Datum examen dinsdag 24 januari 2023

Academiejaar 2022-2023

Opleiding	Toegepaste informatica	**Jaar**	1
Opleidingsonderdeel	OOSD I		
Lesgever(s)	Chloé De Leenheer - Liesbeth Lewyllie – Irina Malfait – Heidi Roobrouck – Stefaan Samyn – Fien Spriet – Sonia Vandermeersch (Sofie Lambert) - Sharon Van Hove - Leen Vuyge		

Behaald resultaat

/ 35

☐ Tijdens het examen mogen **geen** hulpmiddelen gebruikt worden.

☒ Tijdens het examen mogen deze hulpmiddelen gebruikt worden:

1. Browser
   a. https://exam.hogent.be
   b. https://docs.oracle.com/en/java/javase/17/docs/api/allclasses-index.html
2. Pdf reader
   a. Alle pdf's gepubliceerd op Chamilo (theorie – voorbeeldoefeningen – oefeningen); chamilo zelf raadpleeg je NIET en afgedrukte versies van de pdfs mogen NIET gebruikt worden!
   b. Zelfgemaakte, al dan niet ingescande, samenvattingen in pdf
3. Eclipse (of alternatieve IDE) en Visual Paradigm (geïntegreerd of stand alone)
   a. Uitwerken van de examenvragen
   b. Alle gemaakte oefeningen / voorbeelden
4. Handboek "Java How to Program Early Objects" of een ander aangekocht boek

Algemene richtlijnen

Heb je individuele onderwijs- en examenmaatregelen, noteer dan in de rechterbovenhoek van je antwoordpapier 'IOEM' (afkorting voor individuele onderwijs- en examenmaatregelen).

Je mag geen enkele vorm van communicatie – noch offline noch online – gebruiken tijdens dit examen, tenzij anders aangegeven in de exameninstructies. Mobiele telefoons, smartwatches en dergelijke moeten uitgeschakeld zijn (niet op stil, trillen, vliegtuigstand, …). Ze mogen tijdens het examen ook niet gebruikt worden om de tijd te raadplegen. Het niet volgen van de gedragscode wordt gesanctioneerd als examenfraude.

Lees dit aandachtig vooraleer je start

Antwoordformulier.

- In je mapje met je opdracht vind je Antwoordformulier_OOSDI_deel2.txt
- Hernoem dit document (selecteer het document - rechtermuisklik – naam wijzigen) naar

> "Klas_Deel2_Naam_Voornaam.txt"
>
> *(bv. 1.08_Deel2_Jansens_Jan.txt)*

- Open nu het antwoordformulier en vul bovenaan je naam en voornaam in.
- Kopieer voor elke vraag de gevraagde inhoud op de juiste plaats in dit document. Bewaar het document op regelmatige basis!

Vragen.

- In dit document vind je de vragen.

Indienen.

- Denk er aan je antwoordformulier nog een laatste maal te **bewaren** alvorens je **je teksteditor afsluit** en gaat indienen.
- Ga terug naar de opdracht op exam.hogent.be en dien het antwoordformulier en het .vpp-bestand in.

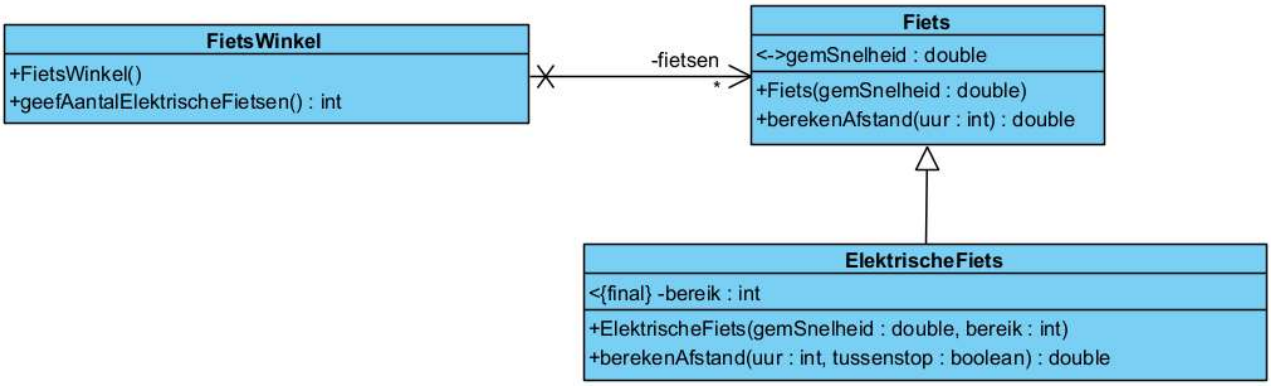
- Respecteer de deadline om in te dienen!

