

☐ Voorblad in te leveren door student? Zo ja ☒, dan volgende blok invullen

| Studentgegevens                                      |  |                         |   |
|--|--|-------------------------|---|
| Naam student   |  |                         |   |
| Studentnummer  |  | Klas                    |   |
| Kruis hiernaast de naam van de eigen vakdocenten aan |  | Johan Talboom           | <input checked="" type="checkbox"/> Ik volgde geen les  |
|  |  | Etiënne Goossens        | <input checked="" type="checkbox"/> Ik weet het niet    |
|  |  | Jessica van der Heijden | <input checked="" type="checkbox"/> Hans van der Linden |

Vermeld op ieder blad je naam, studentnummer en klas

| Algemene gegevens van de toets       |  |                |                         |         |     |
|--------------------------------------|--|----------------|-------------------------|---------|-----|
| Academie                             | AE&I   | Fase           | P                       | Periode | 1.2 |
| Opleiding                            | Technische Informatica   | Onderwijsvorm  | VT                      |         |     |
| Naam vak                             | Introductie Software Development in Java                               | Vakcode        | EITI-ISDJ               |         |     |
| Subtitel                             | Deel A - Theorie   | Auteur(s)      | Johan Talboom           |         |     |
| Datum                                | vr 21 januari 2022   |                | Hans van der Linden     |         |     |
| Tijd                                 | 14:00 – 17:20 (200 min)  | Review door    | Jessica van der Heijden |         |     |
| Aantal pagina's (inclusief voorblad) | 13   | Aantal opgaven | 31                      |         |     |
| Zak/slaaggrens                       | Elke opgave is 1 punt waard; 20 punten of meer levert een voldoende op |                |                         |         |     |

| Bijzonderheden                           |  |  |
|--|--|--|
| Papier (kruis aan wat van toepassing is) | <input checked="" type="checkbox"/> Lijntjes A4  | <input checked="" type="checkbox"/> Kladpapier |
|  |  | <input type="checkbox"/> Ruitjes A4            |
|  | <input checked="" type="checkbox"/> Schrapkaart a-e mk ned   |  |
| Opgave inleveren?                        | <input checked="" type="checkbox"/> (☒=ja ☐=nee)   |  |
| Rekenmachine                             | <input checked="" type="checkbox"/> Technische Informatica: géén rekenmachine toegestaan   |  |
| Toegestane hulpmiddelen                  | <input checked="" type="checkbox"/> Geen hulpmiddelen toegestaan bij dit deel A  |  |
|  | <input type="checkbox"/> Boeken (titel, auteur) boek titel, auteur, druk   |  |
|  | <input type="checkbox"/> Anders nl: anders   |  |
| Opmerkingen                              | Eerst dit deel A maken op papier zonder hulpmiddelen en inleveren. Pas daarna ontvangt de student deel B en mogen de laptop en andere hulpmiddelen gebruikt worden om de uitwerking te maken en via Brightspace in te leveren. |  |
| Contactgegevens academiebureau AE&I      | Gebouw LA, office LA003<br>Olaf van Maurik: 088-5259293<br>Marina Hendrickx: 088-5257198<br>Gebouw CHL 13<br>Barbara Klijs: 088-5257494  |  |
|  | Vakdocent (naam en tel.nr.):<br>Hans van der Linden<br>088-525 7234  |  |

Elke student wordt geacht de bepalingen m.b.t. het afleggen van de toetsen te kennen.

|          |  |
|----------|--|
| Vak      | Introductie Software Development in Java |
| Subtitel | Deel A - Theorie                         |
| Vakcode  | EITI-ISDJ                                |
| Datum    | 21 jan 2022                              |
| Pagina   | 1 van 12                                 |

## Instructie

Dit tentamen maak je in twee delen: deel A en deel B. Voor deel A krijg je een cijfer en voor deel B ook. Beide cijfers tellen voor 50% mee in je eindcijfer.

Dit deel A bevat vragen om je theoriekennis te testen. Bij de meerkeuzevragen is telkens maar één antwoord juist.

