Hertoets iooa - opdracht

IOOA Opdracht – Als input voor het assessment

Modulenaam: Object georiënteerd ontwerpen & programmeren 2
Modulecode: iooa
Gelegenheid: 2ste
Datum: Assessments op 10/12 mei
Faculteit: Science & Technology

Studiejaar	:	2022-2023
Moduleleider	:	Vincent Spijkers
Toetsduur	:	Je hebt tot 25 april
Aantal pagina's	:	9
Antwoorden op	:	Uploaden naar CodeGra.de
Kladpapier	:	Wel toegestaan
Bijlagen	:	n.v.t.
Print	:	n.v.t.
Toegestaan	:	Potlood/gum/pen/laptop
Cesuur	:	Bij 55 punten is het cijfer een 5,5
Uitslag	:	Uitslag via Osiris uiterlijk 15 werkdagen na toetsing.
Soort/aantal vragen	:	open vragen
Puntentelling	:	zie hieronder
Toetsvorm	:	Assessment

Aanwijzingen voor de student

- Er zijn 4 opdrachten;
- Je maakt de opdrachten thuis;
- De Javacode die je maakt upload je naar CodeGra.de.
- Daarnaast maak je twee UML-diagrammen, het sequence diagram en het klasse diagram
- Deze diagrammen upload je naar codegra.de als png, jpg of pdf.
- De deadline voor uploaden is 25 april 23:59
- Je mag meerdere keren uploaden, maar denk er wel aan dat je dan weer **alles** uploadt.
- De indeling van de assessments wordt vooraf bekend gemaakt.
- Je neemt je laptop met je code en UML mee naar het assessment

Puntentelling

Voor de opdrachten zelf krijg je geen punten. Het verslag en de code die je gaat maken dienen als input voor het assessment. Tijdens het assessment zullen we je vragen stellen over de gemaakte opdrachten. Aan de hand van het assessment krijg je dan een cijfer.

Hertoets iooa - opdracht

Voor het assessment zijn 100 punten te halen. Het cijfer wordt bepaald door het aantal punten te delen door 10. Cijfers tussen 0,0 en 1,0 worden afgerond op een 1,0.

Plagiaat
Het is natuurlijk makkelijk om de opdrachten samen met anderen te maken of zelfs te laten maken. Dat noemen we plagiaat, en dat is strafbaar.
Bovendien moet je tijdens het assessment allerlei vragen over de opdrachten kunnen beantwoorden, en dat gaat niet lukken als je de opdrachten niet zelf hebt gemaakt.
Houd de eer aan jezelf en voorkom problemen.

Veel succes en plezier met de opdrachten!

Hertoets iooa - opdracht

(1) Opdracht Parkeerplaats Reservering App

[Code uploaden naar Codegra.de]

Beschrijving casus
Voor het bedrijf *VinFlo* Inc. is er een applicatie ontwikkeld die reserveringen kan plaatsten op beschikbare parkeerplekken in de garage van het bedrijf. Omdat er maar een beperkte hoeveelheid plekken beschikbaar zijn moet er een reserveringssysteem gemaakt worden dat de werknemers zelf kunnen reserveren. Voor dit systeem moet er onderscheid gemaakt worden tussen standaard parkeerplekken en parkeerplekken met een elektrische lader.

Het bedrijf heeft al een programmeur in dienst genomen maar deze is er mee gestopt om een bouwvakker te worden. Nu aan jou de taak om de code van deze ex-programmeur aan te vullen en af te maken. De programmeur heeft al een stel klassen afgerond en een todo lijst aangeleverd met instructies die gevolgd moeten worden.

- De programmeur heeft al volgende klassen gemaakt:
  - ParkeerplaatsSeeder- compleet, niets meer aan doen
  - Parkeerplaats – compleet, niets meer aan doen
  - ParkeerplaatsManager– incompleet, afmaken
  - ParkeerplekStandaard- incompleet, afmaken
  - ReserveringsApp– incompleet, afmaken
  - Logger – incompleet, afmaken

- De volgende klassen zijn nog niet geprogrammeerd
  - Reservering
  - ReserveringStandaard
  - ReserveringElektrisch
  - ParkeerplekMetLader
  - GeenBeschikbareParkeerplekExceptie

Je levert alle code op, dus ook van de al afgemaakte klassen.

De todo-list
LET OP: Houd je **exact** aan de gegeven namen van klassen, attributen, methoden, etc. en aan de gegeven te printen teksten

Ga nu werken aan de volgende klassen:

Hertoets iooa - opdracht

hogeschool Leiden

Reservering
Maak de klasse *Reservering*.

