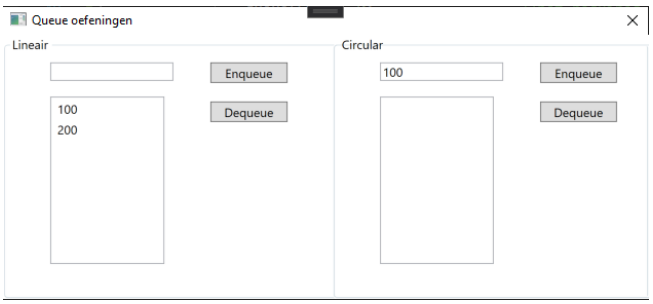
Datastructures, Oefeningen Hoofdstuk 1

Queue

1. Bouw een Lineaire Queue (Versie 1) met behulp van een Array

- Een Queue heeft 2 methodes: **EnQueue en Dequeue**
- Kies een **type** voor je Queue (string, int, double,…)
- Als de queue **volzet** is en er dus geen elementen meer kunnen worden toegevoegd, negeer je alle bijkomende aanvragen.
- Als de queue **leeg** en er wordt een element afgehaald, dan geef je NULL terug of , bv -99999999 bij een int queue.
- Kies bv. **5** voor de **grootte van je array**

Zorg dat je deze kan testen via de WPF applicatie, bv:



2. Bouw ook een circulaire queue (Versie 1) met behulp van een Array.

- Qua uiterlijk (methodes) ziet deze er volledig hetzelfde uit als de lineaire queue
- Enkel intern wordt er nu met 2 variabelen gewerkt: **Front** en **Rear**
- Test deze eveneens met de WPF applicatie (zie screenshot hierboven)

3. Bouw een verbeterde lineaire queue (Versie 2) die ook kan groeien.

- Wanneer de queue vol is en er wordt alsnog een Enqueue aangeroepen
- Maak je een nieuwe array die 2x de grootte heeft van de huidige array
- Zorg ook dat alle aanwezige data wordt **overgekopieerd** naar de nieuwe array.

4. Bouw een verbeterde circulaire queue (Versie 2) die ook kan groeien.

- Gedrag gelijk aan oefening 3
- Merk op dat je bij het kopiëren van de data rekening moet houden met de **Front** en **Rear**
- Nadien zal de **Front** en **Rear** mogelijk ook moeten worden aangepast.

5. Bouw een optelmachine gebruik makende van een queue:

- Voeg een nieuw scherm toe aan de applicatie en voorzie 1 queue voor getallen (int).
- De gebruiker kan via de textbox 1 voor 1 getallen ingeven en via een button telkens toevoegen aan de queue
- Op een bepaald moment willen we de som weten van alle ingevoerde getallen. Voorzie hiervoor een 2^{e} knop ‘Som’. Deze haalt alle getallen van de queue en maakt de som. Toon de som op het scherm.

6. Bouw een automatisch werkend autofabriekje

- We simuleren een mini-autofabriekje. De fabriek krijgt leveringen van banden en motoren. Voor elke auto zijn er uiteraard **4 banden en 1 motor** nodig.
- De **geleverde banden** komen in **Queue 1** wachtende tot er een auto wordt gebouwd.
- De **geleverde motoren** komen in **Queue 2** wachtende tot er een auto wordt gebouwd.
- Een **geproduceerde auto** komt in **Queue 3** tot deze verkocht is.

We hebben voor deze oefening dus 3 queue's nodig. Om te verhinderen dat je 3x dezelfde code moet schrijven kan je ook makkelijker werken met een Jagged -array. Je maakt dan de methodes Enqueue (queueNr, waarde) en Deueue(queueNr). Voor de End variabele maak je dan ook best een array int End[3] zodat je voor elke queue een End waarde kan bijhoden.

- Maak een nieuw WPF window en toon de queues adhv. een Progress bar.

- Bij het opstarten van de applicatie is de stock met auto's uiteraard leeg en ook banden en motoren zijn niet aanwezig.
- De fabriek wordt gestuurd door de verkoop van auto's. Als er **minder dan 10 auto's** in stock zijn wordt er **elke dag 1 nieuwe auto** geproduceerd op **voorwaarde** dat er op dat moment **voldoende banden en motoren** in stock zijn.
- De **levering van 30 nieuwe banden en 10 motoren** gebeurt **1x per week** op voorwaarde dat er op dat moment **niet meer dan 5 auto's** in stock zijn. Anders wordt de nieuwe levering gewoon geweigerd (en moet er een week gewacht worden op de volgende levering).
- De fabriek draait volautomatisch, dus de gebruiker moet niet tussenkomen in het productieproces.
- Wanneer worden er nu auto's **verkocht** ? Dat wisselt. Niet elke dag is er vraag naar een nieuwe auto en onze verkoper is ook al wel eens ziek of met verlof. Gemiddeld is de kans dat **dagelijks een auto verkocht wordt 50%** (gebruik hiervoor dus een random waarde)

