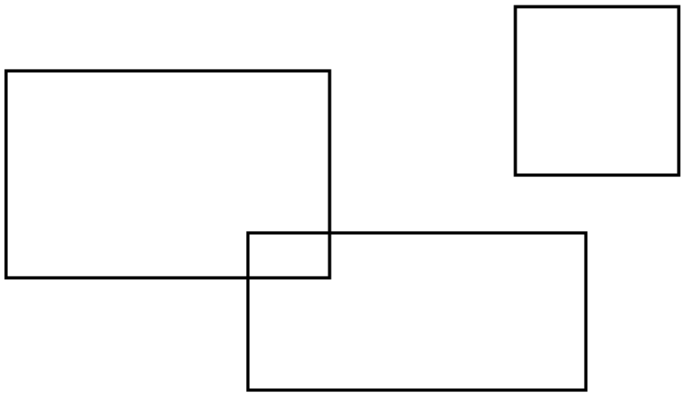
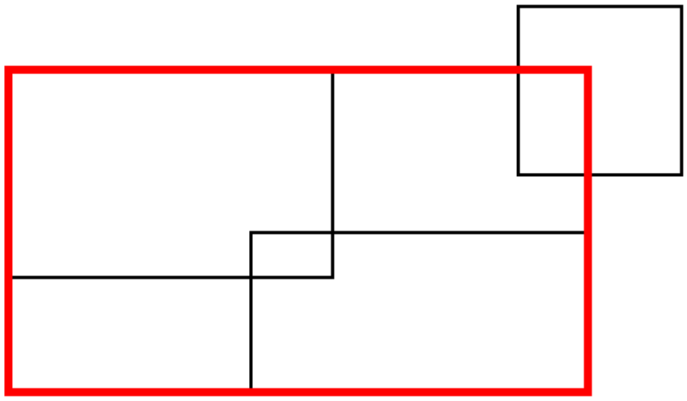
FUNP taak B – Type classes & hogere orde 2019-'20

Met rechthoeken kan je verschillende leuke dingen doen: oppervlakte of omtrek berekenen, zwaartepunt bepalen, … Maar een oefening waar ik om de paar jaar wel iets mee doe, is nog veel leuker, nl. het recursief laten overlappen van rechthoeken. Neem bijvoorbeeld volgende beginsituatie:



De twee rechthoeken linksonder overlappen. Als je dan de overlappende rechthoek maakt, krijg je terug een overlapping, en kan je het proces nog eens herhalen, en nog eens, …



Omdat we een oefening met type classes willen, willen we dit idee generiek maken. Ten eerste kan je "overlappen" anders definiëren afhankelijk van het soort voorwerp:

- voor cirkels kan je de voor de hand liggende definitie nemen van snijden, maar je zou ook een definitie kunnen nemen waarbij ze overlappen wanneer de omsluitende vierkanten overlappen;
- voor verenigingen wanneer er 10 mensen zijn die van beide verenigingen lid zijn
- voor aandelenportefeuilles wanneer ze minstens één aandeel gemeenschappelijk hebben
- voor mensen wanneer ze minder dan 1,5m social distancing in acht nemen.

De actie die je uitvoert wanneer twee voorwerpen overlappen, kan ook verschillend zijn:

- de rechthoeken kan je gewoon samenvoegen tot de overlappende rechthoek
- twee cirkels zou je tot een minimaal overlappende ellips kunnen maken
- voor de verenigingen zou je de namen achter elkaar kunnen plakken en ledenlijsten samenvoegen
- de aandelenportefeuilles zou je in drie kunnen splitsen: de gemeenschappelijke en 2x de aparte
- de mensen zou je een boete kunnen geven

Merk op dat we in de eerste drie voorbeelden één object overhouden, maar in het vierde en vijfde voorbeeld behouden we verschillende objecten, eventueel met een aangepaste status.

Ten derde willen we nog een speciale functie voor een minder strikte definitie van overlappen. Die functie neemt dan een parameter in de vorm van een percentage die aangeeft hoe strikt de overlapping gecontroleerd moet worden. Als de parameter 100% is, is de definitie even strikt als bij het gewone overlappen. In de mate dat het percentage zakt, wordt de definitie minder strikt. Bijvoorbeeld:

- bij de rechthoeken zou je een steeds grotere versie van de rechthoek kunnen nemen
- bij de verenigingen minder en minder gemeenschappelijke leden
- bij de mensen meer en meer afstand

Samengevat willen we voor deze opgave:

- een type class met (minstens) drie functies
- een functie met één parameter **in twee varianten**
- een functie met 4 parameters

De **type class** heeft minstens drie functies:

- Een functie die twee elementen neemt en een boolean terug geeft die zegt of de 2 elementen overlappen
- Een functie die drie parameters neem: twee elementen én bijkomend een percentage.
  Het percentage geeft de strengheid aan waarmee de overlapping moet gebeuren, zoals hierboven beschreven.
- Een functie die twee elementen neemt en een lijst terug geeft met de elementen na overlapping. Dit kan een lijst met één element zijn voor de overlappende rechthoek, maar evengoed een lijst met twee elementen waar de invoerelementen aangepast zijn (bv. boete toegevoegd) of nog meer elementen, bv. de aandelenportefeuilles opsplitsen, …

De **functie met één parameter** heeft als parameter een lijst met voorwerpen die aan de type class voldoen en herhaalt het proces van kijken of twee elementen overlappen en vervangen door de overlappende verzameling tot de lijst niet meer verandert.

- Voor de eerste variant implementeer je de functie met de hand.
- Voor de tweede variant gebruik je onderstaande functie.

De **functie met vier parameters** neemt volgende parameters:

- Een lijst met voorwerpen die voldoen aan de type class
- Een stapgrootte waarmee je het percentage in elke fase vermindert
- Een ondergrens voor het percentage dat gebruikt mag worden
- Een functie die twee van dergelijke lijsten neemt en een boolean terug geeft. Deze functie zegt wanneer je het proces mag stoppen.

Deze functie overloopt een soortgelijk proces als hierboven maar maakt gebruik van de tweede functie van de type class met als parameter een waarde die begint bij 100% en in elke stap vermindert met de opgegeven stapgrootte. De functie stopt ofwel wanneer de parameter onder de ondergrens zakt of wanneer de stop-functie (de vierde parameter) toegepast op de vorige lijst en de huidige lijst een true terug geeft.

Deze opgave valt onder hogere orde omdat de functie met vier parameters een functie als parameter neemt.

Kris Aerts
13 april 2020