Objectgeoriënteerd Programmeren: WPO 4

1. Inhoud

Abstracte klassen, override, ToString(), base, private, public, protected, virtual

2. Oefeningen

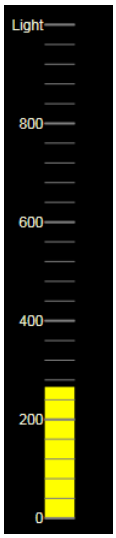
- Demo 1: Oppervlakteberekening
- Demo weerstation: Eigen usercontrol ontwikkelen
- A: Microcontroller communicatie
- A: Airsoft 2.0
- A: Moving objects
- E: Waves
- Weerstation: Thermometer
- Weerstation: Barometer
- Weerstation: Grafieken

2.1 Demo 1: Oppervlakteberekening

In deze demo ga je klassen schrijven om de oppervlakte van een rechthoek, driehoek en een cirkel te bepalen. Elk van deze klassen heeft zijn eigen kenmerken die eigen zijn aan de gerelateerde vorm (hoogte, breedte, straal, enz.). Alle klassen hebben een methode waarmee de oppervlakte berekend kan worden a.d.h.v. deze specifieke eigenschappen (calcSurface). Om deze methode te uniformiseren wordt een abstracte klasse geschreven die door de hierbovenvermelde kindklassen overgeërfd wordt. Deze klasse voorziet de abstracte methode “calcSurface”

2.2 Demo weerstation: Eigen usercontrol ontwikkelen

In deze demo worden de eerste onderdelen van het weerstation vrijgegeven. Download de bijpassende code van de website en maak hierbij een nieuw project aan. In deze demo is het de bedoeling om een usercontrol aan te maken die een waarde weergeeft a.d.h.v. visueel element (zie figuur 1). Hierbij wordt het gebruik van abstracte klassen onderlijnd.



Figuur 1: Voorbeeld verticale gauge.

2.3 A: Microcontroller communicatie

Binnen de elektronica is het niet ongebruikelijk om verschillende microcontrollers op een printed circuit board (PCB) te plaatsen. Vaak moeten deze microcontrollers dan ook samen communiceren om een goede werking van het systeem te garanderen. Een aantal communicatiesystemen zijn:

- UART: universal asynchronous receive/transmit,
- SPI: serial protocol interface
- I^2C: inter IC communicatie

Op microcontrollers is het niet ongebruikelijk om deze communicatiecontrollers al aan boord te hebben. Wel is het zo dat ze via speciale registers (code) ingesteld moeten worden. De exacte werkwijze van deze communicatieprotcollen wordt daarom in deze opgave achterwege gelaten. Om de communicatie in de code te uniformiseren worden een aantal klassen geschreven die de specifieke eigenschappen opnemen, maar tegelijk de gemeenschappelijke eigenschappen (send, initialise en stop) in een superklasse onderbrengen. De functie receive is voor de eenvoud hier niet van toepassing. In deze opgave ga je als volgt te werk:
Schrijf een klasse “communication” waarin je de abstracte methode “send” voorziet. De methode send neemt als argument een string op. De functies “initialise” en “stop” (niet abstract) laten respectievelijk toe om data te versturen of de communicatie stop te zetten. Gebruik hiervoor een booleaanse om dit in de superklasse bij te houden. Erf deze klasse over in de kindklassen (I2C, SPI en UART) en implementeer de methode “send”. De kindklassen hebben elk hun eigenschappen:

- I2C verstuurt data naar een ontvanger die een adres heeft tussen 0 en 256 aan een bepaalde snelheid (100kpbs, 400kbps of 1Mbps). Voor het versturen moeten deze parameters ingegeven zijn via de nodige getters/setters.

- SPI is een point-to-point communicatiesysteem en heeft als enige instelling de transmissiesnelheid, uitgedrukt in kHz. Deze snelheid kan tussen 1kHz en 10MHz liggen.
- UART is eveneens point-to-point, maar hier wordt de transmissiesnelheid beperkt tot 9600, 19200, 38400 of 115200 baud per seconde (verschillend van bits per seconde of bps).

