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).

# 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. Momenteel biedt de camping staanplaatsen (lege plekken waar bezoekers hun eigen tent of caravan kunnen plaatsen) en volledig ingerichte tenten. Ze willen echter dat het systeem eenvoudig kan worden uitgebreid naar andere typen, zoals bungalows. Elke beschikbare plaats op de camping heeft een code (bestaande uit een letter en een getal) en een capaciteit (maximaal aantal personen dat van deze plaats gebruik kan maken). Voor staanplaatsen willen ze opslaan of er wel of geen elektriciteit aanwezig is. Voor tenten moet een lijst worden bewaard met mogelijke extra's die aan de tent kunnen worden toegevoegd (babybedje, microgolfoven...). Elke extra is een String. Ook voor tenten wordt het bouwjaar opgeslagen.

De camping is geopend van 1 mei tot 30 september (inclusief). Elke plaats houdt de informatie bij over welke dag deze al bezet is. Mensen kunnen alleen reserveren voor het lopende seizoen. Het moet mogelijk zijn om te controleren of een plaats voor een bepaalde periode vrij is, om te reserveren en om een ​​reservering te annuleren. Perioden worden gedefinieerd door een startdag en een einddag (=dag dat je de camping verlaat, dus de plaats blijft vrij), waarbij de dagen worden genummerd als gehele getallen: 1 mei = 0, 2 mei = 1 enzovoort tot 30 september = 152. Ze willen ook weten hoeveel dagen een plaats bezet is. De basisprijs van een staanplaats is €12/dag, voor tenten €50/dag. Voor tenten krijg je echter 1 dag gratis voor elke week dat je verblijft. Het moet mogelijk zijn om de prijs voor een bepaalde periode te berekenen.

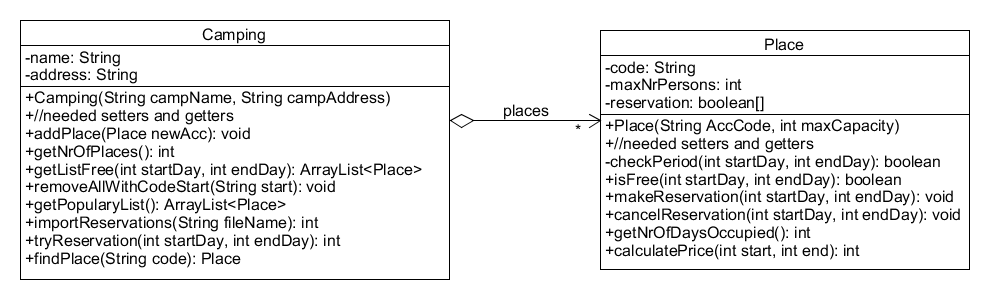
Een camping slaat zijn naam en adres op. Natuurlijk moet je plaatsen (van welk type dan ook) kunnen toevoegen en het totale aantal plaatsen weten. Plaatsen met dezelfde code kunnen niet twee keer worden toegevoegd. Om reserveringen te vergemakkelijken, moet het mogelijk zijn om een ​​lijst te krijgen van alle vrije plaatsen gedurende een bepaalde periode. Ook moet er de mogelijkheid zijn om te controleren of er op de camping een plaats met een bepaalde code beschikbaar is.

Van tijd tot tijd kan de organisatie van de camping veranderen, daarom willen ze de mogelijkheid hebben om alle plaatsen te verwijderen waar de code begint met een bepaalde combinatie. Ze zijn ook geïnteresseerd in de populariteit van de plaatsen, dus willen ze een lijst van alle plaatsen gerangschikt op aantal dagen bezet, met minst bezette plaatsen eerst.

Voor vaste klanten laat de camping toe om reserveringen te maken voor het volgende jaar. Deze informatie wordt opgeslagen in een tekstbestand. Dit bestand wordt dus aan het begin van elk nieuw seizoen gebruikt om bestaande reserveringen te importeren. Deze methode retourneert het aantal geïmporteerde reserveringen. Als er iets misgaat tijdens het importeren, wordt de negatieve waarde van het aantal succesvolle importen geretourneerd.

## Implementatie

Start vanuit het project “campingstart” uit Toledo, en vanuit het volgende klassendiagram. ***Je mag de signatuur van de gegeven methoden en het type van de velden NIET wijzigen. Het kan zijn dat je extra attributen of methodes moet toevoegen om aan alle eisen te voldoen.***



Om te beginnen wordt er geen onderscheid gemaakt tussen staanplaatsen en tenten. Nog wat info over de testen.

* Test3 controleert of je een reservering kunt maken en annuleren en controleert of je het aantal bezette dagen kunt berekenen met getNrOfDaysOccupied(). Voor elke methode waarop je een reservering moet maken, moet je ervoor zorgen dat de opgegeven periode geldig is. Implementeer de benodigde checks in een aparte methode checkPeriod() en roep deze aan waar nodig. Vergeet niet te controleren of een plaats nog vrij is (methode isFree()) als je een nieuwe reservering wilt maken. Als er iets misgaat bij het maken of annuleren van reserveringen, print dan een bericht op het scherm.
* Het formaat van het importbestand is als volgt

- code van plaats

- startdag

- einddag

Deze structuur herhaalt zich een willekeurig aantal keren. De methode returnt het aantal geïmporteerde reserveringen (= aantal perioden). Als de methode ongeldige informatie vindt, returnt deze het aantal reeds uitgevoerde imports als een negatief getal en slaat de import over.

* In test10 krijgen we een lijst van alle plaatsen gerangschikt op populariteit, met de minst populaire plaatsen eerst. De volgorde van plaatsen met dezelfde populariteit is niet gespecificeerd.
* De laatste uitdaging is om de methode int tryReservation(int startDay, int endDay) te implementeren. Met deze methode moet het mogelijk zijn om een ​​oplossing te vinden, ook als er geen plaats vrij is voor de hele periode. De methode zal op meerdere plaatsen naar een oplossing zoeken. Je kunt vanaf de eerste dag beginnen en op zoek gaan naar de plaats met de hoogste beschikbaarheid. Dan herhaal je dit voor de dag die volgt op deze periode en zo verder tot je de hele periode gereserveerd hebt. De methode returnt het aantal benodigde plaatsen. Als er echt geen oplossing wordt gevonden, returnt de methode -1.
* Houd er rekening mee dat als een bepaalde test niet slaagt, de reden tweeledig kan zijn: je hebt iets verkeerd gedaan in je syntaxis (zoals = in plaats van ==) wat resulteert in ander gedrag, of de logica die je hebt geïmplementeerd is niet correct of volledig. Gebruik beschikbare tools om te controleren waar het probleem zich voordoet.

# 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.

## 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.

## 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]](#footnote-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](https://en.wikipedia.org/wiki/Impact_factor) wat een impactfactor is en hoe deze wordt berekend om te beslissen welk type moet worden gebruikt. [↑](#footnote-ref-1)