- Maak deze klasse abstract
- De klasse heeft de volgende attributen:
  - *naam* van het type String
  - *reserveringsMoment* van het type String
  - *parkeerplaatsNummer* van het type int
- Maak getter methodes voor *naam*, *reserveringsMoment* en *parkeerplaatsNummer*
- Maak de setter methode voor de *parkeerplaatsNummer*
- Maak de toString methode zo dat de standaard uitwerking de volgende string teruggeeft waarbij de naam Floris Admiraal is, reserveringsMoment 12-10-2023 is en het parkeerplaatsNummer 1 is:

```
Reservering{naam='Floris Admiraal', reserveringsMoment='12-10-2023', parkeerplaatsNummer=1}
```

  LETOP, deze waardes zijn variabel.

ReserveringStandaard
Maak de klasse *ReserveringStandaard*.

- Deze klasse erft over van de klasse Reservering
- Maak de constructor van de klasse die een naam, reserveringsMoment ontvangt en deze doorgeeft aan de parent klasse

ReserveringElektrisch
Maak de klasse *ReserveringElektrisch*.

- Deze klasse erft over van de klasse Reservering
- Geef deze klasse het attribuut *voertuigMotorCapiciteitInKw* van het type int
- Maak de getter van het attribuut *voertuigMotorCapiciteitInKw*
- Maak de constructor van de klasse die een naam, reserveringsMoment en een voertuigMotorCapiciteitInKw ontvangt. Geef de juiste attributen door aan de parent klasse
- Maak de toString methode zo dat de volgende string door de functie teruggegeven wordt met naam Floris Admiraal, reserveringsMoment 12-10-2023 is, parkeerplaatsNummer 5 is en voertuigMotorCapiciteitInKw 300 is:

```
ReserveringElektrisch{naam='Floris Admiraal', reserveringsMoment='12-10-2023', parkeerplaatsNummer=5, voertuigMotorCapiciteitInKw=300}
```

  LETOP, deze waardes zijn variabel.

ParkeerplekMetLader
Maak de klasse *ParkeerplekMetLader*

- Deze klasse erft over van de abstracte klasse *Parkeerplaats*
- Geef deze klasse het attribuut *laadsnelheidInKw* van het type int
- Maak de getter van het attribuut *laadsnelheidInKw*
- Maak de constructor van de klasse die een parkeerplaatsNummer en laadsnelheidInKw ontvangt.
- De constructor geeft als *type* de string “lader” door aan de parent
- De constructor geeft ook het *parkeerplaatsNummer* door aan de parent

GeenBeschikbareParkeerplekExceptie
Maak de klasse *GeenBeschikbareParkeerplekExceptie*

- Maak van deze klasse een java exceptie klasse
- De constructor van deze klasse ontvangt een *parkeerplekType* parameter van het type String

- Laat de exceptie het volgende op het scherm tonen als deze exceptie zich voordoet, waarbij *parkeerplekType* variabel is:

  Helaas zijn er geen beschikbare parkeerplekken meer voor type: {{parkeerplekType}}

ParkeerplekStandaard

Completeer de klasse *ParkeerplekStandaard*

- De klasse ontvangt in de constructor *parkeerplaatsNummer* van het type int als parameter
- De constructor geeft als *type* de string “standaard” door aan de parent
- De constructor geeft ook het *parkeerplaatsNummer* door aan de parent

