Examenproject Objectgericht Programmeren

2024-2025

Deze tekst beschrijft de opgave voor het project van de cursus Objectgericht Programmeren. Dit project geldt als basis voor het examen van deze cursus. Het project wordt uitgewerkt in dezelfde teams als de practica.

Het project betreft het bouwen van een deel van de kern van een role-playing game (RPG) in een Middeleeuwse setting. Het spel is slechts een weinig gebaseerd op bestaande spellen. Sommige aspecten uit de opgave komen dan ook niet overeen met bestaande spellen. Ze worden ingevoerd om de opgave zo gevarieerd mogelijk te maken. Je oplossing moet de regels volgen zoals ze in deze opgave beschreven staan.

Alle grafische aspecten en alle aspecten in verband met gebruikersinteractie worden buiten beschouwing gelaten. Het project beperkt zich tot het uitwerken van die elementen, die in de kern van het softwaresysteem terug te vinden zijn. Meerbepaald moet het gevraagde systeem ondersteuning bieden voor het beschrijven van helden, monsters, wapens, harnassen, ... In de opgave worden een aantal vereisten gesteld omtrent hoe deze elementen moeten worden ondersteund in het beoogde systeem. Niet alle details worden beschreven, sommige zaken worden pas duidelijk na het volledig doornemen van de opgave. We verwachten dat je zelf ook de nodige basisfunctionaliteit als getters, setters en checkers voorziet. Verder zijn er een aantal niet-functionele vereisten, die zich richten op de kwaliteit van het beoogde systeem. Zo worden op diverse punten vereisten gesteld op het vlak van aanpasbaarheid en herbruikbaarheid. Hou deze zaken in gedachten, het is niet de bedoeling deze uitgebreid in je project uit te werken. Beperk je tot de expliciet gevraagde zaken, maar zorg dat een eventuele uitbreiding makkelijk te realiseren is.

Opgave

1 Helden [*Heroes*]

In dit soort computerspellen staan helden centraal. Zij hebben eigenschappen die in de volgende paragrafen worden besproken.

1.1 Naam [*Name*]

Helden hebben steeds een naam. Deze bestaat enkel uit letters, afkappingstekens (’), dubbele punten (:) en spaties. Ze móeten beginnen met een hoofdletter. Momenteel mogen in een naam maximaal twee afkappingstekens voorkomen. In de toekomst is het echter mogelijk dat deze niet langer mogen voorkomen. Dubbele punten mogen enkel voorkomen in een naam indien ze gevolgd worden door een spatie. Momenteel is de naam “James o’Hara” een voorbeeld van een geldige naam. Deze functionaliteit dient *defensief* uitgewerkt te worden.

De set van toelaatbare tekens kan mogelijks op termijn verder worden uitgebreid met de tekens ’.’, ’&’, ’:’, ’"’, ’/’, ’-’, ’_’ en ’@’. (Dit is een voorbeeld van functionaliteit die nu nog niet dient uitgewerkt te worden, maar die mits weinig moeite aangepast moet kunnen worden.)

1.2 Raakpunten [*Hitpoints*]

Elke held heeft een beperkt aantal raakpunten. Raakpunten worden weergegeven door een natuurlijk getal. Bij de geboorte van een held worden het maximaal aantal raakpunten toegekend. Dit maximum is een niet-negatieve waarde die bovendien zowel kan verhogen als verlagen. Ook kan deze verschillen van held tot held. Indien een held niet aan het vechten is, dan moet zijn aantal raakpunten een priemgetal zijn. In het andere geval geldt deze beperking niet. Deze karakteristieken zullen in de toekomst niet veranderen. Werk deze functionaliteit *nominaal* uit.

1.3 Kracht [*Strength*]

Elke held heeft ook een indicatie van zijn (intrinsieke) kracht. Momenteel wordt dit aangegeven door een getal met 2 cijfers na de komma. In latere versies kan deze precisie mogelijks wijzigen. De intrinsieke kracht van een held kan tijdens het spel wel wijzigen, maar enkel door vermenigvuldigingen of delingen met gehele getallen. Werk dit *totaal* uit.

Bij het berekenen van de slagkracht van een held (dus bij het uitdelen van een slag) moet rekening gehouden worden met wapens. De slagkracht is de som van de huidige intrinsieke kracht van de held (zoals hierboven beschreven), en de schade van de wapens die hij in zijn handen heeft. Aangezien helden zeer getraind zijn in het gebruik van wapens, kunnen zij wapens in hun linker- en rechterhand gelijktijdig gebruiken. Wapens op andere ankerpunten of in een rugzak kunnen zij niet gebruiken bij het uitdelen van een slag en hebben dus geen invloed op de slagkracht.

