SE2 BP3 - OO-Programma

Deze toets test alle programmeervaardigheden die je tot op heden bij SE2 hebt verworven. Het is toegestaan om tijdens de toets de voor dit vak gebruikte boeken, aantekeningen en uitwerkingen (op papier of lokaal op laptop) na te slaan. Een mail- of chatclient geopend hebben tijdens de toets is verboden, evenals het gebruik van file sharing tools. Bij constatering van het gebruik van voorgenoemde applicaties wordt de toegang tot de toets ontzegt en hier melding van gemaakt bij de examenkamer.

> ℹ De beschikbare tijd voor deze toets is 200 minuten. Merk op dat er geen pauzes ingeroosterd zijn: de eindtijd valt dus niet gelijk met de standaard lesuren. De toets wordt afgenomen vanaf 12:45u en eindigt om 16:05u.

Beoordeling

Bij deze toets moet je laten zien dat je de bij SE2 geleerde OOP concepten beheerst. Deze concepten komen terug in de voor dit deel van het vak opgestelde rubrics. De beoordeling geschiedt op basis van je prestaties voor deze leerdoelen. Het uiteindelijke cijfer wordt bepaald na afloop van het verplichte feedbackgesprek. Deze feedback wordt mondeling gegeven en kan gebruikt worden voor het uiteindelijke P-assessment.

Per vraag is een aantal minuten bepaald die naar verwachting nodig zijn voor het afronden van dat deel van de toets. De weergegeven tijdsindicatie is bedoeld als hulpmiddel bij het doorlopen van de toets. Dit is een inschatting die je kunt gebruiken om je voortgang tijdens de toets te controleren, zodat je weet of je op schema ligt om alle opgaven af te ronden.

De beoordeling van de toets gebeurt op basis van de rubrics die gedefinieerd zijn voor dit beroepsproduct. Je zult een voldoende behalen voor de toets als ieder leerdoel aangetoond is. Hierbij geldt vanzelfsprekend dat hoe beter en correcter het leerdoel in je oplevering verwerkt is, hoe hoger je uiteindelijke eindscore zal zijn.

Het inleveren van de toets

Na afronding van de implementatie pak je de map waarin je je applicatie ontwikkeld hebt in als zip- of rar-bestand. Dit bestand lever je in op Canvas bij de assignment "Toets object georiënteerd programma".