> **12u00**

Vraag 1. Vul de ontbrekende stukken code aan. (10p)

Vul de code aan rechtstreeks op het antwoordformulier!
Tip: Je hebt geen tijd om al deze code over te nemen in Eclipse.

Als je geen code moet aanvullen zet je specifiek "Niets aan te vullen"



Code 1: Vul de definitie van de klasse/klassen aan obv de UML

Code 2: Vul de methodesignatuur voor de setter aan.

Code 3: Vul de constructor aan, alle attributen moeten correct geïnitialiseerd worden.

Code 4: Vul aan zodat de potentiële totale afstand op basis van het uur en de snelheid zo efficiënt mogelijk wordt berekend. In de volgende lijnen code zal dan gekeken worden of deze totale afstand mogelijk is met het bereik van de batterij. Noteer ook de annotatie indien van toepassing.

Code 5: Werk de methode uit zodat het aantal elektrische fietsen in de lijst van fietsen wordt berekend.

```
public class Fiets      1a
{
    private double gemSnelheid;

    public Fiets(double gemSnelheid) {
        setGemSnelheid(gemSnelheid);
    }

    public double berekenAfstand(int uur) {
        return gemSnelheid*uur;
    }

    public double getGemSnelheid() {
        return this.gemSnelheid;
    }

    2
        this.gemSnelheid = gemSnelheid;
    }

}
```

```
public class ElektrischeFiets
{

    private final int bereik;

    public ElektrischeFiets(double gemSnelheid, int bereik) {

    }

    public double berekenAfstand(int uur, boolean tussenstop) {
        double potentieleAfstand =
        int bereikVoorDezeRit = tussenstop? bereik*2: bereik;
        return Math.min(potentieleAfstand, bereikVoorDezeRit);
    }

    public int getBereik() {
        return this.bereik;
    }
}
```

1b

3

4a

4b

```
public class FietsWinkel {

    private List<Fiets> fietsen;

    /*
     * De lijst met fietsen wordt in de constructor ingevuld vanuit de database.
     * Deze methode wordt niet getoond wegens niet relevant.
     */
    public FietsWinkel() {

    public int geefAantalElektrischeFietsen() {

    }
}
```

5

Vraag 2: Stel het DCD op (15 p)

Use Case: Schaar-Steen-Papier wedstrijd

Situering: De speler speelt tegen de computer een wedstrijd Schaar-Steen-Papier. Een wedstrijd bestaat uit een oneven aantal spelletjes. In een spel kiest de speler uit de 3 mogelijkheden (schaar, steen of papier) en wordt voor de computer een random keuze gegenereerd. Schaar wint van papier, steen wint van schaar en papier wint van steen. Als de speler en de computer hetzelfde hebben gekozen, is er nog geen winnaar. In elk spel wordt gespeeld tot er een winnaar is en deze krijgt dan 1 punt. Van zodra duidelijk is dat de speler de computer niet meer kan inhalen (of omgekeerd), is de winnaar van de wedstrijd gekend.

