Objectgerichte Softwareontwikkeling 2021-2022
Professoren: S. Desmet en K. Gerling
Labo-assistenten: M. Aufheimer, L. Bruynseels, S. Langendries, K. Pelsmaekers, Z. Sun, R. Vanhoof, N. Vinckier, G. Vranken en J. Wauters.

Lab Sessie Week 12
Proefexamen

Deze sessie bestaat uit een opdracht **Java-implementatie** en drie extra oefeningen over het ontwerpen van **UML-diagrammen**. Het eindexamen zal bestaan uit een implementatieopdracht gelijkaardig aan deze (70% van de score) en **één** modelleeroefening (30% van de score).

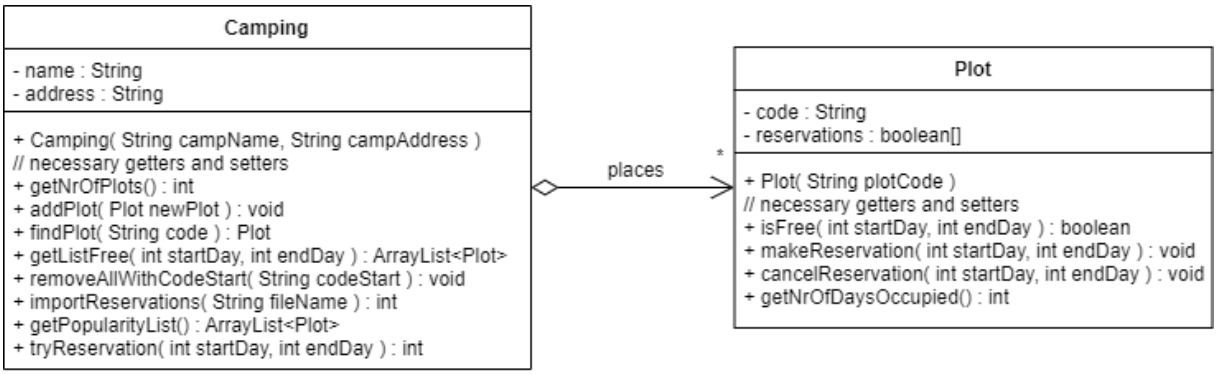
Objectgerichte Softwareontwikkeling 2021-2022
Professoren: S. Desmet en K. Gerling
Labo-assistenten: M. Aufheimer, L. Bruynseels, S. Langendries, K. Pelsmaekers, Z. Sun, R. Vanhoof, N. Vinckier, G. Vranken en J. Wauters.

Oefenexamen OOS: programmeergedeelte

Dit deel van het oefenexamen behandelt Java-implementatie. Het is de bedoeling dat het binnen 3 uur wordt opgelost. Je mag een samenvatting van 4 pagina's en de Java-documentatie gebruiken. Het programmeergedeelte telt mee voor 70% van je totaalscore op het examen.

Probleembeschrijving

Een camping wil het beheer van zijn reserveringen automatiseren. De camping stelt een aantal percelen (in het Engels: plots) ter beschikking. Voor elk perceel wordt een code bijgehouden en een array van booleans met dagen waarop dit perceel gereserveerd is (true indien nog vrij, false indien gereserveerd). De camping is open van 1 mei (index 0 in de array) tot 30 september (index 152 in de array). In deze oefening zullen data dus steeds voorgesteld worden als integers tussen 0 en 152.



Test 1 [0.5 punt] Test de constructor van *Plot* en enkele getters.

Test 2 [0.5 punt] Test of alle dagen gemarkeerd zijn als vrij na aanmaak van een nieuw perceel.

Test 3 [1 punt] Test de methode *isFree*, die *true* returnt als het huidige perceel nog vrij is in de periode tussen *startDay* (inbegrepen) en *endDay* (niet inbegrepen), en anders *false*. De methode geeft ook *false* terug als de start- en/of einddag niet tussen de juiste grenzen liggen. Daarnaast testen we hier ook de methode *getNrOfDaysOccupied*, die telt hoeveel dagen dit perceel reeds gereserveerd is.

Test 4 [1 punt] Test of we een reservering kunnen maken en opzeggen. Uiteraard is het niet mogelijk om reserveringen te maken als (een deel van) de periode die we willen reserveren niet vrij is. Als we een reservatie proberen maken op een dag die niet vrij is, print je een foutboodschap (je mag zelf kiezen welke).