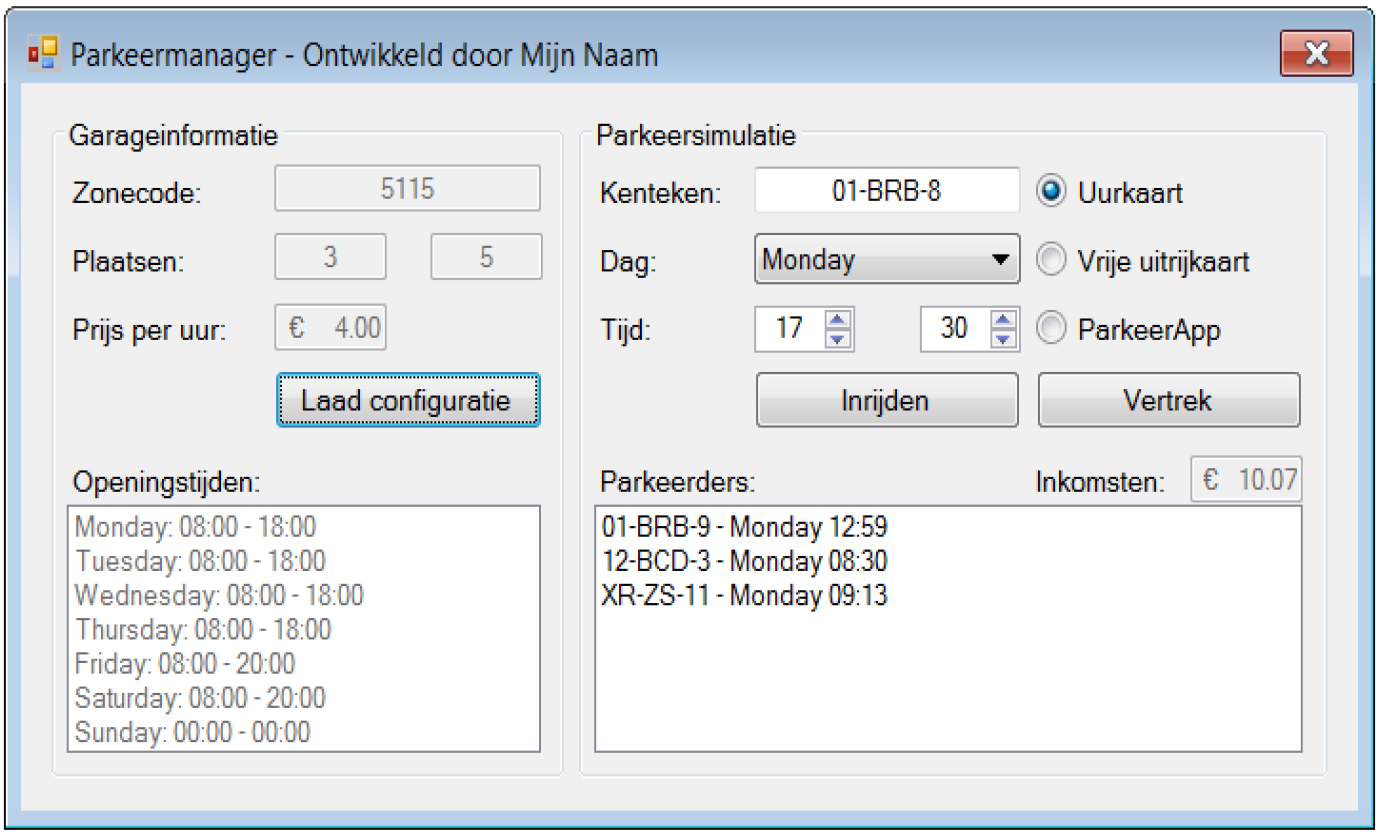
Opdrachtomschrijving

De software voor de parkeergarages van ParkIntVak™ moet op de schop genomen worden. Deze voldoet niet meer aan de eisen die parkeerders verwachten en zal bijgewerkt worden naar hedendaagse standaarden. ParkIntVak™ heeft een veelvoud aan kleinere en grotere parkeergarages verspreid over het land. De software zal dus zo opgezet moeten worden dat deze overal hetzelfde is, maar wel makkelijk te configureren zal zijn aan de hand van de specifieke garage.

Voor de applicatie is de volgende functionaliteit vereist:

- De applicatie moet kunnen omgaan met configuratiebestanden waarin het aantal plaatsen, openingstijden en prijzen vermeldt zijn.
- Er moet een simulatie gemaakt worden waarmee het in- en uitrijden van auto's getest kan worden.
- Aan de hand van de inrij- en vertrektijd en gekozen betaalwijze moet er een te betalen bedrag gerekend worden.
- De applicatie moet robuust en makkelijk te gebruiken zijn: lijsten moeten gesorteerd weergegeven worden, fouten worden naar behoren afgehandeld.
- Om de functionaliteit te borgen dient er gebruik gemaakt te worden van unit testen.