Primaire actor: de speler

Preconditie: geen

Postconditie: er is een winnaar van de wedstrijd

1. **Normaal verloop**
    1. Het systeem vraagt de naam van de speler
    2. De speler geeft de naam in
    3. Het systeem vraagt het aantal te spelen spelletjes
    4. De speler geeft het aantal spelletjes in [DR_AANTAL_SPELLETJES]
    5. Het systeem valideert
    6. Het systeem vraagt de keuze van de speler
    7. De speler geeft zijn keuze in [DR_KEUZE_SPELER]
    8. Het systeem valideert
    9. Het systeem bepaalt de keuze van de computer en bepaalt indien mogelijk de winnaar van het spel [DR_KEUZE_COMPUTER][DR_WINNAAR_SPEL]
    10. Het systeem toont de spelsituatie [DR_TOON_SPELSITUATIE]
    11. Zolang het spel nog niet ten einde is, ga terug naar stap 6 [DR_EINDE_SPEL]
    12. Het systeem toont de winnaar van het spel
    13. Het systeem toont de tussenstand van de wedstrijd
    14. Zolang de wedstrijd nog niet ten einde is, ga terug naar stap 6 [DR_EINDE_WEDSTRIJD]
    15. Het systeem toont de winnaar van de wedstrijd [DR_WINNAAR_WEDSTRIJD]

2. **Alternatieve verlopen**
   Niet relevant voor deze opgave

3. **Domeinregels**
   DR_AANTAL_SPELLETJES
   Het aantal spelletjes moet minimum 1 en kan maximum 25 zijn.
   Het aantal spelletjes moet een oneven getal zijn.

   DR_KEUZE_SPELER
   De speler kan kiezen uit schaar, steen of papier. Hiervoor geeft hij een getal (1 t.em. 3) in.

   DR_KEUZE_COMPUTER
   Ook de computer kan kiezen uit schaar, steen of papier. Zijn keuze wordt random bepaald.

   DR_WINNAAR_SPEL
   Als de speler en de computer dezelfde keuze maken, is er nog geen winnaar.
   Anders is de winnaar:

* degene die schaar kiest als de keuzes schaar en papier zijn
* degene die papier kiest als de keuzes papier en steen zijn
* degene die steen kiest als de keuzes steen en schaar zijn

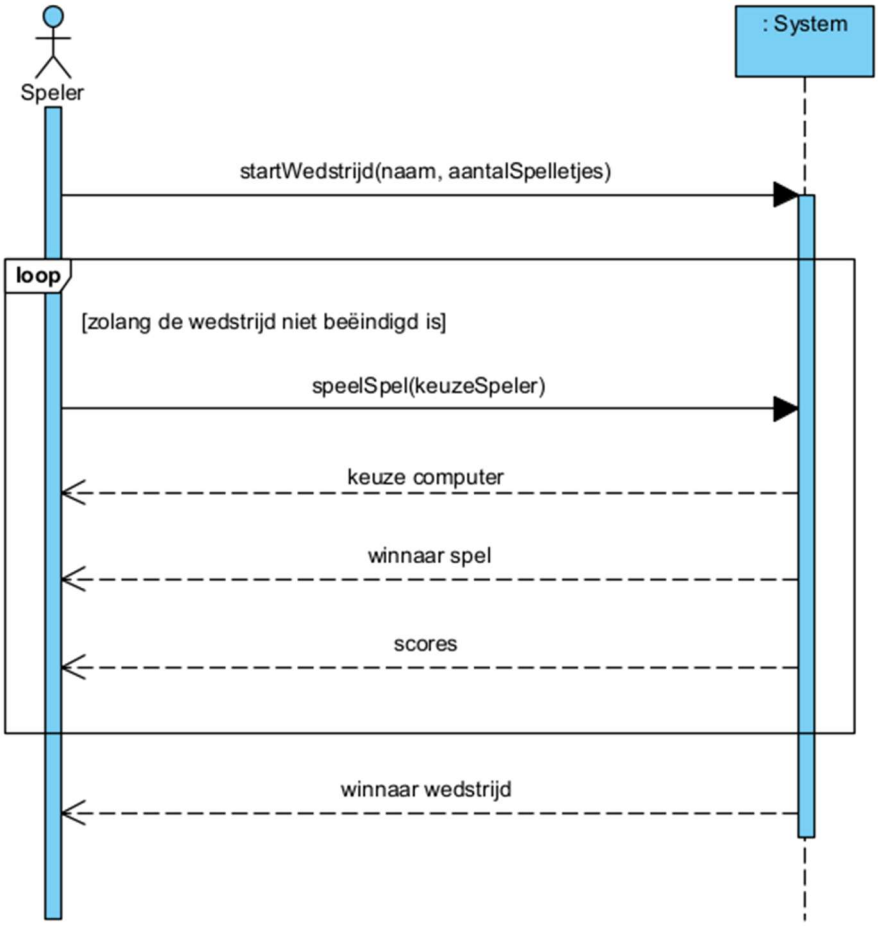
DR_TOON_SPELSITUATIE
De keuze van de computer wordt getoond alsook de scores en de eventuele winnaar van het spel.