Noteer de antwoorden op de schrapkaart. Als je meer ruimte nodig hebt voor bijvoorbeeld opmerkingen bij antwoorden, gebruik daarvoor dan het extra lijntjespapier. Zorg dat je naam staat op elk vel dat je inlevert!

Het tentamen maak je aan de hand van een casus. Die casus wordt zowel in deel A als deel B gebruikt. Het is daarom belangrijk dat je de casus goed bestudeert. Daarnaast bevat dit deel A ook algemene theorievragen die onafhankelijk zijn van de casus

## Succes!

## De casus: RocketLaunch

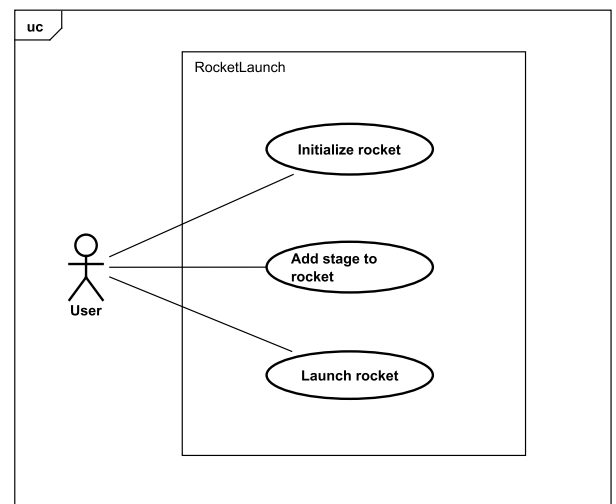
In dit tentamen werk je aan een simulatie van een raketlancering. Het model en de code zijn geschreven in het Engels, dus we zullen hier de Engelse termen gebruiken.

De raket (Engels: rocket) bestaat uit een aantal op elkaar gestapelde trappen die we 'stages' noemen. Elke trap bevat raketmotoren die tijdens de lancering na elkaar branden. Als de eerste trap is opgebrand neemt de tweede trap het over, enz. totdat alle trappen zijn opgebrand. Er zijn in deze casus twee verschillende soorten rakettrappen (stages): een solid fuel stage en een liquid fuel stage. Elke stage heeft fuel die tijdens de launch stap voor stap vermindert tot de stage is opgebrand (exhausted).

Voor deze casus zijn diverse UML diagrammen beschikbaar waarin de software is gemodelleerd. Enkele van die diagrammen zijn hier getoond, bijvoorbeeld hier rechts een use-case diagram met de drie use cases van dit RocketLaunch systeem.

De diagrammen en de beschrijving van de casus zijn zowel in deel A als deel B opgenomen dus je hoeft de casus niet uit je hoofd te leren.

Bestudeer de beschrijving en de diagrammen goed bij het beantwoorden van de vragen van deel A.