1.4 Capaciteit [*Capacity*]

De intrinsieke kracht van een held bepaalt ook zijn maximale draagcapaciteit, i.e. het maximale gewicht van alle spullen die de held met zich mee kan dragen. De capaciteit van een held komt overeen met de intrinsieke kracht, vermenigvuldigd met 20kg. Op termijn is het mogelijk om hier een andere berekening voor te gebruiken, maar het resultaat zal steeds afhankelijk zijn van de intrinsieke kracht.

De uitwerking (*totaal*, *nominaal* of *defensief*) is vrij te kiezen. Hou in het achterhoofd dat ook andere objecten zoals harnassen en wapens een gewicht hebben, en dat een held niet méér zal kunnen meeslepen dan zijn maximale draagcapaciteit.

1.5 Beschermingsfactor [*Protection*]

Elke held heeft een beschermingfactor. Deze karakteristiek weerspiegelt hoe goed een held slagen kan ontwijken tijdens gevechten. De beschermingsfactor van een held zal steeds een strikt positief geheel getal zijn. In de huidige versie van het spel heeft elke held een standaard beschermingsfactor van 10. Later zal deze standaardwaarde mogelijk worden aangepast, maar zal steeds dezelfde zijn voor alle helden.

Net zoals bij de kracht van een held rekening moet worden gehouden met de toevoeging van wapens tijdens het vechten, moet hier rekening gehouden worden met harnassen. Bij het slaan zal de reële bescherming van een held berekend worden als de som van de standaard beschermingsfactor en de beschermingsfactor van het harnas dat hij mogelijks op zijn *lichaam* draagt.

1.6 Ankerpunten [*Anchors*]

Helden hebben een aantal *ankerpunten* waarmee ze allerlei spullen met zich mee kunnen dragen. In de huidige versie van het spel heeft elke held precies 5 ankerpunten: een *linkerhand*, een *rechterhand*, een *rug*, een *lichaam* en een *riem*. Het moet echter mogelijk zijn om helden op termijn uit te breiden met bijkomende ankerpunten die dan ook een specifieke naam zullen krijgen. Elk ankerpunt kan gebruikt worden om een wapen, een harnas of een rugzak mee te nemen. Aan een ankerpunt zullen nooit meerdere elementen kunnen bevestigd worden. Helden krijgen bij hun geboorte altijd een harnas aangetrokken, hiervoor wordt het ankerpunt *lichaam* gebruikt. Voor de geldbeurs wordt het ankerpunt *riem* gebruikt. Dit ankerpunt kan bovendien enkel daarvoor gebruikt worden. Een geldbeurs zal bij creatie steeds een willekeurig aantal dukaten bevatten en heeft een standaard capaciteit van 100 dukaten.

1.7 Slaan [*Hit*]

Helden vechten enkel tegen monsters, ze doen dit door herhaaldelijk om beurten te slaan. Het toebrengen van een slag verloopt als volgt:

1. Er wordt een random getal gegenereerd tussen 0 en 100.

2. Het resulterend getal wordt vergeleken met de beschermingsfactor van de tegenstander (inclusief zijn eventuele harnas). Als het resulterende getal groter is dan of gelijk aan de beschermingsfactor van de tegenstander, dan is er raak geslagen. In het andere geval heeft de slag geen impact.

3. In het geval van een rake slag worden de raakpunten van de tegenstander verminderd met de schade die de slag bij hem teweeg brengt. In de huidige versie van het spel bekomt men de schade die een slag veroorzaakt door de slagkracht (de intrinsieke kracht plus de waarde van de wapens die hij vast heeft) van de held te verminderen met 10, en vervolgens door 2 te delen waarbij cijfers na de komma worden weggelaten. Merk op dat deze mogelijk negatief kan worden. In dat geval zullen we deze negatieve waarde door 0 vervangen. De regels omtrent de schade die een rake slag veroorzaakt, moeten makkelijk kunnen aangepast worden, zonder dat dit een impact heeft op andere, bestaande of toekomstige delen van het softwaresysteem.

4. Als door het toebrengen van een slag, de aangebrachte schade groter is dan het aantal raakpunten van de tegenstander, dan zal de aanvallende held bij het toebrengen van de genadeslag enigszins automatisch genezen. De raakpunten van de held worden verhoogd met een percentage van het verschil tussen zijn huidige en zijn maximale aantal raakpunten. Het percentage wordt willekeurig bepaald.