DR_EINDE_SPEL
Het spel eindigt van zodra er een winnaar is. Deze krijgt dan 1 punt.

DR_EINDE_WEDSTRIJD
De wedstrijd eindigt indien de score van de speler zo hoog is dat de computer hem niet meer kan inhalen, zelfs al zou hij vanaf nu alle spelletjes winnen. Of omgekeerd, indien de score van de computer zo hoog is dat de speler hem niet meer kan inhalen. Dit doet zich met andere woorden voor wanneer de speler of de computer meer dan de helft van de voorziene spelletjes gewonnen heeft.

DR_WINNAAR_WEDSTRIJD
De winnaar is de speler als die meer dan de helft van de voorziene spelletjes gewonnen heeft, of de computer indien hij een dergelijke score heeft.

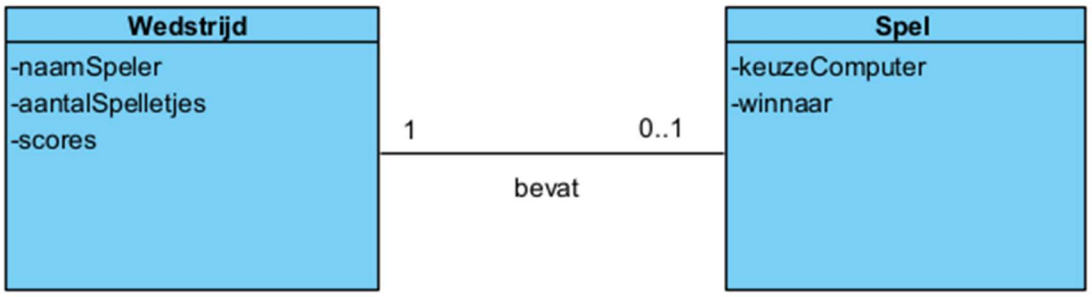
ANALYSE – SSD en OC



Operation	startWedstrijd(naam, aantalSpelletjes)
Cross references	UC Schaar-Steen-Papier wedstrijd
Preconditions	/
Postconditions	• Er werd een Wedstrijd w aangemaakt met de opgegeven naam voor de speler en met het gevraagde aantal spelletjes. • In het attribuut scores van w werd de score van de speler en de computer op 0 ingesteld.

Operation	speelSpel(keuzeSpeler)
Cross references	UC Schaar-Steen-Papier wedstrijd
Preconditions	Er is een Wedstrijd w gestart
Postconditions	• Er werd een Spel huidigSpel aangemaakt. • Het huidigSpel werd met de wedstrijd geassocieerd. • In huidigSpel werd het attribuut keuzeComputer ingesteld (zie DR_KEUZE_COMPUTER). • Het attribuut winnaar van huidigSpel kreeg een waarde (zie DR_WINAAR_SPEL). • Indien een winnaar voor het Spel gekend is werd het attribuut scores van w aangepast (zie DR_EINDE_SPEL).