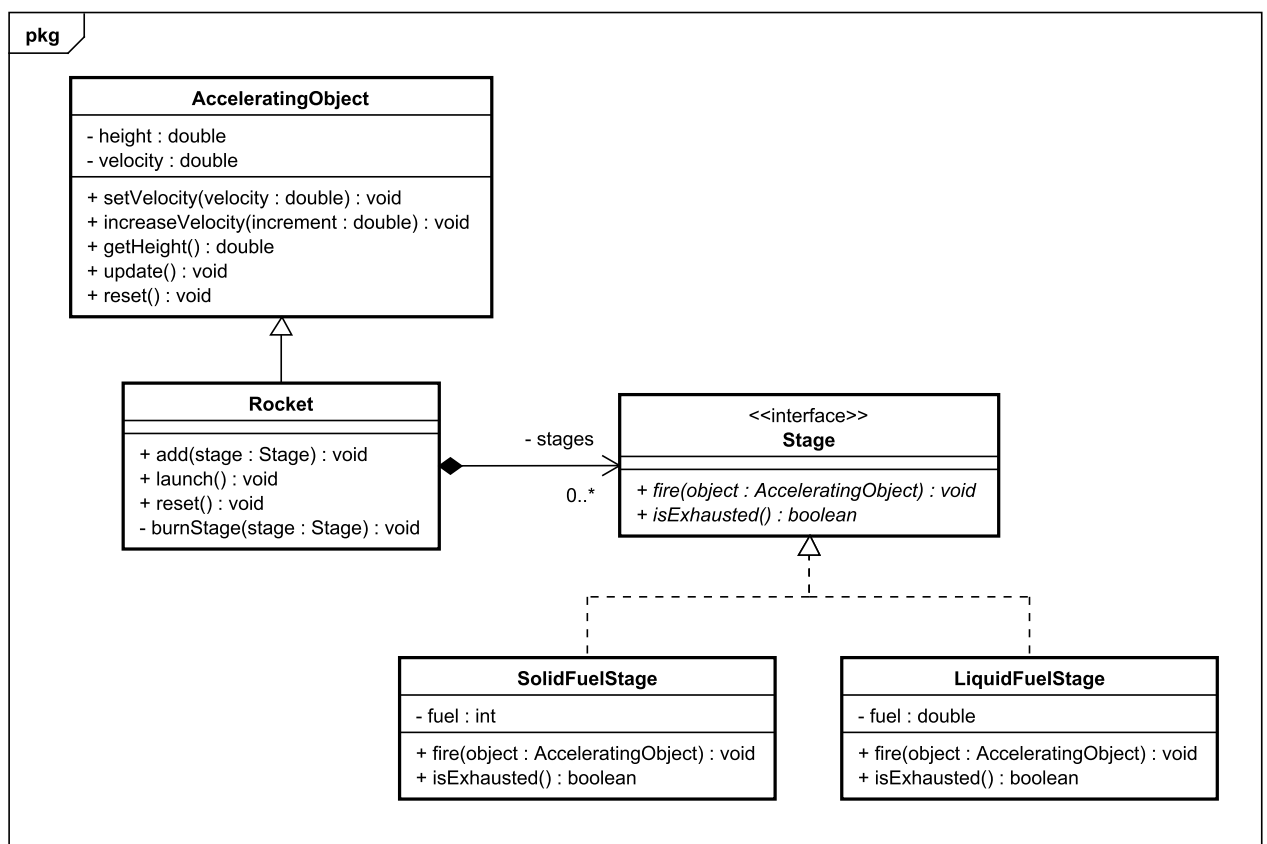


Elke student wordt geacht de bepalingen m.b.t. het afleggen van de toetsen te kennen.

|          |  |
|----------|--|
| Vak      | Introductie Software Development in Java |
| Subtitel | Deel A - Theorie                         |
| Vakcode  | EITI-ISDJ                                |
| Datum    | 21 jan 2022                              |
| Pagina   | 2 van 12                                 |