1.8 Genezen [*Heal*]

Zoals hierboven reeds werd aangegeven, kunnen helden herstellen bij het toebrengen van de genadeslag. Concreet wordt bij het herstellen een willekeurig percentage berekend. De raakpunten van de held worden vervolgens verhoogd met het gegenereerde percentage van het verschil tussen de maximale raakpunten van de held en zijn huidige raakpunten.

1.9 Schattenrapen [*Collect treasures*]

Helden *kunnen* bij het toebrengen van de genadeslag aan een vijand diens bezittingen overnemen. Op dit vlak moeten alle mogelijke opties open zijn, gaande van het achterlaten van alle bezittingen van het stervend wezen tot het overnemen van al diens bezittingen. Je mag hier vrij een concreet algoritme voor uitwerken.

1.10 Constructor

Voorzie twee constructoren: een zo eenvoudig mogelijke en een meer uitgebreide constructor waarbij zeker één of meerdere objecten meegegeven worden die aan de ankerpunten komen te hangen.

2 Monsters [*Monsters*]

Monsters zijn de vijanden van helden. Ze zullen helden dan ook aanvallen. Monsters hebben weinig tot geen intelligentie. Ze zullen daarom ook onderling vechten. Een monster wordt gekenmerkt door volgende eigenschappen.

2.1 Naam [*Name*]

Monsters hebben steeds een naam. In de huidige versie van het spel kunnen in namen van monsters enkel letters, spaties en afkappingstekens voorkomen. De naam van een

monster moet beginnen met een hoofdletter. “Destroyer o’Hope” is een voorbeeld van een geldige naam voor een monster. Ook hier geldt dat in toekomstige versies van het spel enkel bijkomende tekens kunnen worden toegelaten.

2.2 Ankerpunten [*Anchors*]

Monsters hebben, net zoals helden, een aantal *ankerpunten* waarmee ze allerlei spullen met zich mee kunnen dragen. Het aantal ankerpunten voor monsters kan echter *wel* verschillen van monster tot monster. Monsters zonder enig ankerpunt behoren ook tot de mogelijkheden. Zoals bij helden, kan elk ankerpunt van een monster gebruikt worden om precies één item mee te nemen.

Bij de geboorte zal ieder monster potentieel voorzien worden van een item voor ieder van zijn ankerpunten. Deze elementen zullen willekeurig verspreid worden over de ankerpunten van het nieuwe monster. De draagcapaciteit van het nieuwe monster wordt zodanig gekozen, dat het minstens alle meegegeven spullen kan dragen.

De ankerpunten van monsters worden verder niet benoemd.

2.3 Schade [*Damage*]

Monsters gebruiken klauwen, tanden of andere lichaamsdelen om te vechten. Monsters zijn niet intelligent genoeg om wapens te hanteren. Elk monster wordt gekenmerkt door de schade die het kan aanrichten bij de tegenstander in een gevecht. Hiervoor gelden dezelfde bepalingen als voor de schade die wapens kunnen aanrichten bij tegenstanders.

2.4 Beschermingsfactor [*Protection*]

Monsters hebben een taaie, dikke of geschubde huid, die als een natuurlijk harnas fungeert. Monsters zijn niet intelligent genoeg om échte harnassen aan te trekken. Voor de beschermingsfactor inherent aan een monster gelden dezelfde bepalingen als voor de beschermingsfactor die een harnas biedt.

2.5 Raakpunten [*Hitpoints*]

Elk monster heeft, net zoals elke held, een beperkt aantal raakpunten. Hiervoor gelden dezelfde bepalingen als voor raakpunten van helden, met die uitzondering dat de raakpunten van monsters niet verhogen op het einde van een gevecht dat ze als winnaar afgesloten hebben. Ze kunnen dus niet genezen.

2.6 Slaan [*Hit*]

Monsters kunnen vechten tegen andere monsters en tegen helden. Ze doen dit door de tegenstander herhaaldelijk te slaan. Zoals voor helden, moet in de context van dit project enkel het toebrengen van één slag door een monster aan zijn tegenstander worden uitgewerkt. Het toebrengen van een slag door een monster verloopt als volgt:

1. Er wordt een random getal gegenereerd tussen 0 en 100. Indien dit getal lager ligt dan de huidige raakpunten van het monster, wordt met dit getal verder gewerkt. Anders wordt gewerkt met de huidige raakpunten van het monster.

2. Het resulterende getal wordt vergeleken met de totale beschermingsfactor van de tegenstander. Als het resulterende getal groter is dan of gelijk aan de beschermingsfactor van de tegenstander, dan is er raak geslagen. In het andere geval heeft de slag geen impact.

3. In het geval van een rake slag worden de raakpunten van de tegenstander verminderd met de schade die de slag bij hem teweeg brengt. Deze schade is eenvoudigweg de schade zoals hierboven beschreven, zonder toevoegingen van wapens ed.