ParkeerplaatsManager

Completeer de klasse *ParkeerplaatsManager*

- Completeer de constructor van deze functie zodat:
  - deze eerst een ParkeerplaatsSeeder object aanmaakt
  - hierna het parkeerplaatsen attribuut vult met parkeerplaatsen die uit de seeder komen
- Completeer de functie *reserveerParkeerplaats(Reservering reservering)*
  - Haal een parkeerplek object op met behulp van de *krijgParkeerplaats* functie
    - Geef deze functie het type parkeerplaats mee uit de reservering parameter
  - Zet het parkeerplaatsnummer uit de parkeerplek in de reservering parameter
  - Zet op de parkeerplek de gereserveerdeNaam op de naam uit de reservering parameter
  - Zet de parkeerplek op gereserveerd
  - Roep de printReservering methode aan
  - Roep de logReservering functie aan op het logger klasse attribuut en geef de reservering mee
- Completeer de functie *reserveerParkeerplaats(ReserveringElektrisch reservering)*
  - Haal een parkeerplek object op met behulp van de *krijgParkeerplaats* functie
    - Geef deze functie het type parkeerplaats mee uit de reservering parameter
  - Zet het parkeerplaatsnummer uit de parkeerplek in de reservering parameter
  - Zet op de parkeerplek de gereserveerdeNaam op de naam uit de reservering parameter
  - Zet de parkeerplek op gereserveerd
  - Bereken een oplaadDuur van het elektrische voertuig
    - Berekening is: motorCapiciteitInKw / laadsnelheidInKw
    - Rond dit getal af op 2 decimale achter de komma
  - Print de volgende tekst naar de console:
    De lader charged met {{laadsnelheidInKw}} kw, je voertuig is in {{oplaadDuur}} uur opgeladen
  - Roep de printReservering methode aan
  - Roep de logReservering functie aan op het logger klasse attribuut en geef de reservering mee
- Completeer de functie *krijgParkeerplaats*
  - Loop met de functie door de lijst van parkeerplekken
    - Als het parkeerplek type overeenkomt met de parameter type **en** als de parkeerplaats niet bezet is, return de parkeerplaats
    - Als er niet aan deze conditie voldaan wordt moet de *GeenBeschikbareParkeerplekException* op gegooid worde

Hertoets iooa - opdracht

Logger

Completeer de klasse *Logger*

- Maak de functie logReservering
  - De functie is van het type void en ontvangt een standaard reservering als parameter
  - De functie print de uitkomst van de toString methode van de reservering naar de console met als prefix de string “LOG:”
  - Bijvoorbeeld:

```
LOG: Reservering{naam='Vincent Spijkers', reserveringsMoment='03-02-2023', parkeerplaatsNummer=1}
```

- Maak de functie logReservering
  - De functie is van het type void en ontvangt een elektrische reservering als parameter
  - De functie print de uitkomst van de toString methode van de elektrische reservering naar de console met als prefix de string “LOG:”
  - Bijvoorbeeld:

```
LOG: ReserveringElektrisch{naam='Floris Admiraal', reserveringsMoment='12-10-2023', parkeerplaatsNummer=5, voertuigMotorCapiciteitInKw=300}
```

ReserveringsApp

Completeer de klasse *ReserveringsApp*

- Completeer de functie *reserveerAutoPlek*
  - Vraag de gebruiker de naam voor de reservering
    - Output tekst: *“Onder welke naam moet de reservering komen?”*
  - Vraag de gebruiker om de reserverings datum.
    - Deze datum moet in het volgende format ingevuld worden: dd-mm-yyyy -> 10-03-2023 het is dus een string
    - Output tekst: *“Voer de datum in voor het reserveren”*
  - Maak een standaard reserverings object aan met de opgegeven data
  - Roep op de parkeerplaatsManager van de klasse de reserveerParkeerplaats functie aan en geef de reservering mee
  - Vang een mogelijke exceptie hierop, als de exceptie fout gaat moet de message van de exceptie op het scherm geprint worden

- Completeer de functie *reserveerLaadpaalPlek*
  - Vraag de gebruiker de naam voor de reservering
    - Output tekst: *“Onder welke naam moet de reservering komen?”*
  - Vraag de gebruiker de motor capiciteit in KW van het voertuig
    - Output tekst: “*Wat is de motor capiciteit in KW van uw voertuig?”*
  - Vraag de gebruiker om de reserverings datum.
    - Deze datum moet in het volgende format ingevuld worden: dd-mm-yyyy -> 10-03-2023 het is dus een string
    - Output tekst*: “Voer de datum in voor het reserveren”*
  - Maak een elekterische reserverings object aan met de opgegeven data
  - Roep op de parkeerplaatsManager van de klasse de reserveerParkeerplaats functie aan en geef de reservering mee
  - Vang een mogelijke exceptie hierop, als de exceptie fout gaat moet de message van de exceptie op het scherm geprint worden

Hertoets iooa - opdracht

hogeschool Leiden

Output van werkende applicatie:

```
Welkom bij de reserverings app van VinFlo inc. Wat wil je reserveren?
1. Auto parkeerplek
2. Laadpaal parkeerplek
3. Afsluiten
1
Onder welke naam moet de reservering komen?
Vincent Spijkers
Voer de datum in voor het reserveren
12-01-2023
De reservering onder naam Vincent Spijkers is gereserveerd voor parkeerplek 1 van het type standaard
LOG: Reservering{naam='Vincent Spijkers', reserveringsMoment='12-01-2023', parkeerplaatsNummer=1}
```