Een klasse kan pas data versturen indien de "initialise"-functie uitgevoerd is. Van zodra "stop"-functie angeroepen wordt kan de klasse geen data meer versturen. De "send"-functie print (messagebox) het volgende af:

- I2C: I2C @<snelheid> naar <adres>:<boodschap>
- SPI: SPI @<frequentie>:<boodschap>
- UART: UART @<baudrate>:<boodschap>

Zorg ervoor dat deze boodschap verschijnt wanneer je de "send"-methode van een klasse aanroept. Je kan hierbij op het formulier de 3 opties voorzien via 3 aparte buttons.

2.4 A: Airsoft 2.0

In de gewone airsoft is het de bedoeling dat je op je tegenstanders schiet met een replica en BB's. Je mikt hierbij op je tegenstander in de hoop hem te raken (en te doen afzien). In airsoft 2.0 krijgen de BB's een upgrade. De basisversie van de BB blijft uiteraard bestaan, maar de nieuwigheden zijn de volgende:

- De gewone BB: volgt een parabolische baan en stopt met bewegen van zodra deze de grond of een persoon raakt. De in te stellen waarden zijn de afschietsnelheid en de hoek t.o.v. het horizontale vlak. Zie ook een voorgaand WPO hiervoor.
- De laserBB: volgt een rechtlijnige baan tot aan het object (of een wand). De in te stellen waarden zijn v_x en v_y. Deze BB's kennen dus geen zwaartekracht!
- De deadmanBB: is een BB die orgineel hetzelfde doet als de gewone BB. Alleen als deze BB een persoon raakt kleurt deze in een andere kleur. De in te stellen waarden zijn de afschietsnelheid, de hoek t.o.v. het horizontale vlak en de eindkleur (indien iemand raken).

Schrijf een klasse BB waarin je de gemeenschappelijke parameters in onderbrengt. Welke zijn de parameters? Schrijf dit op! Schrijf in deze (abstracte) klasse de nodige getters en setters om aan deze parameters aan te kunnen. Het doel van een BB is om doelwitten te raken. Schrijf een functie die de afstand van de BB tot een bepaald doelwit (x,y) kan berekenen. Indien de afstand kleiner is dan 25, is het doelwit geraakt. Erf deze klasse over in de 3 aangehaalde klassen (BB's) van hierboven. Voorzie de nodige extra eigenschappen voor elke klasse. Schrijf nu in de superklasse een abstracte methode "shoot". Deze methode erf je over en implementeer je in de kindklassen. In deze methode komt de telkens de manier van bewegen in. Het bewegen verloopt visueel en a.d.h.v. een timer (bij elke tik zet het object dus een stap). Gebruik een canvas om de BB's weer te geven. Het doelwit geef je in via 2 tekstboxen. Voor de eenvoud mag je je telkens beperken tot het tekenen van 1 enkel object. De methode shoot wordt gebruikt om de BB's te doen bewegen en de positie ervan te updaten. Elke BB start vanuit positie 0,0.

2.5 A: Moving objects

In deze opgave ga je een uitgebreidere versie maken van de biljarttafel. Naast de bewegende schijven programmeer je ook bewegende voorwerpen van eender welk type:

- vierkanten,
- driehoeken en
- rechthoeken.

Hierbij mag je de code van voorgaande oefeningen hernemen en herschikken. Schrijf een abstracte klasse waarin je de positie, het bewegen en het detecteren van het botsen tegen een wand schrijft. Voorzie hierbij de nodige properties en/of methoden. Schrijf ook een abstracte methode DrawObject(Canvas cvs). Erf deze klasse over in de volgende kindklassen:

- schijf,
- vierkant,
- driehoek en
- rechthoek.

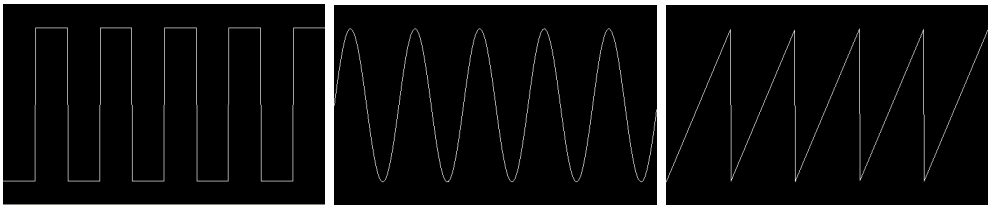
Elk van deze klassen implementeert de methode DrawObject(Canvas cvs) volgens de eigen specifieke eigenschappen van de klasse. Nadien maak je 4 lijsten aan (main programma) waarin je de objecten in gaat opslaan. Je loopt de lijsten af en roept telkens de methode DrawObject aan. De objecten mogen in het begin op een randompositie op het scherm gezet worden (met een randombeweging volgens X en Y). In deze opgave maakt het niet uit of het object al dan niet buiten de canvas komt. Het principe van abstracte klassen overheerst hier.