4. Als door het toebrengen van een slag, de aangebrachte schade groter is dan het aantal raakpunten van de tegenstander, zal het monster bij het toebrengen van die genadeslag potentieel zijn eigen bezittingen uitbreiden met bezittingen van de stervende tegenstander. Dit wordt hieronder verder toegelicht. Merk op dat monsters nooit kunnen herstellen van opgelopen verwondingen. Alle wapens en harnassen die de verslagen tegenstander met zich mee droeg, én die achterblijven na het toebrengen van de genadeslag vernietigen zichzelf; rugzakken en geldbeugels die niet meegenomen worden, blijven achter op de grond. (Dus enkel de wapens en harnassen die een monster zelf niet meeneemt na een gevecht, worden vernietigd.)

2.7 Schattenrapen [*Collect treasures*]

Monsters kunnen geen inschatting maken van de waarde of het nut van diverse soorten dingen. Monsters voelen zich wel sterk aangetrokken tot alles wat blinkt. Niets sluit uit dat ze dingen die ze momenteel meesleuren wisselen voor dingen die gedragen werden door de stervende tengestander. Je mag hier vrij een implementatie voor voorzien.

3 Wapens [*Weapons*]

Helden kunnen wapens gebruiken om monsters mee af te slachten. Wapens hebben de volgende kenmerken.

3.1 Identificatie [*Identification*]

Elk wapen heeft een identificatienummer (een groot geheel getal (type `long` in Java)). Het identificatienummer van een wapen zal steeds een positief, even getal zijn, deelbaar door 3. Het nummer zal na de productie van het wapen niet meer veranderen. Het nummer van elk wapen is uniek. Er kunnen echter wel andere soorten items in omloop zijn, waarvan het identificatienummer gelijk is aan dat van een wapen.

Bij het aanmaken van een nieuw wapen zal het identificatienummer intern gegenereerd worden (het mag dus niet als argument aan de constructor worden meegegeven). Alle methodes in verband met identificatienummers voor wapens en andere spelelementen moeten *totaal* worden uitgewerkt.

3.2 Gewicht [*Weight*]

Elk wapen heeft een gewicht. Het gewicht van een wapen wordt bij het vervaardigen eens en voor altijd vastgelegd. Het spel zal nooit mogelijkheden bieden om het gewicht van wapens na creatie te veranderen. Uiteraard kan het gewicht van een wapen nooit negatief zijn. Alle methodes in verband met gewicht van wapens en andere spelelementen moeten *totaal* worden uitgewerkt.

3.3 Schade [*Damage*]

Elk wapen wordt gekenmerkt door de schade die het kan aanrichten aan de tegenstander in een gevecht. De schade die een wapen kan toebrengen kan variëren in de tijd. In de huidige versie van het spel wordt de schade voor alle wapens uitgedrukt als een geheel getal dat ligt tussen 1 en 100 en moet een veelvoud van 7 zijn. Hoe hoger de waarde, hoe meer schade het wapen kan toebrengen. Het moet mogelijk zijn om in volgende versies van het spel de hoogst mogelijk waarde voor de schade aan te passen, zonder dat dit een impact heeft op andere klassen. Een nieuwe hoogste grenswaarde zal steeds van toepassing zijn op alle wapens. Alle methodes in verband met de schade die wapens en andere spelelementen kunnen aanbrengen moeten *nominaal* worden uitgewerkt.

3.4 Waarde [*Value*]

Elk wapen heeft een waarde. De waarde van een wapen wordt uitgedrukt in een hoeveelheid dukaten. Het is niet de bedoeling dat in de toekomst andere eenheden worden gebruikt om de waarde van wapens of andere spelelementen uit te drukken. Uiteraard kan de waarde van een wapen nooit negatief zijn. De waarde van een wapen is afhankelijk van de schade die het kan aanrichten. Elke eenheid schade die het kan aanrichten, is in de huidige versie van het spel 2 dukaten waard. Zorg ervoor dat deze berekening makkelijk aangepast kan worden in een volgende versie van het spel. Het moet zelfs mogelijk zijn om de koppeling tussen de waarde van een wapen en de schade die het kan berokkenen te breken. In dat geval zal de waarde van het wapen bij constructie worden meegegeven. De waarde zal wel altijd een aantal dukaten zijn tussen 1 en 200. Werk deze functionaliteit zo uit dat toekomstige wijzigingen zo weinig mogelijk impact op de specifciatie van de (andere) klassen heeft.