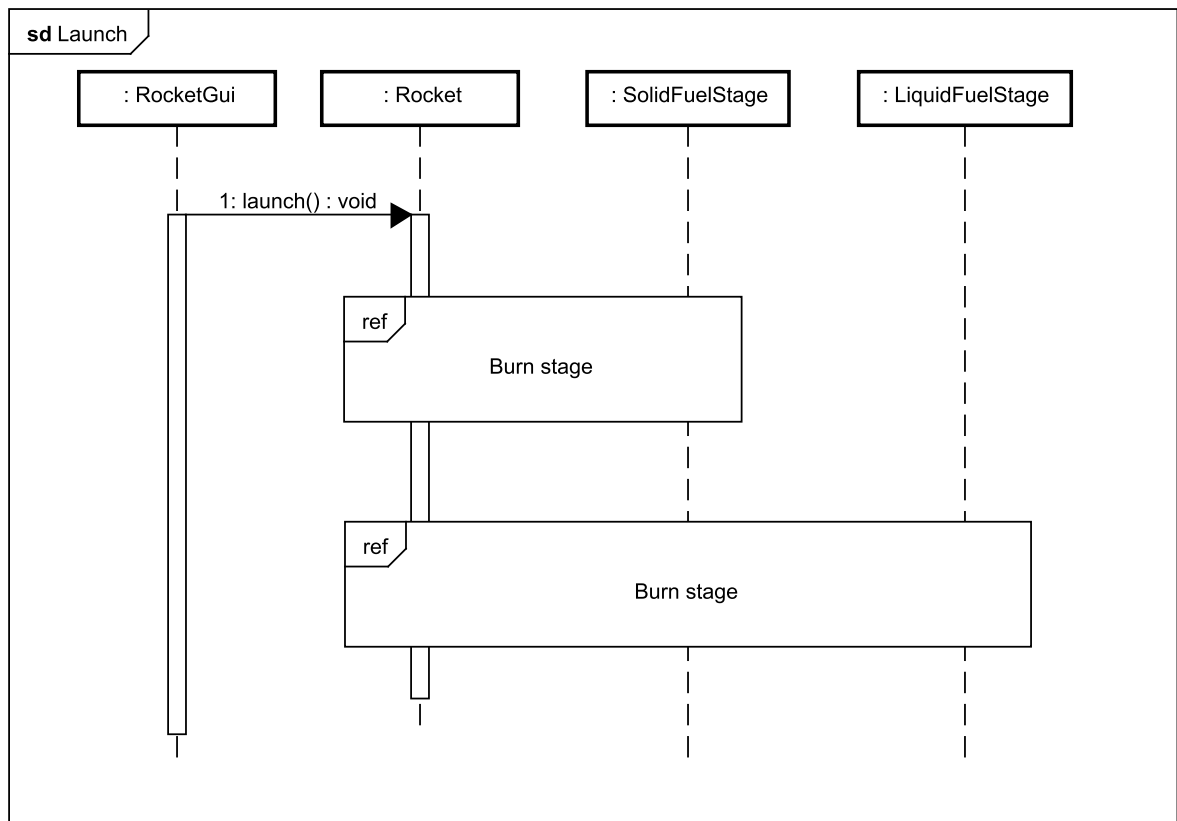
In onderstaand klassendiagram worden de klassen (en interfaces) van het RocketLaunch systeem getoond, met hun onderlinge relaties. Zoals te zien is in dit diagram bestaat de Rocket uit Stages, waarvan er twee soorten zijn. De Rocket is een AcceleratingObject en die klasse heeft zijn eigen attributen en methoden om de hoogte en de snelheid te bepalen.

De twee soorten Stages houden ieder hun hoeveelheid brandstof (fuel) bij. Elke stage vermindert tijdens de fire() methode op zijn eigen wijze de hoeveelheid fuel en verandert daarbij ook op zijn eigen wijze de snelheid en hoogte van het AcceleratingObject waarvan de stage deel uitmaakt.



|          |  |
|----------|--|
| Vak      | Introductie Software Development in Java |
| Subtitel | Deel A - Theorie                         |
| Vakcode  | EITI-ISDJ                                |
| Datum    | 21 jan 2022                              |
| Pagina   | 3 van 12                                 |

In onderstaand sequence diagram wordt getoond hoe de simulatie van een lancering wordt gestart. Via een Graphical User Interface klasse RocketGui krijgt de Rocket een launch bericht. In reactie daarop zal de Rocket na elkaar elk van zijn stages laten ontbranden. In dit specifieke geval is dat eerst een SolidFuelStage en daarna een LiquidFuelStage.



|          |  |
|----------|--|
| Vak      | Introductie Software Development in Java |
| Subtitel | Deel A - Theorie                         |
| Vakcode  | EITI-ISDJ                                |
| Datum    | 21 jan 2022                              |
| Pagina   | 4 van 12                                 |

## Opgaven deel A

1. Kijkend naar het klassendiagram van RocketLaunch (hierboven), wat voor relatie geeft de pijl tussen **SolidFuelStage** en **Stage** aan?

- 
- A. **SolidFuelStage** is afhankelijk van **Stage**
  - B. **SolidFuelStage** erft de methoden van **Stage**
  - C. **SolidFuelStage** implementeert de methoden van **Stage**
  - D. **SolidFuelStage** is onderdeel van **Stage**

2. Kijkend naar het klassendiagram van RocketLaunch (hierboven), wat voor geldige uitspraak kun je doen over klasse **Rocket**?