Test 5 [1 punt] Test de constructor van *Camping* en enkele getters.

Test 6 [2 punten] Test of we nieuwe percelen kunnen toevoegen aan de camping. Opgelet, percelen moeten een unieke code hebben. Als we dus een perceel proberen toevoegen met een code die al bestaat, gebeurt er niets.

Test 7 [2 punten] Test of we een perceel kunnen opzoeken aan de hand van zijn code. Deze methode returnt het gevonden perceel, of null als er geen perceel met de gegeven code bestaat.

Test 8 [2 punten] Test of we een lijst kunnen genereren van alle percelen die beschikbaar zijn tussen een bepaalde start- en einddatum.

Test 9 [2 punten] Test of we alle percelen kunnen verwijderen waarvan de code begint met een gegeven string.

Test 10 [2 punten] Voor jouw planningssoftware bestond werden de reservaties van de camping bijgehouden in tekstbestanden. Hier testen we de methode *importReservations*, die een aantal reservaties importeert uit zo een tekstbestand. De tekstbestanden hebben de volgende formaat:

- Code van perceel
- Startdag reservatie
- Einddag reservatie

Deze structuur herhaalt zich een aantal keren. De methode returnt het aantal geïmporteerde reserveringen. Als de methode ongeldige informatie vindt (bv. een code van een niet bestaand perceel), returnt deze het aantal tot dan toe uitgevoerde imports als een negatief getal en stopt verder met importeren.

Test 11 [2 punten] De camping wil vanaf nu een onderscheid maken tussen twee verschillende soorten percelen: Staanplaatsen (“pitch” in het Engels, dit zijn lege plekken waar bezoekers hun eigen tent of caravan kunnen plaatsen) en volledig ingerichte tenten. Voor staanplaatsen wordt er opgeslagen of er al dan niet elektriciteit aanwezig is. Voor tenten wordt het bouwjaar opgeslagen, en een *ArrayList* van strings met mogelijke extra’s die aan de tent kunnen worden toegevoegd (babybedje, microgolfoven, …).

Implementeer deze functionaliteit door gebruik te maken van inheritance. Check de testcode om te zien welke klassen en methoden er verwacht worden.

Test 12 [2 punten] Test of we een lijst kunnen maken van alle percelen gerangschikt naar oplopende populariteit (d.w.z., het perceel met de minste reserveringen zit vooraan in de lijst, daarna dat met de op één na minste, enzovoort). Mochten er percelen voorkomen met een gelijk aantal reserveringen maakt het niet uit in welke volgorde je die zet.

Test 13 [2 punten] Als laatste uitdaging maak je een methode *tryReservation*, die een reservatie probeert te spreiden over meerdere percelen als er geen perceel beschikbaar is voor de volledige periode van de reservatie. De methode werkt als volgt: eerst zoek je het perceel dat de langste vrije periode heeft vanaf de startdag van de reservatie, en reserveer je de beschikbare periode. Daarna doe je hetzelfde vanaf de eerste dag die nog niet gereserveerd kon worden, enzoverder tot de volledige periode is gereserveerd. De methode returnt hoeveel percelen nodig waren voor je reservatie. Denk eraan dat je extra hulpmethoden mag definiëren als je dit nuttig vindt.

Objectgerichte Softwareontwikkeling 2021-2022
Professoren: S. Desmet en K. Gerling
Labo-assistenten: M. Aufheimer, L. Bruynseels, S. Langendries, K. Pelsmaekers, Z. Sun, R. Vanhoof, N. Vinckier, G. Vranken en J. Wauters.

Oefenexamen OOS: UML-oefeningen

De volgende oefeningen dienen als extra oefening bij het vertalen van een probleembeschrijving naar een klassendiagram. Je krijgt één vergelijkbare taak op het examen voor 30% van je totale cijfer. Voorbeeldoplossingen voor deze oefeningen zullen op Toledo worden gepubliceerd.

Album

Maak een gedetailleerd klassendiagram op basis van de volgende probleembeschrijving: definieer de benodigde klassen, hun attributen, de signatuur van hun methoden en de relaties tussen de verschillende klassen.