ANALYSE – Domeinmodel

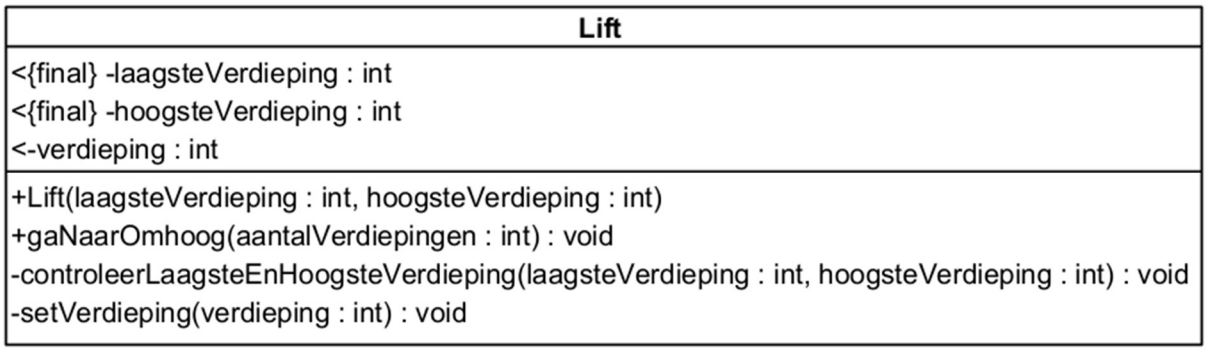


Gevraagd
Maak, gebaseerd op de Use Case en de analyse die hierboven beschreven staat, het ontwerp. Het resultaat is een **DCD**. Je maakt gebruik van Visual Paradigm om dit klassendiagram te maken. Noem je bestand NaamVoornaam_DCD.vpp *(bv. studente Roos Gecko slaat haar bestand op als GeckoRoos_DCD.vpp).*

Belangrijk: Je moet dit **.vpp bestand** uploaden als antwoord op deze vraag. Screenshots zullen niet aanvaard worden.

Vraag 3: Testen (10p)

Gegeven: Onderstaand UML klassendiagram voor de klasse Lift en de Java code die werd gegenereerd door VP in het startproject. Onder de UML vind je ook nog een tekstuele toelichting en domeinregels.



Een lift wordt gemaakt voor een gebouw met een hoogste en een laagste verdieping. Een lift kan een aantal verdiepingen omhoog gaan.

Domeinregels - Lift

- **laagsteVerdieping** moet een getal strikt kleiner dan 0 zijn
- **hoogsteVerdieping** moet een getal strikt groter dan 0 zijn
- **verdieping** moet in het interval [laagsteVerdieping, hoogsteVerdieping] liggen en staat bij een nieuwe lift automatisch ingesteld op 0

Er worden IllegalArgumentExceptions geworpen indien niet aan deze domeinregels is voldaan

Domeinregels – gaNaarOmhoog

- een geldig **aantalVerdiepingen** is strikt groter dan 0 (*er wordt een IllegalArgumentException geworpen indien aantalVerdiepingen niet geldig is)*
- de verdieping van de lift wordt vermeerderd met het opgegeven **aantalVerdiepingen**.
  let op: de lift plafonneert op de **hoogsteVerdieping**, dit betekent dus dat de lift nooit hoger zal gaan dan de hoogste verdieping

Gevraagd:

Deel 1: **Ontwerp test cases**. Op het antwoordformulier vind je een template voor test cases voor de **constructor Lift**. Geef voor elke test case een waarde op voor de parameters laagsteVerdieping en hoogsteVerdieping en omschrijf wat je verwacht van de test case. Voor dit laatste beperk je je tot 'MaaktLift' of 'WerptException'.

Deel2: **Implementatie unit testen.** Maak een **klasse LiftTest** en geef een volledige implementatie voor de unit testen voor de **methode gaNaarOmhoog**. Maak gebruik van geparameterizeerde testen waar mogelijk. Plak de code van deze test klasse in het antwoordformulier.

Maak voor beide delen gebruik van EP en BVA om relevante test-cases/unit-testen af te leiden. De domein klasse Lift hoef je **NIET** te implementeren!