Screenshot

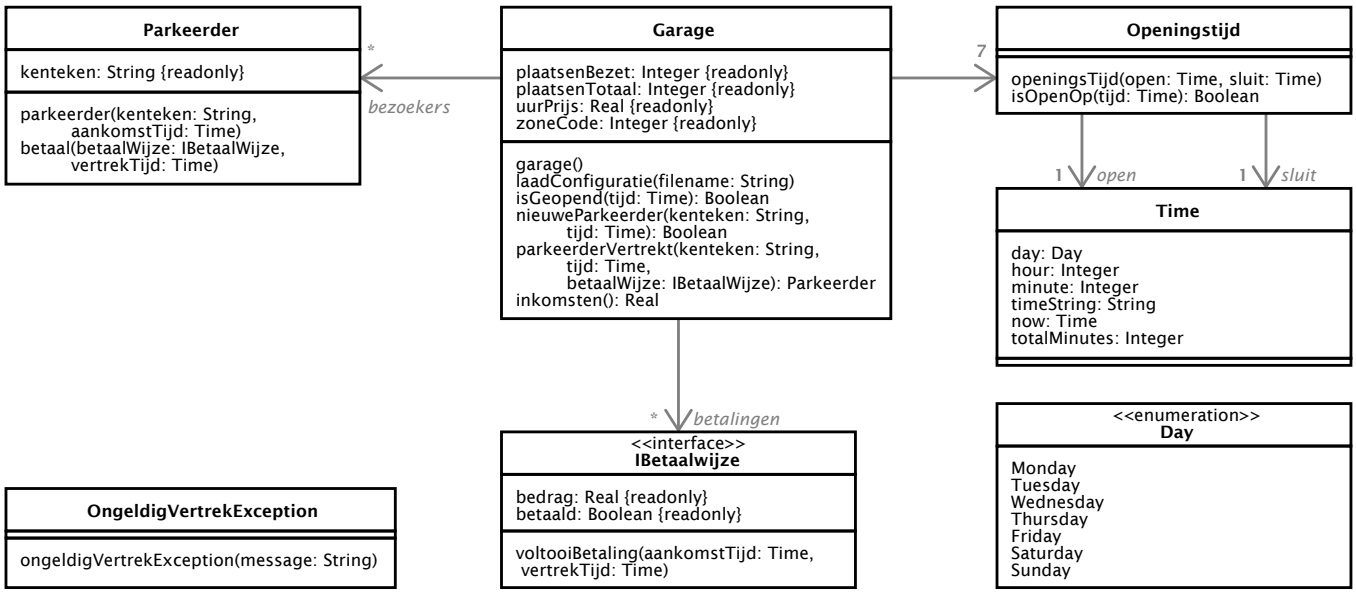


Klassendiagram

Hieronder vind je het domeinmodel wat voor de applicatie opgezet is. Een belangrijke opmerking hierbij: de klasse `Time` en de enumeratie `Day` zijn reeds geïmplementeerd en aanwezig in het project wat bij de toets geleverd is. Daarnaast is de `Time` klasse ook niet volledig gedocumenteerd in dit diagram vanwege de ruimte. Er zijn ook een groot aantal constructors en methodes aangemaakt.

De `Time` klasse is zo opgezet dat je makkelijk met tijden kunt werken. Er zijn op veel verschillende manier instanties aan te maken van deze klassen. Ook is het mogelijk om tijden met elkaar te vergelijken (groter dan, gelijk aan en dergelijke) en tijden bij elkaar op te tellen of juist van elkaar af te trekken.

De overige klassen in het diagram (als ezelsbrug, alle Nederlandstalige klassen) dien je zelf te implementeren. Dit is het begin van de toets: zie hiervoor de volgende pagina.



Opgave 1: Klassenstructuur ± 40 minuten

Realiseer de structuur zoals weergegeven in het domeinmodel. Let op de volgende zaken:

- Zorg dat je naam in de titelbalk van de applicatie komt te staan.
- Implementeer alle weergegeven klassen en relaties.
- Implementeer alle weergegeven properties (bespaar jezelf tijd en gebruik automatic properties waar mogelijk).
- Geef invulling aan de `ToString` methode voor iedere klasse.
- De methodes hoeven nu nog niet van een invulling voorzien te worden (met uitzondering van de eerder genoemde `ToString` methodes).
- Zorg ervoor dat al je klassen `public` zijn zodat je verderop niet tegen problemen aanloopt met toegankelijkheid.
- De `OngeldigVertrekException` dient over te erven van de `Exception` klasse.
- Verderop in de toets dienen mogelijk nog nieuwe klassen toegevoegd te worden. Ook private attributen en methoden moeten mogelijk aangemaakt worden.

Opgave 2: Configuraties laden ± 25 minuten

In de map `configuraties` vind je een aantal bestanden. Deze bestanden stellen verschillende configuraties voor van garages. De bestanden hebben allemaal de structuur zoals in het volgende voorbeeld (maar dan zonder commentaar).

```
5114                                // Zonecode voor kentekenparkeren
10                                  // Aantal plaatsen in de garage
4.80                                // Prijs per uur
Monday:06:00;Monday:18:00           // Openingstijden voor maandag
Tuesday:06:00;Tuesday:18:00         // Openingstijden voor dinsdag
Wednesday:06:00;Wednesday:18:00     // Openingstijden voor woensdag
Thursday:06:00;Thursday:18:00       // Openingstijden voor donderdag
Friday:06:00;Friday:22:00           // Openingstijden voor vrijdag
Saturday:06:00;Saturday:22:00       // Openingstijden voor zaterdag
Sunday:08:00;Sunday:18:00           // Openingstijden voor zondag
```

De verschillende openingstijden komen in een lijst terecht. Bij het uitlezen van de bestanden kunnen de regels met openingstijden gesplitst worden (`String.Split()`) in een openings- en sluitingstijd. De resulterende string kan direct meegegeven worden aan een constructor in de `Time` klasse. Denk ook aan de meegeleverde enumeratie `Day`.

Na het afronden van deze stap moet de volgende functionaliteit geïmplementeerd zijn:

- De gebruiker kan een configuratiebestand selecteren op de harde schijf.
- Fouten bij het inlezen van een bestand worden op een correcte manier afgehandeld.
- De gegevens uit het bestand worden getoond in de relevante controls op het formulier.
- De dagen en tijden voor de openingstijden maken gebruik van de reeds geïmplementeerde `Time` klasse.
- De code voor deze functionaliteit staat op een logische plek conform de opzet van het klassendiagram.

Opgave 3: Het inrijden van auto's ± 50 minuten

Voor de simulatie moeten auto's toegang krijgen tot de garage. Hiervoor zijn drie dingen nodig: een kenteken, datum en tijd. Deze kunnen ingevuld worden met de controls in de groupbox *Parkeersimulatie*. De knop *Inrijden* dient om daadwerkelijk een auto toe te voegen aan de garage. Alle informatie op het scherm dient ook correct te zijn na het toevoegen. Ook willen we de lijst met parkeerders sorteren.