3.5 Eigenaar [*Holder*]

Helden kunnen wapens vasthouden door ze te bevestigen aan een van hun ankerpunten. Een wapen kan in een beginstadium los voorkomen in het spel (het kan bijvoorbeeld ergens op de grond liggen waardoor het op dat moment geen eigenaar heeft). Zorg voor functionaliteit voor het oprapen en opnieuw laten vallen van wapens. Een wapen kan zoals eerder vermeld ook overgaan van de ene eigenaar naar een andere. Bij die overdracht zal het wapen aan één van de ankerpunten van het andere personage worden bevestigd, of in een van de rugzakken worden gestoken die dat personage met zich meedraagt.

Van elk wapen moet het mogelijk zijn om de eigenaar op te vragen die het momenteel vast heeft. Hou er rekening mee dat een wapen ook in een rugzak kan

zitten en dat we dan de eigenaar van de rugzak willen kennen.

4 Harnassen [*Armors*]

Harnassen zijn een mooi staaltje van Middeleeuwse technologie. Harnassen kunnen ervoor zorgen dat een held geen of minder schade oploopt in een gevecht. Harnassen hebben de volgende kenmerken.

4.1 Identificatie [*Identification*]

Elk harnas heeft een identificatie in de vorm van een groot geheel getal (type `long` in Java). Het identificatienummer van een harnas zal steeds een positief priemgetal zijn, dat na productie van het harnas niet meer kan veranderen. Bij het aanmaken van een nieuw harnas zal het identificatienummer door de gebruiker worden meegegeven. Deze nummers moeten uniek zijn binnen de set harnassen. Er kunnen wel andere items bestaan met een gelijk identificatienummer.

4.2 Gewicht [*Weight*]

Elk harnas heeft een gewicht. Voor het gewicht van harnassen gelden dezelfde bepalingen als voor het gewicht van wapens.

4.3 Beschermingsfactor [*Protection*]

Elk harnas heeft een maximale beschermingsfactor. Dit is een geheel getal dat steeds ligt tussen 1 en 100. In de huidige versie van het spel zal de maximale beschermingsfactor van een harnas niet meer veranderen na de productie ervan. Het moet mogelijk zijn om na verloop van tijd verschillende soorten harnassen in te voeren, die elk een eigen vaste bovengrens hebben voor de beschermingsfactor (de ondergrens voor de maximale beschermingsfactor zal steeds 1 zijn). Zo moet het bijvoorbeeld mogelijk zijn om na verloop van tijd onderscheid te maken tussen tinnen harnassen en bronzen harnassen, waarbij alle tinnen harnassen een maximale beschermingsfactor hebben van 70, en alle bronzen harnassen een maximale beschermingsfactor hebben van 90.

Harnassen beschikken naast een maximale beschermingsfactor ook over een feitelijke beschermingsfactor. De feitelijke beschermingsfactor kan variëren in de tijd onder andere door slijtage of schade aan het harnas. Verhogingen zijn ook niet uitgesloten, onder andere door herstellingen aan het harnas. De waarde van de feitelijke beschermingsfactor zal echter steeds tussen 0 en de maximale beschermingsfactor liggen en zal gebruikt worden om slagen af te weren.

4.4 Waarde [*Value*]

Elk harnas heeft een waarde, die net zoals voor wapens, wordt uitgedrukt in een hoeveelheid dukaten. De waarde van een harnas kan nooit negatief zijn. Doorheen het spel wordt de waarde van een harnas steeds bepaald door zijn hoogst mogelijke waarde (die onveranderlijk is en bepaald wordt bij productie van het harnas) vermenigvuldigd met de procentuele beschermingsfactor van het harnas. (Voorbeeld: een

harnas met een hoogst mogelijke waarde van 100 dukaten en een beschermingsfactor van 10 op een maximum van 20, heeft een huidige waarde van 50 dukaten). De hoogst mogelijk waarde van een harnas zal steeds een even positief getal zijn, niet groter dan 1000.

4.5 Eigenaar [*Holder*]

Harnassen kunnen, net zoals wapens, worden vastgehouden door een held of een monster. Hiervoor gelden dezelfde bepalingen als voor wapens. Helden hebben daarenboven de intelligentie om een harnas aan te trekken, waardoor ze beter beschermd worden in gevechten. Helden kunnen nooit meer dan één harnas aantrekken. Uiteraard kunnen helden het harnas dat ze aangetrokken hebben ook terug uittrekken. Helden kunnen ook beslissen om een ander harnas dat ze meesleuren of ergens vinden aan te trekken. Zij zijn handig genoeg om in één beweging hun huidig harnas uit te trekken en een ander harnas aan te trekken. Een aangetrokken harnas zal zich op het ankerpunt *body* bevinden. Uitgetrokken harnassen zullen aan een ander ankerpunt gehangen worden, maar ze zullen dan geen bescherming bieden.

