**Naam student: Frederic Hamelink**

**Feedback op de oplossing van: Mano Lemmens**

**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 |
| Het design laat toe de implementatie vlot te gebruiken. |  | X |  |  |  | Implementatie is simpel en gemakkelijk te verstaan |
| Het design is modulair, de modules hebben duidelijke contracten. |  |  | X |  |  | Elke klasse heeft dezelfde methode (weinig methodes) |
| In het design wordt gesproken over WAT, niet HOE (geen verwijzingen naar implementatie). | X |  |  |  |  | Goed gedaan |
| Duidelijke naamkeuze (van variabelen, constanten, functies, …) |  |  | X |  |  | Niet veel variabelen en methodes, maar degene die er zijn zijn duidelijk |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Zijn er methodes voorzien om makkelijk van ADT te wisselen? Je wil bv wisselen van 2-3-boom naar een hashmap. |  |  |  |  | X | Geen methodes? idk |
| De gegevens van een ADT zijn enkel via de ADT operaties bereikbaar. | X |  |  |  |  | Correct |
| Gegevens die niet buiten het ADT moeten/mogen bekend zijn, worden verborgen. |  |  |  | X |  | geen prvate methodes |
| Er is object-georienteerd gewerkt | X |  |  |  |  | Klasses |
| Deze oplossing komt in aanmerking als voorbeeld voor de andere studenten |  | X |  |  |  | De constructors zijn zeer duidelijk |

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 reservatiesysteem gebruiken om de vertoningen op te slaan. |
| *Het is niet duidelijk waar de boom gebruikt wordt.* |

|  |
| --- |
| We willen dat vertoningen nu gesorteerd worden op vrije plaatsen. |
| *ID verandert naar vrije plaatsen* |

|  |
| --- |
| Er zijn nu ook ochtendvoorstellingen, m.a.w. er komt een slot bij om 11u. |
| *Slot in de instructor toevoegen van vertoning* |
| TOPS - volgende onderdelen van de oplossing vond ik bijzonder goed: |
| *De initializer methods* |

|  |
| --- |
| 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. |
| Meer methodes (usecases) toevoegen, te gelimiteerd; |