2.6 E: Waves

In deze opgave ga je een gelijkaardige oefenging maken aan een voorgaande oefening. Een groot deel van de code kan je dus hernemen en anders schikken. In deze opgave is het de bedoeling dat je de waves op een georganiseerde manier gaat programmeren. Hierbij ga je als volgt tewerk. Schrijf eerst een abstracte klasse waarin je de gemeenschappelijke eigenschappen voorziet:

- de periode van de golf,
- de amplitude van de golf, en
- de fase van de golf

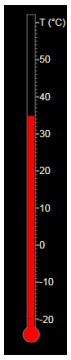
Voorzie hierbij ook de nodige properties en/of methodes. Nadien schrijf je een abstracte methode Draw(Canvas cvs). Erf deze klasse over in de kindklassen sinewave, blockwave en sawtooth. Elk van deze klassen implementeert de abstracte methode Draw. Hierbij maak je gebruik van de superklasse (velden, enz.). Je mag voor dit programma de code van een voorgaande oefening overnemen en aanpassen.



Figuur 2: Voorbeelden van een blokgolf, zaagtand en sinusgolf.

2.7 Weerstation: Thermometer

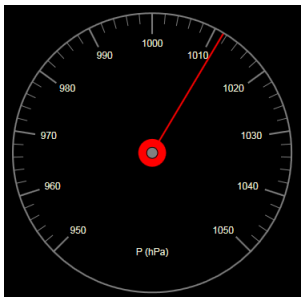
Schrijf een gelijkwaardige usercontrol zoals gedemonstreerd in voorgaande demo. In deze opgave ga je een thermometer maken die de temperatuur van het weerstation weergeeft. In figuur 3 wordt hiervan een voorbeeld gegeven.



Figuur 3: Thermometer.

2.8 Weerstation: Barometer

Schrijf een gelijkwaardige usercontrol zoals gedemonstreerd in de demo. In deze opgave ga je een barometer maken die de luchtdruk gemeten door het weerstation weergeeft. In figuur 4 wordt hiervan een voorbeeld gegeven.



Figuur 4: Barometer.