Net als bij wapens, moet het mogelijk zijn om de eigenaar van een harnas op te vragen. Ook hier zou het kunnen dat een harnas zich in een rugzak bevindt. De eigenaar van de rugzak moet dan worden teruggegeven.

In tegenstelling tot helden, kunnen monsters geen harnassen aantrekken. Hun huid biedt voldoende bescherming. Om het spel speelbaar te houden, kunnen helden nooit meer dan 2 harnassen meesleuren (inclusief het eventuele aangetrokken harnas).

5 Rugzakken [*Backpacks*]

De helden van dit spel maken gebruik van rugzakken om op een handige manier hun bezittingen met zich mee te nemen. Een rugzak is dus iets waar men andere spullen kan instoppen, zoals wapens, harnassen en zelfs andere rugzakken. Rugzakken hebben een capaciteit en kunnen dus slechts een beperkt gewicht dragen. (Het volume van de inhoud wordt hier buiten beschouwing gelaten.)

5.1 Identificatie [*Identification*]

Elke rugzak heeft een unieke identificatie in de vorm van een groot geheel getal (type `long` in Java). Er zijn geen specifieke eisen omtrent de waarde van dit getal. Na de productie van de rugzak zal dit getal niet meer veranderen.

5.2 Inhoud [*Content*]

Een rugzak kan gebruikt worden om allerlei spullen zoals wapens, harnassen en zelfs andere rugzakken in op te bergen. Helden of monsters kunnen zich nooit in een rugzak bevinden. Alle methodes in verband met de inhoud van rugzakken en andere spelelementen moeten *defensief* worden uitgewerkt. Je mag daarbij zelf kiezen welke uitzonderingsklasse(n) je daarbij gebruikt.

Zorg voor functionaliteit zodat kan worden opgezocht of een element met een bepaald identificatienummer zich in de rugzak bevindt. Rugzakken kunnen meerdere

objecten bevatten met hetzelfde identificatienummer. De tijd om op te zoeken of een object in een rugzak zit moet lineair zijn met het aantal objecten met hetzelfde identificatienummer. Indien er dus slechts één object met hetzelfde nummer in de rugzak zit, moet deze opzoeking in bijna constante tijd gebeuren (onafhankelijk van het aantal objecten in de rugzak).

De basisfunctionaliteit om objecten toe te voegen en te verwijderen moet uiteraard ook worden uitgewerkt.

5.3 Gewicht [*Weight*]

Elke rugzak heeft ook een eigengewicht. Voor het gewicht van rugzakken gelden dezelfde bepalingen als voor het gewicht van wapens en harnassen. Naast het opvragen van het eigengewicht, moet het mogelijk zijn het totale gewicht van een rugzak op te vragen, dus inclusief het gewicht van alle dingen die opgeborgen zijn in de rugzak.

5.4 Waarde [*Value*]

Elke rugzak heeft een eigenwaarde. De totale waarde van een rugzak is steeds de som van de eigenwaarde van de rugzak en van de waarde van alle dingen die er in opgeborgen zijn. De eigenwaarde van een rugzak kan nooit negatief zijn en nooit groter dan 500 dukaten. De eigenwaarde van een rugzak kan in de loop van het spel veranderen.

5.5 Capaciteit [*Capacity*]

Elke rugzak heeft een capaciteit. De capaciteit van een rugzak weerspiegelt het maximale gewicht van alle dingen die zich in de betrokken rugzak kunnen bevinden. De capaciteit van een rugzak wordt bij de productie ervan eens en voor altijd vastgelegd. Het spel zal nooit mogelijkheden bieden om capaciteiten van rugzakken te veranderen. Hou er rekening mee dat het gewicht van sommige dingen in een rugzak kan veranderen, waardoor de capaciteit overschreden zou kunnen worden. Alle methodes in verband met de capaciteit van rugzakken en andere spelelementen moeten *totaal* worden uitgewerkt.

5.6 Eigenaar [*Holder*]

Zoals wapens en harnassen kunnen rugzakken worden vastgehouden door helden en monsters. Rugzakken kunnen ook worden opgeborgen in andere rugzakken. Rugzakken kunnen ten alle tijde los voorkomen in het spel (zij kunnen bijvoorbeeld ergens op de grond liggen). Eigenaars kunnen op eender welk moment sommige van de rugzakken die ze vasthouden, laten vallen. Die rugzakken kunnen dan later terug worden opgepikt.

