deze opgave grofweg als volgt gemodelleerd:

- Een containerschip is aangekomen in de Rotterdamse haven en ligt aan de kade.
- Containers worden van het schip afgetakeld met behulp van twee kranen.
- De kranen zetten containers neer op de kade, gesteld dat er genoeg plek is om een container neer te zetten.
- De op de kade geplaatste containers worden opgehaald en weggereden door drie vrachtwagens.
- Het uit het schip takelen en op de kade plaatsen van een container kost tijd. De ene keer duurt dat iets langer dan de andere
- Iedere vrachtwagen zal enige tijd bezig zijn met het wegrijden van een container. Ook dit duurt de ene keer iets langer dan de andere.

Details: Het containerschip

- Het containerschip bevat 100 containers. Dit is, evenals andere getallen die in deze opgave worden gebruikt, overigens niet realistisch.
- Wanneer een container uit het schip getakeld is, maakt het schip daar melding van door een boodschap op het scherm af te drukken.

Details: Containers

- Om bij te kunnen houden welke container door welke kraan en welke vrachtwagen wordt behandeld, heeft iedere container een uniek volgnummer.
- Er zijn 3 types containers (random gedistribueerd op de containerschepen): een standaard type, een gekoelde container en een verwarmde container. De manier waarop de containers van het schip afgehaald worden verschilt per type container:
    - de standaard type hoeft geen bijzondere handelingen bij het uitladen,
    - de verwarmde container moet eerst losgekoppeld uit de verwarmingselementen (dit moet zichtbaar gemaakt worden d.m.v. een melding op het scherm),
    - de gekoelde container moet eerst losgekoppeld uit de koelingselementen (dit moet zichtbaar gemaakt worden d.m.v. een melding op het scherm die anders is dan de melding voor de verwarmde container)

Details: De kranen

- Iedere kraan kan één container tegelijk uit het containerschip takelen.
- Beide kranen werken simultaan: de ene kraan hoeft dus niet te wachten tot de andere klaar is. Beide werken volledig onafhankelijk van elkaar.
- Wanneer een kraan een container uit het schip getakeld heeft, zet deze de container op de kade neer, gesteld dat op de kade genoeg plek is om de container neer te zetten.
- Wanneer het schip leeg is, stoppen de kranen met werken.
- Indien er geen plek op de kade is om de container neer te zetten, zal de kraan moeten wachten tot er een plek vrijkomt.
- Om de ene kraan van de andere te kunnen onderscheiden, heeft iedere kraan een unieke naam.
- Zoals eerder gesteld, zal het uit het schip takelen en op de kade neerzetten van een container tijd kosten. Een kraan doet hier minimaal 1 minuut en maximaal 6 minuten over (het zit soms wel eens tegen met het vastmaken van de kabels). Om ervoor te zorgen dat het nakijken van deze opgave geen 350 minuten duurt, gebruiken we bij het programmeren de regel dat 1 seconde telt voor 1 minuut. Het behandelen van een container programmeer je dus door de randomgenerator van Java een getal te laten genereren dat ligt tussen de 1000 en 6000 milliseconden.
- Een kraan maakt voortdurend duidelijk wat deze aan het doen is:
    - Wanneer de kraan een container uit het schip haalt, maakt deze daar melding van.
    - Wanneer de kraan de container op de kade heeft neergezet, maakt de kraan daar melding van.
    - Etc.

Details: De kade

- Op de kade is plek voor maximaal 5 containers.
- De volgorde, waarin containers op de kade worden geplaatst doet er niet toe. Als er plek is, kan een kraan een container neerzetten.
- De volgorde, waarin containers door de vrachtwagens worden opgehaald doet er niet toe. Als er een container op de kade staat, kan een vrachtwagen deze container wegrijden.
- De kade maakt voortdurend duidelijk hoeveel containers er op de kade staan.