Na het afronden van deze stap moet de volgende functionaliteit geïmplementeerd zijn:

- Auto's worden toegevoegd (inrijden) op basis van de ingevulde informatie op het formulier.
- Auto's kunnen alleen inrijden binnen de gestelde openingstijden.
- Er moet rekening gehouden worden met het maximum aantal beschikbare plaatsen: dit mag niet overschreden worden.
- Kentekens moeten uniek zijn: het is niet mogelijk dat er twee auto's met hetzelfde kenteken zich in de parkeergarage bevinden.
- Het formulier moet bijgewerkt worden met de nieuwe status. Het aantal beschikbare plaatsen moet bijgewerkt worden en daarnaast moet de lijst met parkeerders up to date zijn.
- De lijst met parkeerders moet gesorteerd worden weergegeven. Maak hiervoor een nieuwe klasse aan die instanties van de `Parkeerder`-klasse kan sorteren op kenteken en zorg dat de sortering op het scherm klopt.
- De code voor deze functionaliteit staat op een logische plek conform de opzet van het klassendiagram.

Opgave 4: De betaalwijzen ± 20 minuten

Bij het vertrekken kan aangegeven worden op welke manier er betaald is. We hebben hiervoor een drietal mogelijkheden. Als eerste is er de uurkaart, waarmee de totale parkeertijd per kwartier wordt afgerekend. Dan is er de vrije uitrijkaart: deze wordt door bedrijven verstrekt aan parkeerders zodat deze zonder kosten uit kunnen rijden. Tot slot is er een app: hiermee betalen parkeerders per minuut.

Al deze betaalwijzen moeten in het systeem verwerkt worden. Maak nu de klassen aan die je hiervoor nodig denkt te hebben, inclusief alle noodzakelijke attributen en methodes. De invulling van alles gaan we later aan werken: zorg voor nu dat de definities van de klassen geïmplementeerd zijn. Kijk ook nog eens terug naar het klassendiagram en zorg dat de nieuwe klassen op een logische plek ondergebracht worden.

Houd rekening met de volgende punten:

- Er dienen drie betaalwijzes te zijn: uurkaarten, vrijkaarten en een app waarmee via de telefoon betaald kan worden.
- Je dient zelf de structuur van deze klassen en de integratie ervan in de applicatie te bedenken en verzorgen.
- De klassen en methodes hoeven alleen maar aangemaakt te worden. In opgave 6 gaan we de echte implementatie verzorgen. Verzin voor nu een simpele implementatie om je code te laten compileren.

Opgave 5: Het uitrijden van auto's ± 25 minuten

Na het afronden van deze stap moet de volgende functionaliteit geïmplementeerd zijn:

- Auto's kunnen op ieder tijdstip uitrijden (knop *Vertrek*).
- Afhankelijk van de geselecteerde radiobutton wordt een van de drie betaalwijzen gebruikt. Voor de tijd worden dezelfde controls gebruikt als bij het inrijden.
- Indien de vertrektijd voor de aankomsttijd ligt, dient er een `OngeldigVertrekException` opgegooid en afgehandeld te worden.
- Na het succesvol uitrijden van een auto moet een MessageBox getoond worden met het kenteken, de aankomsttijd, de betaalwijze en het betaalde bedrag.
- Voor de precieze berekening van de parkeerkosten, zie de volgende opdracht.
- Het formulier moet bijgewerkt worden met de nieuwe status. Het aantal beschikbare plaatsen en de lijst met parkeerders moeten up to date zijn, evenals de totale inkomsten.
- De code voor deze functionaliteit staat op een logische plek conform de opzet van het klassendiagram.

Opgave 6: Parkeerkosten berekenen ± 40 minuten

Omdat het van belang is dat de parkeerkosten te allen tijde juist worden berekend, gaan we hier unit testen voor schrijven. Maak test cases aan zodat onderstaande eisen *volledig* worden getest.

Vrije uitrijkaart

- De betaling lukt alleen als de dag van aankomst gelijk is aan de dag van vertrek.
- Het te betalen bedrag is altijd € 0,00.

Uurkaarten

- Als de dag van vertrek niet gelijk is aan de dag van aankomst wordt er een boete betaald van 25 keer het uurtarief.
- De parkeertijd wordt per kwartier berekend. Als iemand 5 minuten geparkeerd heeft, wordt er voor 15 minuten betaald. Is er 20 minuten geparkeerd, wordt er 30 minuten in rekening gebracht. Bij 0 minuten wordt ook 15 minuten betaald; bij precies 15 minuten wordt het 30 minuten betalen.

ParkeerApp

- Als de dag van vertrek niet gelijk is aan de dag van aankomst wordt er een boete betaald van 25 keer het uurtarief.
- De parkeertijd wordt per minuut berekend. Als er 5 minuten geparkeerd zijn wordt het te betalen bedrag $^{5}/_{60}$ van het uurtarief.