6 Geldbeurzen [*Purses*]

Een geldbeurs dient om dukaten in op te bergen. Andere dingen kunnen niet worden weggeborgen in een geldbeurs. Geldbeurzen hebben volgende kenmerken:

6.1 Identificatie [*Identification*]

Elke geldbeurs heeft een identificatie in de vorm van een groot geheel getal (type `long` in Java). Het identificatienummer van een geldbeurs kan na de productie van die geldbeurs niet meer veranderen.

6.2 Inhoud [*Content*]

Een geldbeurs kan enkel gebruikt worden om een aantal dukaten in op te bergen. Wapens, harnassen, geldbeurzen, helden, monsters, ... kunnen niet worden opgeborgen in een geldbeurs. Zorg voor functionaliteit die toelaat om (de inhoud van) een geldbeurs toe te voegen tot een andere geldbeurs. In dat geval worden de dukaten overgeheveld, waarna de lege geldbeurs op de grond gegooid wordt.

6.3 Gewicht [*Weight*]

Elke geldbeurs heeft een eigengewicht. Voor het gewicht van geldbeurzen gelden dezelfde bepalingen als voor het gewicht van wapens, harnassen en rugzakken. Naast het opvragen van het eigengewicht, moet het mogelijk zijn het totale gewicht van een geldbeurs op te vragen. Dit is het eigengewicht plus de som van de gewichten van de dukaten in de beurs. Elk dukaat weegt 50 gram.

6.4 Waarde [*Value*]

Elke geldbeurs heeft een waarde. Deze waarde is steeds de som van de waarde van alle dukaten die er in opgeborgen zijn. (Er is dus geen eigenwaarde.)

6.5 Capaciteit [*Capacity*]

Elke geldbeurs heeft zijn eigen capaciteit. De capaciteit van een geldbeurs wordt uitgedrukt in het aantal dukaten dat maximaal kan opgeborgen worden in de betrokken geldbeurs. De capaciteit van een geldbeurs wordt bij de productie ervan ingesteld. Iedere poging om meer dukaten in een geldbeurs op te bergen dan aangegeven door de capaciteit, resulteert in het scheuren van die geldbeurs. Een gescheurde geldbeurs kan niet meer hersteld worden, en alle opgeborgen dukaten verdwijnen uit het spel. Het scheuren van een geldbeurs impliceert dat de geldbeurs waardeloos wordt. Niets verhindert dat helden en monsters gescheurde (en dus lege) geldbeurzen toch nog met zich meesleuren, hoewel het nut ervan erg klein zal zijn.

6.6 Eigenaar [*Holder*]

Zoals andere spelelementen, kunnen geldbeurzen worden vastgehouden door helden en monsters. Een geldbeurs kan ten alle tijde ook los voorkomen in het spel (ze kan bijvoorbeeld ergens op de grond liggen). Helden en monsters kunnen geldbeurzen ten alle tijde laten vallen. Ze kunnen dan later terug worden opgepikt door hetzelfde wezen of door een ander.

7 Relaties

Je dient minstens één bidirectionele relatie te implementeren. Herrinner dat het voor elk item mogelijk moet zijn om de eigenaar ervan op te vragen. Ingeval zo'n item in een rugzak zit, willen we dus de eigenaar van de rugzak terugkrijgen.

8 Hoofdprogramma

De werking van alle ontwikkelde klassen moet je demonstreren aan de hand van volgend voorbeeld:

1. Construeer een held, gebruik makend van één van de constructoren die je daarvoor gebouwd hebt. Schrijf de totale waarde van alle bezittingen van de nieuwe held uit op de standaard uitvoerstroom.

2. Construeer een monster, gebruik makend van één van de constructoren die je daarvoor gebouwd hebt. Schrijf de totale waarde van alle bezittingen van het nieuwe monster uit op de standaard uitvoerstroom.

3. Werk een gevecht uit tussen de held en het monster, waarbij ze mekaar om beurt proberen te slaan. Laat één van beiden willekeurig beginnen. Het gevecht wordt voortgezet tot één van beiden sterft. Ook het eventuele herstellen van opgelopen verwondingen en het schattenrapen moeten geactiveerd worden.

4. Schrijf na afloop van het gevecht uit wie de winnaar is, en wat de totale waarde is van al zijn bezittingen (na het schattenrapen).

Je mag altijd meer informatie uitschrijven op de standaarduitvoerstroom, mocht je dat nodig/interessant vinden. **In geen geval mogen er zich schrijfopdrachten bevinden buiten de `Main` klasse waarin dit hoofdprogramma zich bevindt.**

9 Testen

Schrijf voor de aangemaakte klassen `JUnit` testklassen die de nodige testen bevatten. We verwachten minstens een coherent stel testen voor:

- de meest uitgebreide constructor van een held
- de methode om te slaan
- het toevoegen van een object aan een rugzak
- een object wegnemen van een anker

We raden sterk aan om alle methodes consequent te testen. Uiteraard is het de bedoeling dat je testen foutloos kunnen worden uitgevoerd.