Aangezien jij de programmeerexpert van de familie bent, vragen ze je om een applicatie te maken die in staat is om interactieve albums te maken van hun vakantiebeelden (momenteel alleen foto's en films) met enkele extra functies.

Alle beelden hebben een naam, een grootte (een geheel aantal bytes) en een datum waarop ze zijn gemaakt in het formaat “dd/mm/jjjj”. Er is een mogelijkheid om een beoordeling toe te voegen (een geheel getal hoger dan nul en maximaal 10) en om maximaal 3 tags toe te voegen aan elke visual. Het moet mogelijk zijn om te controleren of een visual een bepaalde tag bevat. Voor films moet je ook hun afspeeltijd in seconden opslaan. Films kunnen worden afgespeeld en gepauzeerd, foto's kunnen alleen worden getoond, maar het moet mogelijk zijn om een foto te tonen met of zonder fade-in effect.

Een album heeft een naam en je kunt er een onbeperkt aantal foto's en filmpjes aan toevoegen. Zorg ervoor dat een album niet 2 beelden met dezelfde naam en dezelfde datum kan bevatten. Maak de optie om te zoeken naar de hoogst beschikbare beoordeling in het album. Je zou ook alle visuals met een bepaalde tag moeten kunnen selecteren, door een verzameling te genereren van alle visuals die deze tag bevatten. Om de grootte van een album te beperken, heb je een manier nodig om eenvoudig alle afbeeldingen met een beoordeling lager dan een bepaalde waarde te verwijderen. Houd er rekening mee dat sommige visuals mogelijk nog geen beoordeling hebben (hun beoordelingswaarde is nog steeds 0) en dat deze nooit mogen worden verwijderd. Voorzie een manier om te melden hoeveel visuals werden verwijderd. Ook heb je de functionaliteit nodig om een slideshow te tonen waarbij alle visuals met minimaal een bepaalde rating in willekeurige volgorde worden afgespeeld of getoond. De volgorde moet elke keer dat je deze methode aanroept anders zijn. De methode returnt de namen van de beelden in de volgorde waarin ze zijn afgespeeld.

Een andere vereiste is de optie om visuals uit een tekstbestand in een album te importeren. Ga ervan uit dat het bestand alle benodigde informatie bevat. Voorzie een check om te kijken hoeveel visuals ingelezen werden.

Een laatste vereiste is om in een album te zoeken naar alle visuals met een grootte die kleiner is dan een bepaalde limiet. Hier wordt de totale grootte van alle geselecteerde visuals weergegeven.

KU LEUVEN

Objectgerichte Softwareontwikkeling 2021-2022

Professoren: S. Desmet en K. Gerling
Labo-assistenten: M. Aufheimer, L. Bruynseels, S. Langendries, K. Pelsmaekers, Z. Sun, R. Vanhoof, N. Vinckier, G. Vranken en J. Wauters.

Festival

Maak een gedetailleerd klassendiagram op basis van de volgende probleembeschrijving: definieer de benodigde klassen, hun attributen, de signatuur van hun methoden en de relaties tussen de verschillende klassen.

Om een optimale beleving te creëren voor alle bezoekers van een festival, willen de organisatoren experimenteren met een systeem waarbij je op het moment dat je je ticket koopt ook zelf de acts kiest die je wilt zien. In deze experimentele versie wordt alleen een geheel getal opgeslagen dat de act vertegenwoordigt. Ze willen echter onderscheid maken tussen 2 soorten tickets. Bij een standaardticket mag je maximaal 10 acts selecteren, een VIP-ticket kent geen beperkingen. Ook is de prijs van beide verschillend en wanneer bezoekers hun ticket willen annuleren, is ook het terugbetalingsbeleid anders. Standaardtickets kunnen niet worden terugbetaald, VIP's zouden de mogelijkheid moeten hebben om een terugbetaling te vragen, die het totale bedrag dat ze hebben betaald teruggeeft. Elk ticket bevat informatie over de naam van de eigenaar, zijn rijksregisternummer en de prijs, die afhankelijk is van het aantal geselecteerde acts. Het moet mogelijk zijn om acts aan een ticket toe te voegen (momenteel alleen het nummer dat de act vertegenwoordigt) en je mag ervan uitgaan dat de opgegeven nummers altijd geldig zijn. Elke act die je aan een ticket toevoegt, verhoogt de totale ticketprijs. Aangezien je niet weet wanneer een ticket compleet is (=alle acts zijn toegevoegd), heb je een manier nodig om dit te beheren. Een standaard ticket moet automatisch op 'voltooid' worden gezet als er 10 acts zijn geselecteerd.

