**Naam student:**

**Feedback op de oplossing van:**

**Wat vond je van deze oplossing?**

Deel A: STELLINGEN

*Zet achter elke stelling een kruisje in het voor jou passende hokje. Licht je antwoord toe. Feedbackformulieren waarin geen toelichtingen gegeven worden gaan je medestudent niet vooruit helpen en worden dan ook als niet afgegeven beschouwd.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Volledig akkoord | Akkoord | Neutraal | Niet akkoord | Helemaal niet akkoord | Toelichting |
| De design laat toe de implementatie vlot te gebruiken. | x |  |  |  |  | Er worden vanzelfsprekende namen gebruikt, hiermee bedoel ik dat je met het lezen van de naam al diret een beeld krijgt van wat de functies doen |
| De design is modulair, de modules hebben duidelijke contracten | x |  |  |  |  | De contracten zijn opgesteld per implementatie, waardoor het makkelijk wordt om een andere ADT te gebruiken, een functie te veranderen ... |
| In de design wordt gesproken over WAT, niet HOE (geen verwijzingen naar implementatie) | x |  |  |  |  | Er staat enkel wat je moet ingeven en wat je terug kan verwachten, dus niets over de verdere implementatie |
| Duidelijke naamkeuze (van variabelen, constanten, functies, …) | x |  |  |  |  | De namen van de functies spreken voor zich |
| De tabellen maken in het design gebruik van 1 datastructuur (bv Toets, Punt, ...). |  | x |  |  |  | Er staat telkens bij vermeld met welke datastructuur het geimplementeerd moet worden |
| De tabellen werken voor alle datastructuren (bv Toets, Punt, ...). | x |  |  |  |  | Er zijn nergens echt datastructuur-specifieke operaties, aangezien dit eerder een deel is van de implementatie. De datastructuur aanpassen is dus heel eenvoudig in deze OC’s |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Zijn er methodes voorzien om makkelijk van ADT te wisselen? Je wil bv wisselen van 2-3-boom naar een hashmap. |  |  |  |  | x | Deze methodes staan nergens bij vermeld. De operation contracts gaan er van uit dat bij de implementatie de datastructuur vast ligt |
| De gegevens van een ADT zijn enkel via de ADT operaties bereikbaar. |  |  | x |  |  | Gegevens van een klasse worden soms rechstreeks aangeroepen, zoder get- of set-functie |
| Gegevens die niet buiten het ADT moeten/mogen bekend zijn, worden verborgen. |  | x |  |  |  | Door met klasses te werken blijft alles achter de “datamuur” |
| Er is object-georienteerd gewerkt | x |  |  |  |  | Er worden meerdere klassen gebruikt voor 1 implementatie |
| Deze oplossing komt in aanmerking als voorbeeld voor de andere studenten |  |  | x |  |  | Er zijn nog te veel methodes die ontbreken, er komen geen pre- en postconditions in voor. De uitleg erbij is wel goed, maar van een voorbeeld wordt verwacht dat het perfect is. |

Deel B: OPEN VRAGEN

*Beantwoord kort onderstaande open vragen. Probeer in je antwoord zo concreet mogelijk te zijn.*

Flexibiliteit

*Een design is pas flexibel als het goed tegen wijzigingen kan. Stel nu dat we de volgende wijzigingen doen. Geef de klassen die* ***toegevoegd*** *moeten worden (enkel bij naam, geen methodes of attributen) en de klassen die vermoedelijk moeten* ***aangepast*** *worden in dit ontwerp. Baseer je dus enkel op de contracten van dit ontwerp. Als je dat niet kan, duid dan goed aan waar het niet duidelijk is wat er moet gebeuren en/of waar er moet aangepast worden.*

|  |
| --- |
| We implementeren een AVL-boom en willen die in het puntensysteem gebruiken om de toetsen op te slaan. |
| *Hier zijn nog geen functies voor voorzien. Alle bestaande functies zijn nu geimplementeerd aan de hand van een vaste datastructuur. Dit maakt het wijziggen hiervan redelijk lastig.* |

|  |
| --- |
| We willen dat toetsen nu gesorteerd worden op maximum. |
| *Er is geen sorteerfunctie geimplementeerd, dus dit zal redelijk veel tijd en implementatiekennis vragen om aan te passen.* |

|  |
| --- |
| Er zijn nu ook attitudes, die geen maximum hebben en gequoteerd worden met schalen (bv de attitude nauwkeurigheid wordt gequoteerd met uitstekend, goed, net voldoende of onvoldoende). |
| *Makkelijk te implementeren door elke attitude te aanschouwen als een score op 3:*  *0 = onvoldoende*  *1 = net voldoende*  *2 = goed*  *3 = uitstekend* |
| TOPS - volgende onderdelen van de oplossing vond ik bijzonder goed: |
| *Bij iedere functie staat er duidlijk uitgelegd wat deze functies doen. Bovendien bevatte deze oplossing al een paar van de belangrijkste functies.* |

|  |
| --- |
| TIPS - welke wijzigingen (qua structuur, inhoud, vormgeving, klasses die je mist, methodes die je mist, …) die je nog niet vermeldde bij één van de bovenstaande vragen, kan je suggereren? Licht toe. |
| Tegen de eindversie is het aangeraden om er ook pre- en postcondities bij te zetten. De functies en methodes kunnen zeker bij opdracht 2 nog veel uitgebreider zijn. Er staat nu wel een duidelijke beschrijving voor wat welke klasse allemaal inhoud, maar er zijn nog geen basisfuncties toegevoegd, om bv. Punten toe te voegen aan de puntenlijst ... |