10 Praktische Richtlijnen

10.1 Ontwerptekening

Naast de specificatie en implementatie van de gevraagde klassen, moet ook een schets van het ontwerp worden gemaakt. Deze bevat alle ontwikkelde klassen, behalve de zelfgedefinieerde uitzonderingsklassen en de testklassen. Het ontwerp moet uitgetekend worden in UML. Lijst voor elke klasse de naam op, samen met alle velden en methoden die in de klasse aanwezig zijn.

10.2 Verdere richtlijnen rond de oplossing

- Enkel de expliciet opgelegde vereisten moeten voorzien worden. Het is niet de bedoeling om de bijkomende elementen uit de opgave te voorzien. Gebruik je tijd voor de essentiële zaken.
- Hoewel de opgave vrij uitgebreid is, hebben we niet gepoogd alle mogelijke details in te vullen. Alles wat niet uitdrukkelijk vermeld is in de opgave mag je dan ook naar eigen inspiratie invullen. Kies daarbij steeds voor de gemakkelijkste oplossing.
- Als opgelegd wordt dat methodes rond een bepaalde karakteristiek nominaal, totaal of defensief moeten worden uitgewerkt, dan geldt dit ook voor het manipuleren van die karakteristieken in meer complexe methodes (vb. constructoren). Als er niets wordt opgelegd, mag er vrij worden gekozen.
- Indien je ergens een uitzondering dient te gooien, en er kan geen zinnig gebruik gemaakt worden van een `IllegalArgumentException` of een `IllegalStateException`, dan dien je zelf een uitzonderingsklasse te definiëren.

11 Beperkte opgave voor kleinere teams

Teams die uit minder studenten bestaan dan tijdens de practica, dienen dit zo snel mogelijk te melden via email aan tommy.messelis@kuleuven.be en jan.goedgebeur @kuleuven.be, met vermelding van de reden. Studenten die dit vak voor de tweede keer opnemen maken ook het project individueel.

Alle teams die uit slechts 1 of 2 studenten bestaan mogen alle zaken in verband met geldbeurzen negeren. Studenten die individueel werken mogen daarnaast ook alle zaken in verband met harnassen negeren.

12 Generative AI

Het gebruik van generatieve AI systemen is niet toegelaten bij het uitwerken van jullie oplossing. Beperkte tools binnen je IDE (zoals code-completion) kunnen wel worden gebruikt.

13 Verplichte uitwerking

Het examenproject OGP is verplicht uit te werken: ieder team moet een oplossing voor de vooropgestelde datum inleveren! **Het tijdig en correct indienen van het project is een voorwaarde voor deelname aan het examen!** Het project moet in dezelfde groepjes gemaakt worden als de practica. Er mag altijd overlegd worden met andere studenten over mogelijke oplossingen. Uiteindelijk moet ieder groepje voor zichzelf beslissen hoe bepaalde aspecten best worden uitgewerkt.

13.1 Indienen van het project

Het project wordt ingediend via de opdracht op Toledo, **ten laatste op vrijdag 16 mei 2025, 23:59**. 1 indiening per team volstaat. Zorg ervoor dat de namen van de teamleden vermeld worden in de hoofding van de klasse(n) (als `@author`).

Je oplossing bestaat uit een .zip of .jar bestand dat alle broncode bevat, en een pdf met de ontwerptekening. Let erop dat je al je ontwikkelde klassen (ook de `JUnit` klassen) selecteert (Select the resources to export), en neem zeker de *.java* bestanden op in de .jar! (**Eclipse neemt deze default niet op in het JAR-bestand!**)

13.2 Examen

Het examenproject wordt op 5 van de 20 punten van het vak gequoteerd en dient als basis voor de vraagstelling op het examen zelf. De overige 15 punten worden bepaald **door je antwoorden op het examen zelf**. Een goed project leidt dus niet automatisch tot een goede score. Omgekeerd maakt een project van mindere kwaliteit het uiteraard wel moeilijker, doch niet onmogelijk, om een goed examen af te leggen.

13.3 Peer review en self assesment

Na afloop van het project zullen jullie opnieuw worden uitgenodigd om jezelf en je teamleden te evalueren. Verdere instructies volgen op dat moment.

13.4 Vragen

Vragen omtrent het project kan je steeds per email stellen aan tommy.messelis@kuleuven.be en jan.goedgebeur@kuleuven.be. Stuur je email steeds naar beide adressen.