- 
- A. De Java code van klasse **Rocket** bevat een abstracte methode **burnStage**
  - B. De Java code van klasse **Rocket** heeft een private attribuut **stages**
  - C. De Java code van klasse **Rocket** heeft een private attribuut **velocity**
  - D. De Java code van klasse **Rocket** moet de methode **update** van **AcceleratingObject** implementeren

3. Kijkend naar het klassendiagram van RocketLaunch (hierboven), op welke plek(ken) in de Java code bevindt zich dan de implementatie (dus de ingevulde body) van methode **isExhausted**?

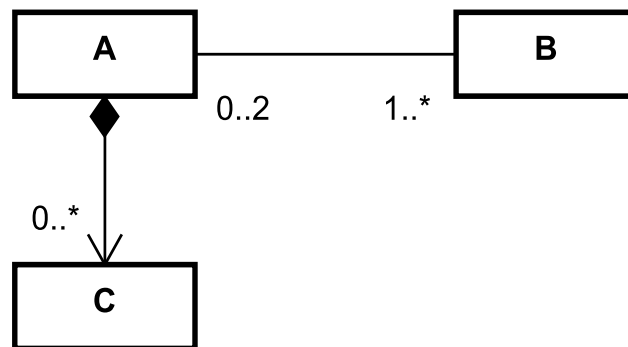
- 
- A. Alleen in **Stage**
  - B. Alleen in klasse **SolidFuelStage** en in klasse **LiquidFuelStage**
  - C. In **Stage**, in **SolidFuelStage** en in **LiquidFuelStage**
  - D. Naar keuze in ofwel **Stage**, ofwel **SolidFuelStage** ofwel **LiquidFuelStage** (dus één van deze drie)

4. Kijkend naar het sequencediagram Launch (hierboven), welke uitspraak is dan juist?

- 
- A. Klasse **Rocket** heeft een methode **launch**
  - B. Klasse **RocketGui** heeft een methode **launch**
  - C. Klasse **RocketGui** heeft een compositierelatie met klasse **Rocket**
  - D. Klasse **RocketGui** heeft een inheritance relatie met klasse **Rocket**

|          |  |
|----------|--|
| Vak      | Introductie Software Development in Java |
| Subtitel | Deel A - Theorie                         |
| Vakcode  | EITI-ISDJ                                |
| Datum    | 21 jan 2022                              |
| Pagina   | 5 van 12                                 |

5. Welke bewering over het klassendiagram met klassen A, B en C is waar?



- A. Een object van klasse A is gekoppeld aan 0 of 2 objecten van klasse B
- B. Een object van klasse A is gekoppeld aan 0, 1 of 2 objecten van klasse B
- C. Een object van klasse B is gekoppeld aan 0 of 2 objecten van klasse A
- D. Een object van klasse B is gekoppeld aan 0, 1 of 2 objecten van klasse A

6. Welke bewering over het klassendiagram hierboven met klassen A, B en C is waar?

- A. Een object van klasse B kan methoden van klasse A aanroepen
- B. Een object van klasse C kan methoden van klasse A aanroepen
- C. Een object van klasse B kan methoden van klasse C aanroepen
- D. Geen van de klassen A, B en C heeft methoden

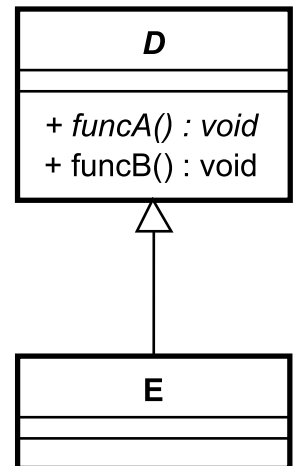
7. Welke bewering over het klassendiagram hierboven met klassen A, B en C is NIET waar?