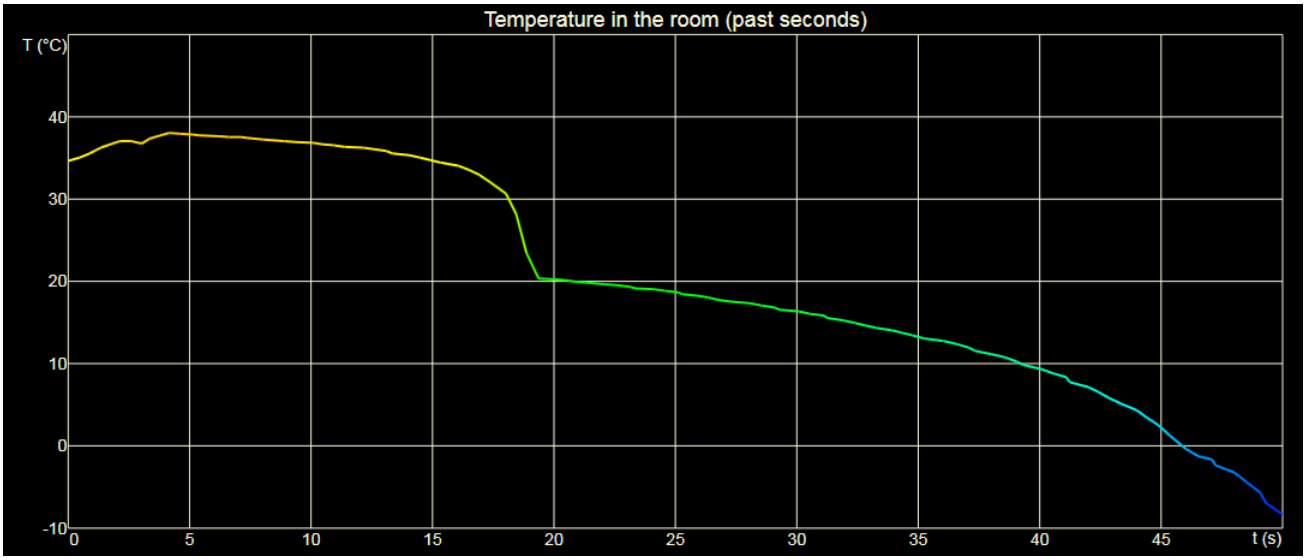
2.9 Weerstation: Grafieken

Naast het gebruik van de gewone usercontrols (barometer, thermometer, enz.) is het ook handig dat men het verloop van de verschillende gemeten grootheden over een bepaalde tijdspanne kan weergeven. Dit kan door gebruik te maken van grafieken. In figuur 5 wordt een grafiek weergegeven.

Om grafieken te kunnen tekenen kan je ook vertrekken van de abstracte klasse zoals hierboven aangeleverd. Echter zal je de abstracte klasse moeten aanpassen (kopieer deze eerst naar een nieuw bestand) zodat je niet 1 punt weergeeft, maar een set van punten (array of lijst). Voorzie in de abstracte klasse de volgende nieuwe methoden en/of eigenschappen:

- het maximaal aantal weer te geven waarden (minstens 1, maximaal n),
- het kunnen toevoegen van een nieuw punt (tijdstip, waarde meting),
- het kunnen aanpassen van de assen ("T(°C)", "P(hPa)", enz.),
- de grafieknaam kunnen aanpassen ("Temperatuur in de klas", "Luchtvochtigheid laatste 24u", enz.) en
- andere eigenschappen die nuttig zouden kunnen lijken.

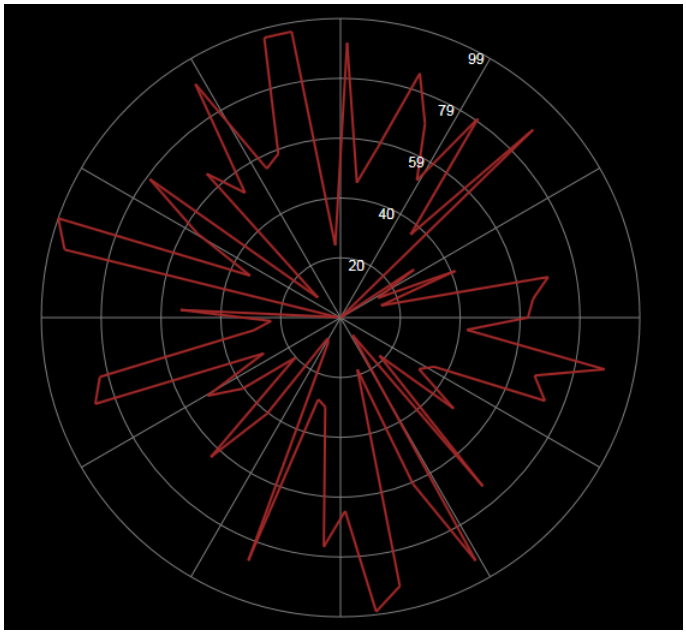
Merk op dat wanneer men een nieuwe waarde toevoegt aan de grafiek, men de oudste waarde dient te verwijderen indien men het maximaal aantal waarden heeft bereikt. Dit kan omzeild worden indien men toelaat dat de grafiek een oneindig waarden bijhoudt. Dit moet dan ook expliciet vermeld worden en als optie beschikbaar zijn.



Figuur 5: Gekleurde grafiek.

Eenmaal de abstracte klasse aangepast is, kan je deze laten overerven in een aantal verschillende grafieken. Een aantal voorbeelden zijn:

- Gewone monochrome grafiek,
- Gradiënt grafiek (zie figuur 5),
- Polaire grafieken (zie figuur 6), enz.



Figuur 6: Polaire grafiek.