Figuur 1 Keuze 1, Vincent Spijkers, 12-01-2023, parkeerplek beschikbaar

```
Welkom bij de reserverings app van VinFlo inc. Wat wil je reserveren?
1. Auto parkeerplek
2. Laadpaal parkeerplek
3. Afsluiten
2
Onder welke naam moet de reservering komen?
Floris Admiraal
Wat is de motor capiciteit in KW van uw voertuig?
300
Voer de datum in voor het reserveren
13-05-2023
De lader charged met 75kw, je voertuig is in 4.0 uur opgeladen
De reservering onder naam Floris Admiraal is gereserveerd voor parkeerplek 5 van het type lader
LOG: ReserveringElektrisch{naam='Floris Admiraal', reserveringsMoment='13-05-2023', parkeerplaatsNummer=5, voertuigMotorCapiciteitInKw=300}
```

Figuur 2 Keuze 2, Floris Admiraal, 13-05-2023, elektrische lader beschikbaar

```
Welkom bij de reserverings app van VinFlo inc. Wat wil je reserveren?
1. Auto parkeerplek
2. Laadpaal parkeerplek
3. Afsluiten
2
Onder welke naam moet de reservering komen?
Bas van der Veen
Wat is de motor capiciteit in KW van uw voertuig?
250
Voer de datum in voor het reserveren
22-06-2023
Helaas zijn er geen beschikbare parkeerplekken meer voor type: lader
```

Figuur 3 Keuze 2, Bas van der Veen, 250, 22-06-2023, geen plek meer beschikbaar in het systeem

```
Welkom bij de reserverings app van VinFlo inc. Wat wil je reserveren?
1. Auto parkeerplek
2. Laadpaal parkeerplek
3. Afsluiten
3
```

Figuur 4 Keuze 3, app sluit af

Hertoets iooa - opdracht

(2) Opdracht UML klassendiagram (applicatiemodel)

Maak het applicatiemodel van de *Parkeerplaats Reservering App*. Upload deze naar Codegra.de als *applicatiemodel.png* of *applicatiemodel.jpg* of *applicatiemodel.pdf*

(3) Opdracht UML Sequence diagram

Maak het sequence diagram van het reserveren van een parkeerplek met lader vanuit de *Parkeerplaats Reservering App,*

Upload deze naar Codegra.de als *sequence-diagram.png* of *sequence-diagram.jpg* of *sequence-diagram.pdf*

(4) Opdracht Timers

[Code uploaden naar Codegra.de]

Casus
Timers zijn erg nuttig tijdens het koken, om bijvoorbeeld bij te houden hoe lang iets in de oven moet. Wanneer je aan het koken bent wil je soms verschillende timers tegelijkertijd laten lopen, bijvoorbeeld als je pasta een aantal minuten moet koken terwijl je saus voor een andere tijdseenheid laat sudderen.

In deze opdracht ga je een timer maken in Java met behulp van threads, ook zorg je ervoor dat je verschillend timers tegelijkertijd kan laten lopen

De klasse Timer

- De instanties van de timer runnen op een thread;
- Elke timer loopt af van een bepaald aantal seconden, de hoeveelheid seconden wordt meegeven tijdens het aanmaken.
- Wanneer de timer gestart wordt zal hij beginnen met aftellen vanaf het gegeven aantal seconden.
- Zodra de timer de nul heeft bereikt zal het melden:
  De teller die stond voor (aantalSeconden) seconden is klaar met tellen!

De klasse TimerApp

- Deze klasse heeft de method main();
- In de klasse worden twee Timer instanties gemaakt;
- De twee Timer instanties hebben verschillende aantal seconden waarvan ze gaan aftellen, bijvoorbeeld 5 en 10
- Zodra de timers gestart zijn zal dit geprint worden in de console met het volgende bericht:
  De timers zijn gestart!

- Pas wanneer allebei de timers klaar zijn met tellen moet het volgende bericht geprint worden:
  Allebei de timers zijn klaar met tellen!

Tenslotte
Upload al je werk naar codegra.de, bereid je goed voor op het assessment de nadruk van het vak ligt op het kunnen uitleggen van je keuzes en waarom deze keuzes dan ook goed zijn. Denk aan de theorie die geleerd is de SOLID principes en andere programmeer theorieën zoals polymorfisme en encapsulatie zullen terugkomen op je assessment.