- A. Een object van klasse C omvat nul of meer objecten van klasse A
- B. Een object van klasse A omvat nul of meer objecten van klasse C
- C. Een object van klasse C kan alleen bestaan als onderdeel van een object van klasse A
- D. Klasse A is verantwoordelijk voor het aanmaken van objecten van klasse C

|          |  |
|----------|--|
| Vak      | Introductie Software Development in Java |
| Subtitel | Deel A - Theorie                         |
| Vakcode  | EITI-ISDJ                                |
| Datum    | 21 jan 2022                              |
| Pagina   | 6 van 12                                 |

8. Welke bewering over klassen D en E is NIET waar?

- A. Klasse E implementeert funcA()
- B. Van klasse D kan geen object gemaakt worden
- C. Van klasse E kan geen object gemaakt worden
- D. Klasse E erft attributen van klasse D

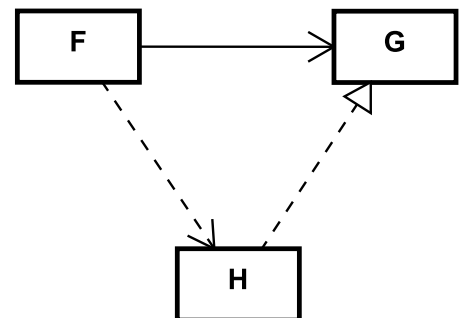


9. Welke bewering over klassen D en E is waar?

- A. Methode funcA() is abstract
- B. Methode funcB() is abstract
- C. Methoden funcA() en funcB() zijn beide abstract
- D. Methoden funcA() en funcB() zijn geen van beide abstract

10. Welke bewering over klassen F, G en H is waar?

- A. Klasse F heeft een dependency op klasse G
- B. Klasse F heeft een dependency op klasse H
- C. Klasse H heeft een dependency op klasse G
- D. Dit diagram bevat geen dependency relaties

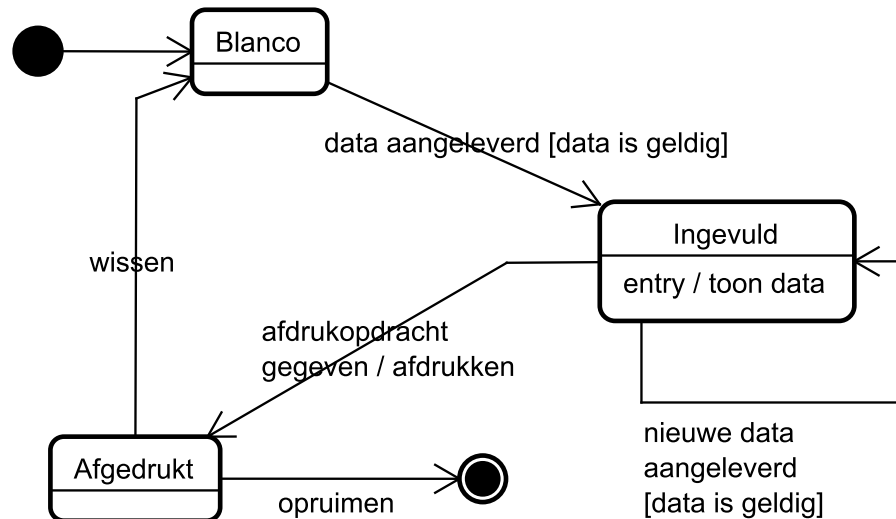


11. Welke bewering over sequence diagrammen is waar?

- A. Een sequence diagram toont berichten die worden uitgewisseld tussen klassen
- B. Een sequence diagram toont relaties tussen klassen
- C. Een sequence diagram toont berichten die worden uitgewisseld tussen objecten
- D. Een sequence diagram toont relaties tussen objecten

|          |  |
|----------|--|
| Vak      | Introductie Software Development in Java |
| Subtitel | Deel A - Theorie                         |
| Vakcode  | EITI-ISDJ                                |
| Datum    | 21 jan 2022                              |
| Pagina   | 7 van 12                                 |

12. Welke bewering over dit toestandsdiagram is waar?



- A. Als geldige data wordt aangeleverd wordt de 'toon data' actie uitgevoerd
- B. Een blanco object kan worden opgeruimd door het eerst te wissen
- C. Het object kan gewist worden zonder het eerst af te drukken
- D. Het object kan meermaals afgedrukt worden zonder het eerst te wissen