Aangezien we in deze applicatie geen knoppen hebben zijn er ook geen events waar we ons kunnen op inschrijven. Hoe kunnen we dan de tijd nabootsen en het productieproces laten werken…? Je kan hiervoor gebruik maken van een DispatchTimer. Deze timer zal om de zoveel tijd (instelbaar) een event genereren. Dit event kan je op inschrijven en dus op die manier kan je dan code laten uitvoeren. Elk keer het event afgaat is er een dag verlopen. (uiteraard gaan we de tijd van de timer wel korter instellen zodat je niet werkelijk een dag moet wachten 😊) Hier vind je terug hoe je vanuit c# code de dispatchtimer moet instellen en inschrijven op het timer event: https://www.wpf-tutorial.com/misc/dispatchertimer/

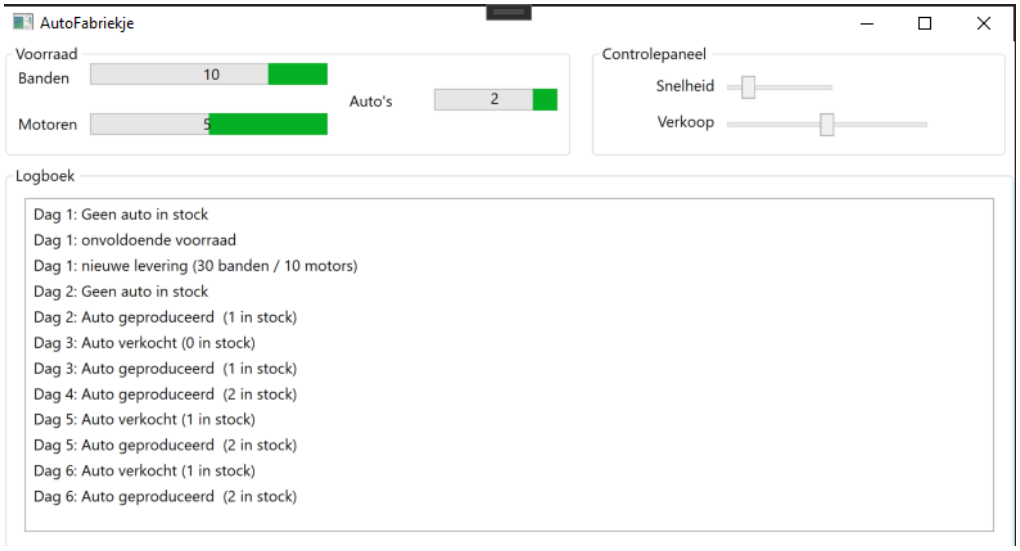
- Stel de timer bv. in op 5 seconden. Elke tick stelt dan 1 dag voor. Dus de code in het tick event gaat ervan uit dat er dan telkens 1 dag verlopen is.
- Check vervolgens in het timer event:
    - Wordt er vandaag al dan niet een auto verkocht ? Is er een auto in voorraad ? Zo ja, haal 1 auto van de queue.

- Zijn er minder dan 10 auto's in stock en is er voldoende voorraad banden en een motor ? Zo ja, produceer 1 auto, door de 4 banden en 1 motor van de queues te halen en 1 auto op de Auto stock queue te plaatsen.
- Zijn er 7 dagen voorbij dan worden er 30 nieuwe banden en 10 motoren geleverd. Tenzij er momenteel meer dan 5 auto's in stock zitten dan wordt er terug een week gewacht.

- Na elke tick moet de nieuwe status van de queues op het scherm worden aangepast. Maak hiervoor een aparte methode die je bij elke tick aanroept.

<u>Uitbreidingen (zie onderaan voor de screenshot van de volledige applicatie):</u>

- Voorzie een **logboek**, waarin alle acties worden gelogd zodat de operator van de fabriek goed kan opvolgen wat er dagelijks gebeurt.
- Voorzie ook een controlepaneel voor de operator zodat hij het proces kan bijsturen indien nodig:
  - **Snelheid**: hiermee kan hij de snelheid van de simualatie instellen. Gaande van 0,1 seconde = 1 dag tot 5 seconden = 1 dag.
  - **Verkoop**: hiermee kan hij de verkoop regelen, van 0% tot 100% kans dan een auto per dag verkocht wordt.



7. Zet je queue om naar een OO versie.

- Maak een klasse **LineairQueueInt** en een klasse **CircularQueueInt**
- Voorzie deze van de nodige methodes (Enqueue, Dequeue, …) en maak hiermee terug een werkende queue.
- Je kan nu je oefeningen aanpassen zodat deze gebruik maken van deze klasse.
- Ook de autofabriek kan je nu in plaats van de jagged array gewoon 3 queue objecten gebruiken