Details: De vrachtwagens

- Er rijden drie vrachtwagens rond over het haventerrein.
- Als er een container op de kade staat, kan deze door een vrachtwagen worden opgehaald en weggereden.
- De vrachtwagen moet kijken van welke type de container op de kade is, en specifieke handelingen verrichten voor speciale type containers:
    - Voor standaard types alleen een melding afdrukken op het scherm dat de container is geladen op de vrachtwagen
    - Voor verwarmde types moet de container gekoppeld worden aan het verwarmingselement en hiervan een melding afdrukken op het scherm dat de container is gekoppeld aan het verwarmingselement
    - Voor gekoelde types moet de container gekoppeld worden aan het koelingselement en hiervan een melding afdrukken op het scherm dat de container is gekoppeld aan het koelingselement (die anders is dan de melding voor de verwarmde container)
- Vrachtwagens werken, evenals de kranen, simultaan. De ene vrachtwagen hoeft dus niet op de andere te wachten.
- Wanneer er op de kade geen containers te vinden zijn, wacht een vrachtwagen tot er een beschikbaar is.
- Een weggereden container verdwijnt uit het model. Je hoeft dus niet te programmeren waar deze container uiteindelijk blijft. De vrachtwagen rijdt met een container weg en komt leeg terug voor de volgende.
- Een vrachtwagen doet er minstens 1 minuut over om een container weg te rijden.
- Iedere vrachtwagen maakt voortdurend duidelijk wat deze aan het doen is:
    - Wanneer de vrachtwagen een container ophaalt, geeft de vrachtwagen daar een melding van.
    - Wanneer de vrachtwagen een container heeft weggereden, geeft de vrachtwagen

daar een melding van.

- Om de ene vrachtwagen van de andere te kunnen onderscheiden, heeft iedere vrachtwagen een unieke naam.

Vragen

Bij de demonstratie en verdediging van je programma moet je de volgende vragen kunnen beantwoorden:

- Aan de hand van de meldingen die schip, kranen, kade en vrachtwagens maken, moet precies te herleiden zijn welke container wanneer en door welke kraan en vrachtwagen werd behandeld. Van iedere container is dus precies na te gaan hoe deze door het ”systeem” loopt.
- Van de vrachtwagens is bekend dat deze minstens 1 minuut nodig hebben om een container weg te rijden. Tracht er middels experimenteren achter te komen hoe lang de vrachtwagens over het wegrijden van een container mogen doen, voordat er geen plek meer is op de kade en de kranen voortdurend op de vrachtwagens moeten wachten.

Aanwijzingen en suggesties

Tijd en threads

In deze opgave zijn er twee partijen gebonden aan tijd:

- De kranen
- De vrachtwagens

Wellicht is het handig de klassen Kraan en Vrachtwagen te programmeren en daarin de threads te verwerken.

Gedeelde data

Wanneer twee kranen tegelijkertijd containers uit een schip halen, zul je voorzorgsmaatregelen moeten nemen om te voorkomen dat één container per ongeluk meerdere keren uit het schip gehaald wordt. Voor de kade ligt dat nog wat ingewikkelder: kranen leveren containers aan de kade, terwijl vrachtwagens containers bij de kade vandaan halen. Ook hier zul je maatregelen moeten nemen om ervoor te zorgen dat containers niet worden overgeslagen of dubbel behandeld worden.

Het is wellicht een idee om de klassen Schip en Kade te programmeren. Beide klassen bevatten een array met Container objecten en beide klassen hebben een aantal gesynchroniseerde methodes, waarmee containers op een veilige manier gehaald en ontvangen kunnen worden.