Tickets kunnen dan aan een festival worden toegevoegd. Een festival heeft een vast aantal acts die plaatsvinden. Alleen tickets die “compleet” zijn worden geaccepteerd en om te voorkomen dat dezelfde persoon meerdere tickets koopt, wordt er ook gecontroleerd of een ticket met hetzelfde rijksregisternummer nog niet beschikbaar is. Wanneer een ticket wordt toegevoegd aan het festival, wordt het aantal bezoekers voor alle gekozen acts automatisch bijgewerkt. Je hebt ook de functionaliteit nodig om te vragen naar het actnummer met momenteel het hoogste aantal gegadigden. Bezoekers kunnen hun ticket annuleren, afhankelijk van het type ticket wordt het bedrag terugbetaald en worden de aantallen van aanwezigen bijgewerkt.

Aangezien het nationale nummer is opgebouwd uit de geboortedatum gevolgd door een getal, in het formaat jjjjmmdd-nr, willen de organisatoren de mogelijkheid hebben om te controleren of er op een bepaalde datum ten minste één ticketeigenaar is die jarig is. Om enkele snelle simulaties met het systeem uit te kunnen voeren, willen ze “dummy” tickets kunnen importeren uit een tekstbestand met alle nodige informatie.

KU LEUVEN

Objectgerichte Softwareontwikkeling 2021-2022

Professoren: S. Desmet en K. Gerling
Labo-assistenten: M. Aufheimer, L. Bruynseels, S. Langendries, K. Pelsmaekers, Z. Sun, R. Vanhoof, N. Vinckier, G. Vranken en J. Wauters.

Bibliografie

Maak een gedetailleerd klassendiagram op basis van de volgende probleembeschrijving: definieer de benodigde klassen, hun attributen, de signatuur van hun methoden en de relaties tussen de verschillende klassen.

Je wordt gevraagd een applicatie te ontwikkelen om het beheer van een bibliografie te vergemakkelijken. Je eerste klant heeft alleen papieren publicaties en webpublicaties, maar zorg ervoor dat jouw oplossing eenvoudig uitbreidbaar is naar andere soorten publicaties.

Elke bibliografie wordt geïdentificeerd door de naam van een bepaalde opsteller. Elke publicatie is te herkennen aan een titel, het jaar van uitgave en een unieke identificatiecode. Het moet mogelijk zijn de auteur en alle co-auteurs van een publicatie te bewaren. Voor een papieren publicatie wordt de naam van het tijdschrift waarin deze is gepubliceerd en de impactfactor[1] van dit tijdschrift opgeslagen, voor een webpublicatie de URL en het aantal views. Om de invoer van gegevens te vergemakkelijken, moet het mogelijk zijn om een basiskopie te maken van een bestaande publicatie. Momenteel is deze functie alleen geïmplementeerd voor papieren publicaties.

Je moet publicaties aan de bibliografie kunnen toevoegen. Het moet mogelijk zijn om een string te krijgen met een overzicht van alle publicaties in de bibliografie, waarbij per publicatie minimaal titel en jaartal worden getoond. Het moet mogelijk zijn om de oudste publicatie te zoeken, alle publicaties van een bepaalde (co-)auteur te verwijderen en de verwijderde publicaties als lijst terug te geven. Omdat impactfactoren in de loop van de tijd kunnen veranderen, moet het mogelijk zijn om in één functie alle impactfactoren van een bepaald tijdschrift bij te werken en wil je de som van alle impactfactoren voor alle publicaties weten. Voeg een manier toe om een overzicht van de publicaties in chronologische volgorde af te drukken (oudste eerst); alle publicaties in hetzelfde jaar staan in alfabetische volgorde op titel.

Om het opstarten van het systeem te vergemakkelijken, moet er de mogelijkheid zijn om een lijst van alle publicaties uit een bepaald tekstbestand te importeren.

[1] Opzoeken wat een impactfactor is en hoe deze wordt berekend om te beslissen welk type moet worden gebruikt.