13. Welke informatie kun je direct afleiden uit een use-case diagram?

- A. Een use-case diagram laat zien hoe de user interface van een systeem eruit ziet
- B. Een use-case diagram laat zien welke functionaliteiten een systeem heeft vanuit het oogpunt van de gebruiker
- C. Een use-case diagram laat zien welke methoden de hoofdklasse van een systeem heeft
- D. Een use-case diagram laat zien welke stappen worden doorlopen in een gebruiksscenario van het systeem


14. Welk element komt NIET voor in activiteitsdiagrammen?

- A. Een beslispoint
- B. Een guard conditie
- C. Een merge point
- D. Een toestand



|          |  |
|----------|--|
| Vak      | Introductie Software Development in Java |
| Subtitel | Deel A - Theorie                         |
| Vakcode  | EITI-ISDJ                                |
| Datum    | 21 jan 2022                              |
| Pagina   | 8 van 12                                 |

15. Welke bewering over activiteitsdiagrammen en toestandsdiagrammen is waar?

- A. Zowel toestandsdiagrammen als activiteitsdiagrammen beschrijven het gedrag van één object (maar ze gebruiken verschillende notaties daarvoor)
- B. Toestandsdiagrammen hebben een eindpunt , activiteitsdiagrammen hebben dat niet
- C. Toestandsdiagrammen noteren toestanden in blokken, activiteitsdiagrammen noteren activiteiten juist met pijlen in plaats van in blokken
- D. Activiteitsdiagrammen gebruiken splitsingen (forks) en synchronisaties (joins) voor parallelle activiteiten, toestandsdiagrammen kennen geen notatie voor parallelle activiteiten

16. Welk van de volgende beweringen over Exceptions is waar?

- A. Een exception moet altijd opgevangen worden in een try...catch blok
- B. Een exception moet altijd in de methode afgevangen worden waar deze wordt opgegooid
- C. Een exception kan een bericht bevatten om aan te geven wat er mis is gegaan
- D. Een exception kan niet gebruikt worden in combinatie met een return-statement

17. Wat gebeurt er in de volgende code?

```
HashMap<String, String> map = new HashMap<>();
System.out.println(map.get("hallo"));
```

- A. Deze code compileert niet
- B. Deze code geeft een nullpointer exception
- C. Deze code print de tekst "null"
- D. Deze code geeft een KeyNotFoundException

|          |  |
|----------|--|
| Vak      | Introductie Software Development in Java |
| Subtitel | Deel A - Theorie                         |
| Vakcode  | EITI-ISDJ                                |
| Datum    | 21 jan 2022                              |
| Pagina   | 9 van 12                                 |

18. Wat is de uitvoer van de volgende code?

---

```
HashMap<String, String> map = new HashMap<>();
map.put("a", "a");
map.put("b", "b");
map.put("a", "b");
System.out.println(map.get("a"));
```

- A. Deze code geeft de uitvoer "a"
- B. Deze code geeft de uitvoer "b"
- C. Deze code geeft een fout bij het uitvoeren
- D. Deze code geeft geen uitvoer

19. Waarom compileert de volgende code niet?

---

```
class A {
    private int counter;
    public abstract void increase();
    public void decrease() {
        this.counter--;
    }
}
```

- A. Deze klasse is niet abstract
- B. De methode increase() heeft geen implementatie
- C. De methode decrease moet ook abstract zijn
- D. In een klasse met abstracte methoden moeten ook de attributen abstract zijn

20. Wat is het minimale dat je moet doen om een methode te overschrijven bij het overerven?

---

- A. Het keyword @Override boven de nieuwe methode zetten
- B. De nieuwe methode moet dezelfde naam, returntype en parameters hebben
- C. De nieuwe methode moet dezelfde naam hebben
- D. De nieuwe methode moet dezelfde naam en return type hebben, maar mag andere parameters hebben

|          |  |
|----------|--|
| Vak      | Introductie Software Development in Java |
| Subtitel | Deel A - Theorie                         |
| Vakcode  | EITI-ISDJ                                |
| Datum    | 21 jan 2022                              |
| Pagina   | 10 van 12                                |

21. Wat is de uitvoer van de volgende code?

---

```
class A {
    public void print() {
        System.out.print("A!");
    }
}
class B extends A {
    public void print() {
        System.out.print("B!");
    }
}
public static void main(String[] args) {
    A a = new B();
    a.print();
}
```

- A. A!
- B. B!
- C. A!B!
- D. B!A!

