Objectgeoriënteerd Programmeren: WPO 6 (Seriële poort)

1. Inhoud

Exceptions, try-catch, throw, correctheid van de inputs, tryparse, seriële communicatie, asynchrone events, bufferstructuren, formulieren

2. Opgaves

- Demo 1: Seriële communicatie
- Demo 2: Extra formulieren
- A: Rekenmachine
- A: Polygon
- A: Lees data uit Arduino
- A: Choose serialport
- E: Stack
- E: Matrices

2.1 Demo 1: Seriële communicatie

Verschillende devices kunnen met elkaar communiceren via een seriële poort. In dit geval kunnen we een device op regelmatige tijdsintervallen data laten doorsturen naar een C# programma dat wij zelf schrijven. Ons programma moet de data kunnen opvangen en kunnen weergeven. Gebruik hiervoor de Arduino code voor demo 1 (te downloaden via de website). De code stuurt op regelmatige tijdsintervallen het resultaat van het uitlezen van een pin (hoog of laag). De data die de Arduino over de seriële lijn doorstuurt is van de aard: "PIN=1" of "PIN=0". Omdat het uitlezen van de seriële poort een blokkerend effect zou hebben op ons formulier, wordt het lezen in een asynchroon event-handler opgevangen. Hierdoor verliezen we de mogelijkheid om de data vanuit de poort rechtstreeks naar het formulier te "brengen". Dit probleem kunnen we oplossen door de data eerst te bufferen om die vervolgens via een timer op te halen.
Hints: Open de seriële poort met de juiste baudrate (typisch 9600), aantal stopbits (1), aantal databits (8), en de nodige checks (meestal "NONE"). Data naar de poort schrijven kan rechtreeks vanuit een formulier (Write, WriteLine, enz.). Er uit lezen kan via de "DataReceivedHandler".

2.2 Demo 2: Extra formulieren

De programma's die we tot hiertoe hebben geschreven zijn altijd ontworpen door gebruik te maken van 1 enkel formulier. Soms kan het handig zijn als men bepaalde acties zou kunnen groeperen en deze in een apart formulier te plaatsen. Tijdens deze demo zullen we een extra formulier maken die ons toelaat om de seriële poort aan te duiden die we willen gebruiken. Een nieuw formulier aanmaken en tonen kan op 2 manieren.

- De gewone "Show" methode: hierbij wordt het formulier getoond zonder dat het hoofdscherm hierdoor "vastloopt".
- "ShowDialog": hiermee maakt men een dialoogvenster waardoor de rest van het programma op de uitkomst van dit dialoogvenster wacht.

Herneem de vorige demo en pas deze aan zodat de poort gekozen kan worden vanuit een keuzeformulier.

2.3 A: Rekenmachine

Herneem de oplossing van het rekenmachine van een voorgaande opgave. Omdat een rekenmachine enkel op getallen kan werken, kan het dus geen inputs toelaten van een ander type in de textboxen. Zorg ervoor dat de inhoud van de textboxen afgetoetst wordt alvorens de berekeningen uitgevoerd worden. Meld ook aan de gebruiker eventuele foutief ingevulde textboxen. Zorg voor een duidelijke boodschap. Eerst mag je ervan uitgaan dat er enkel gehele getallen ingegeven worden. Nadien kan je dit uitbreiden naar kommagetallen.

2.4 A: Polygon

In een voorgaand WPO heb je een opdracht over polygonen geprogrammeerd. Zorg ervoor dat je geen polygonen kan tekenen die buiten het tekenvlak komen. Neem hierbij de canvas 300 bij 300 pixels. Er wordt een exception afgevuurd wanneer de coördinaten buiten de canvas komen. Vang deze exception op in het main programma, wanneer je deze methode Draw van de polygon opgroept.