Een voorbeeld uitvoer

Zie hier een fragment van een mogelijke uitvoer van het programma:

….
Kraan 1 wil container halen...
Schip: container 29 gegeven
Kraan 1: container 29 opgehaald...overladen
Kraan 2 wil container 28 plaatsen...
Kade: container 28 ontvangen op plaats 3
Kraan 2: overladen 28 voltooid...
Kraan 2 wil container halen...
Schip: container 30 gegeven
Kraan 2: container 30 opgehaald...overladen
Wagen A wil container ophalen...
Kade: container 28 gegeven
Wagen A: container 28 opgehaald..wegrijden...
Wagen C wil container ophalen...
Kade: container 27 gegeven
Wagen C: container 27 opgehaald..wegrijden...
Wagen B wil container ophalen...
Kade: container 25 gegeven
Wagen B: container 25 opgehaald..wegrijden...
Kraan 1 wil container 29 plaatsen...
Kade: container 29 ontvangen op plaats 1
Kraan 1: overladen 29 voltooid...
Kraan 1 wil container halen...
Schip: container 31 gegeven
Kraan 1: container 31 opgehaald...overladen
Kraan 2 wil container 30 plaatsen...
Kade: container 30 ontvangen op plaats 2
Kraan 2: overladen 30 voltooid...
Kraan 2 wil container halen...
Schip: container 32 gegeven
Kraan 2: container 32 opgehaald...overladen
Wagen A wil container ophalen...
Kade: container 30 gegeven
Wagen A: container 30 opgehaald..wegrijden...
Kraan 1 wil container 31 plaatsen...
Kade: container 31 ontvangen op plaats 2
Kraan 1: overladen 31 voltooid...
….

Beoordeling

- Correcte implementatie van de opdracht (5.5):
    - Programma levert geen NullPointerExceptions op
    - Correcte implementatie van multithreading
    - Correcte implementatie van inheritance
    - Correcte implementatie van polymorfisme (denk aan abstracte classes)
    - Timing is geimplementeerd
    - Classendiagram en sequence diagram zijn aanwezig en kloppend

- Goede programmeerstijl (0.5)
    - Iedere bestand heeft een comment met auteursnaam en studentnummer
    - Naamgeving is duidelijk en correct
    - Code is consequent (b.v alles in het Nederlands of alles in het Engels, inspringen is hetzelfde, plaatsing van accolades, etc...)
    - Comments gebruiken waar nodig

- Efficientie (1):
    - Geen onnodige herhalingen van code
    - Efficiëntie van multithreading
    - Efficiëntie in termen van timing: er zijn maatregelen genomen om het wachten van de kranen te minimaliseren

Extra's (keuze uit verschillende extra's tot een maximaal van 10 punten):

- Voeg een nieuw type schip toe: een tanker. Deze kan niet uitgeladen worden door een kraan, maar door een pomp. De kraan is in de tussentijd wel beschikbaar voor andere werkzaamheden. Er mogen dus tegelijkertijd zowel een gewoon schip als een tanker uitgeladen worden. De inhoud van de tanker mag alleen op vrachtwagens van type tankwagen (deze moet ook toegevoegd worden) geladen worden. (2 punten)

- De gekoelde en verwarmde containers moeten als eerste geladen worden op de vrachtwagens, en krijgen dus voorrang op de andere containers. Laat de vrachtwagens goed kijken naar het type container om te besluiten welke als eerste weg moeten. (Tip: je zou het simpel kunnen oplossen met een field erbij, of je kunt gebruik maken van "*typeof*", maar deze is nog niet uitgelegd in de les en je zal zelf moeten uitzoeken hoe het werkt). Als er maar een container op de kade staat van het gewone type mag deze wel meteen geladen worden. (1 punt)

- Het wachttijd van uit het schip takelen en wegrijden van de containers is normaal verdeeld, zie https://bit.ly/2HjHy1x voor meer wiskundige informatie over de normale verdeling. Normale verdeling kan in Java geïmplementeerd worden volgens verder beschreven methode, zie https://bit.ly/39zwV6C en https://bit.ly/38tj6Xc voor implementatie. Let op, nextGaussian() levert een getal tussen -1 en 1 op. Je zult zelf moeten bedenken hoe je de minimum en maximum waarde kan bereiken. (1 punt)

LET OP, maximaal 10 punten te behalen!