22. Wat is een risico van het gebruik van casting met klassen?

---

- A. Bij het gebruik van casting kunnen geen exceptions gebruikt worden
- B. Niet alle castings gaan goed, en dit kan exceptions geven
- C. Door te casten verlies je informatie
- D. Casten heeft geen risico's

23. Wat betekent het keyword 'protected'?

---

- A. Dit beschermt je code tegen exceptions
- B. Dit maakt een methode veilig om te gebruiken vanuit een andere klasse
- C. Dit geeft aan dat een attribuut of methode te gebruiken is vanuit een subklasse
- D. Dit zorgt dat een attribuut niet meer te veranderen is, het beschermt tegen veranderingen

|          |  |
|----------|--|
| Vak      | Introductie Software Development in Java |
| Subtitel | Deel A - Theorie                         |
| Vakcode  | EITI-ISDJ                                |
| Datum    | 21 jan 2022                              |
| Pagina   | 11 van 12                                |

24. Welk van de volgende beweringen is waar?

---

- A. In een abstracte klasse staan altijd abstracte methoden
- B. Een interface doet hetzelfde als een abstracte klasse
- C. Je kunt in een klasse precies van 1 abstracte klasse overerven en 1 interface implementeren
- D. Een abstracte klasse kan geen constructor bevatten

25. Wat is geen voordeel van het gebruik van generics?

---

- A. Je hoeft code niet voor ieder verschillend type te schrijven
- B. Je code wordt type-safe
- C. Je code wordt sneller
- D. Je code hoeft minder te casten door specifieke types te gebruiken

26. Welk van de volgende beweringen is niet waar?

---

- A. Door een interface als parameter mee te geven, is het mogelijk code in een parameter mee te geven
- B. Door een interface te gebruiken, kunnen verschillende klassen op dezelfde manier gebruikt worden
- C. Door een interface te gebruiken, kunnen klassen die deze interface implementeren code delen met elkaar
- D. Een interface kan gebruikt worden om afhankelijkheden tussen klassen te verminderen

27. Wat is een verschil tussen een HBox en een FlowPane

---

- A. De HBox en FlowPane kunnen niet evenveel componenten tonen
- B. De FlowPane kan componenten over meerdere regels zetten, de HBox niet
- C. De FlowPane zet componenten naast elkaar, de HBox niet
- D. De FlowPane kan marges tussen componenten instellen, de HBox niet

|          |  |
|----------|--|
| Vak      | Introductie Software Development in Java |
| Subtitel | Deel A - Theorie                         |
| Vakcode  | EITI-ISDJ                                |
| Datum    | 21 jan 2022                              |
| Pagina   | 12 van 12                                |

28. Welke bewering over layoutmanagers is waar?

---

- A. Een borderpane resized alle componenten naar de maximale grootte
- B. De cellen in een gridpane zijn allemaal even groot
- C. Een VBox zet alle componenten naast elkaar
- D. Een layoutmanager kan een andere layoutmanager bevatten

29. Wat is het verschil in functionaliteit tussen een lambda-methode en een anonieme inwendige klasse?

---

- A. Een lambda-methode kan ook attributen aanpassen, een anonieme inwendige klasse niet
- B. Een lambda-methode kan maar 1 regel zijn, een anonieme inwendige klasse niet
- C. Een lambda-methode kan alleen een interface met 1 methode implementeren, een anonieme inwendige klasse niet
- D. Er is geen verschil

30. Waarom gebruiken we in JavaFX een ObservableList in plaats van een ArrayList?

---

- A. Met een ObservableList kan de user interface automatisch updaten
- B. JavaFX kan niet werken met ArrayLists
- C. Een ObservableList is een JavaFX component die we op de GUI kunnen laten zien
- D. Door ObservableList te gebruiken wordt je code sneller in combinatie met JavaFX

31. Welke layoutmanager kun je het beste gebruiken voor een GUI met een menu links, statusbalk onderin en de content in het midden?

---

- A. HBox
- B. VBox
- C. BorderPane
- A. FlowPane