2.5 A: Lees data uit Arduino

Deze oefening is vrij gelijkaardig aan de laatste demo. Zorg ervoor dat je de data van de Arduino kan lezen via de seriële poort. De settings voor de seriële poort zijn:

- 19200 baud per seconde
- 8 databits
- 1 stopbit
- geen pariteit of dergelijke checks

De arduino stuurt telkens 2 tuppels van waarden door die gescheiden worden door een komma en een gelijkheidsteken. Analyseer eerst de data door putty te gebruiken (gratis te downloaden). Vervolgens extraheer je de data door gebruik te maken van de stringsplit methode (zoek op hoe deze werkt). Print de laatste waarden in een textbox af (een apart textbox voor elke waarde, dus 2 textboxen). Zorg er ook voor dat de gebruiker de poort kan openen, instellingen kan aanpassen, en kan stoppen ("close") indien nodig.
Opgave: Begin eerst met de data die door het programma "Counters.ino" wordt aangeleverd.
Uitdaging (niet verplicht): Doe hetzelfde met "CounterUitbreiding.ino". Zorg ervoor dat de data bij de juiste teller bijgehouden wordt (kijk eerst na in putty).

2.6 A: Choose serialport

Herneem voorgaande opgave en voeg nu een formulier toe om de seriële poort te kunnen selecteren. Via onderstaande codefragment kan je uit een lijst van beschikbare poorten de juiste poort selecteren. Gebruik deze manier in combinatie met een combobox om het kiezen aanzienlijk te vergemakkelijken.

Codefragment 1: Beschikbare poorten in C#.

```
string[] ports = SerialPort.GetPortNames();

for (int i=0;i<ports.length;i++)
{
   cmbPortSelector.Items.Add(ports[i]);
}
```

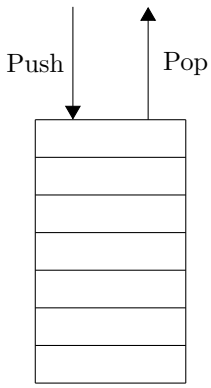
Zorg ervoor dat de parameters van de poort ook in dit formulier gekozen kunnen worden (baudrate, stop bits, pariteit,...).

2.7 E: Stack

In dit programma ga je een buffer emuleren. Buffers worden vaak gebruikt in de microelektronica zodat microcontrollers met elkaar kunnen communiceren. Hierbij wordt het mogelijk om kortstondig data van de fysieke datalijnen te bufferen totdat de bovenliggende applicatie de data ophaalt.

Schrijf een klasse waarin je een array bijhoudt (string). De lengte van de array komt overeen met de lengte die je meegeeft in de constructor van de klasse. De lengte van de array breng je als property naar buiten als "MaxCapacity". Schrijf de methoden Push en Pop. In de methode push voeg je een element toe aan de stack terwijl je via de methode pop een element ervan afhaalt en retourneert. Onderstaande afbeelding geeft weer hoe een dergelijke buffer functioneet.

Zorg ervoor dat je een exception opgooit indien je een element aan de array probeert toe te voegen wanneer deze al vol is. Doe hetzelfde in indien de buffer leeg is en men er toch een element uit probeert te halen. Schrijf een methode of property die toelaat om het huidige aantal elementen in de buffer op te vragen. Dit is het equivelente van Count van een lijst. Vang uiteraard deze exceptions op in het main-programma met de gepaste exception.



Figuur 1: Schematisch overzicht van de stack.

2.8 E: Matrices

Herneem de opgave over matrices en vectoren. In de voorgaande opdracht werd vermeld dat je een null-object retourneert indien de operatie niet mogelijk was vanwege verkeerde matrix of vector dimensies. Een betere manier is echter om gebruik te maken van exceptions. Schrijf de exceptions MatrixSizeDimensionMismatchException en VectorSizeDimensionMismatchException. Zorg ervoor dat deze opgegooid worden indien 2 matrices verkeerde dimensies hebben (MatrixSizeDimensionMismatchException) of 2 vectoren een verkeerde grootte hebben (VectorSizeDimensionMismatchException). Vang deze exceptions ook af wanneer je de operatie in het main-programma probeert op te roepen. Kan je extra informatie geven over de fout tijdens de operatie (bv. grootte van de betrokken vectoren en matrices)?