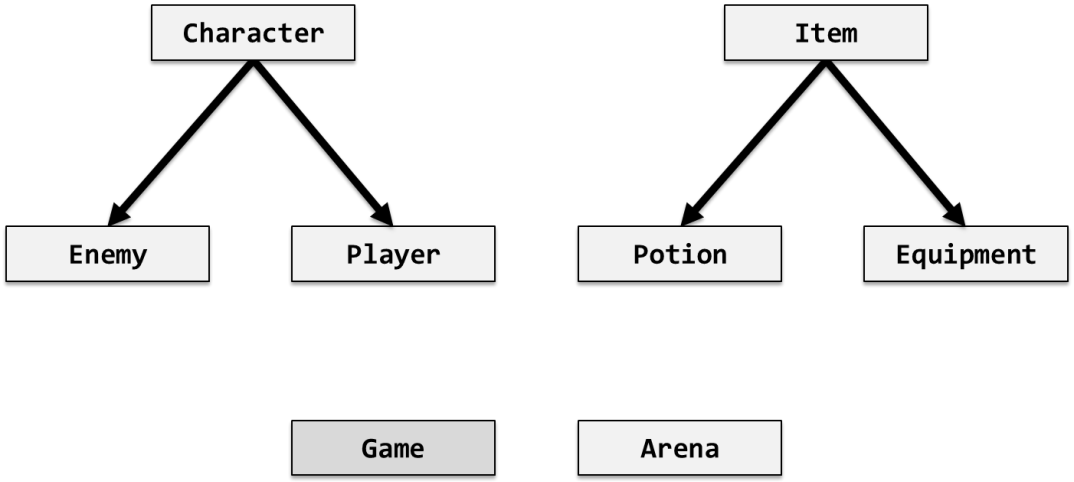
Java Oefeningen – Reeks 4

In deze reeks gaan we een tekstgebaseerde turn-based game maken. De speler vecht in een arena tegen een serie monsters, en moet zo lang mogelijk proberen te overleven.

In deze reeks gaan we gebruik maken van overerving (inheritance). Via het keyword **extends** kunnen klassen attributen en methoden overerven van andere klassen, en hierop voortbouwen. Het schema van de inheritance-relaties staat hieronder afgebeeld:



Oef 1

Maak eerst een klasse **Character**, met als attributen een **naam** en een **level.** Deze twee attributen worden meegegeven in de **constructor**. Een **Character** heeft meerdere stats, voorgesteld door de volgende **protected int**s: strength, defense, speed, huidig HP (health points) en maximum HP. Zet de stats initieel op een willekeurige waarde (maak gebruik van **Random**) tussen de onderstaande beschreven waarden i.f.v. het **level**:

strength, speed	$[8 \cdot level, 12 \cdot level]$
defense	$[4 \cdot level, 6 \cdot level]$
huidig hp = max hp	$[20 \cdot level, 30 \cdot level]$

Maak binnen **Character** de volgende methodes:

- **void plothp()**: print de naam van het Character uit, gevolgd door zijn HP. bv: `Pieter: 53 / 80 HP`
- **int attack(Character other)**: voert een aanval uit op de Character meegegeven als inputparameter (other). De toegebrachte schade is een willeurig getal tussen 0.8 en 1.2 maal de “strength”, min de “defense” van de tegenstander. De schade is altijd minstens 1. Deze geeft de toegebrachte schade terug als returnwaarde.
- **boolean turn(Character other)**: methode die een beurt uitvoert. Print uit hoeveel schade de ene Character de andere toebrengt (via **attack()**). Geeft een boolean terug indien de tegenstander al dan niet verslagen is (HP ≤ 0).
- Maak drie getters voor de attributen van **strength**, **defense** en **speed** (**get_strength()**, **get_defense()** en **get_speed()**)

Oef 2

Een **Enemy** is een uitbreinding van de klasse **Character**. Zijn constructor heeft enkel een **level**. De naam wordt vanzelf gegenereerd (kies willekeurig uit een lijst van minstens 5 namen). **Enemy** heeft ook een methode **expvalue()**, die het aantal "experience points" meegeeft dat de vijand waard is. Dit is een som van al zijn stats (strength, defense, speed en maximum HP).

Een **Player** is ook een uitbreiding van de klasse **Character**. De constructor heeft enkel een naam, het level wordt in het begin op 5 gezet. Een Player kan "experience points" hebben. Voorzie een methode die experience points toevoegt, en checkt of de speler een nieuw level kan bereiken. Indien het aantal exp-points groter is dan $N = 50 \cdot (level + 1)$, verhoog het level met 1, pas de stats aan zoals hierboven aangegeven en verminder de exp-points met N.

Oef 3

De gevechten vinden plaats in de **Arena**. Een Arena bevat twee **Characters**, en een **int speedcounter** die bepaalt wie aan de beurt is i.f.v. de respectievelijke snelheden van de Characters. Maak een methode **boolean fight()** die teruggeeft of Character 1 (true) of Character 2 (false) gewonnen heeft. Zet **speedcounter** initieel op het verschil tussen de twee Characters. Vervolgens voer je in een lus het volgende uit:

speedcounter > 0	Beurt aan speler 1. speedcounter -= snelheid van speler 2
speedcounter < 0	Beurt aan speler 2. speedcounter += snelheid van speler 1

Als speedcounter == 0, kies je willekeurig één van de twee gevallen uit. Plot bij elke beurt de HP's uit van de twee Characters d.m.v. **plothp()**.

We gaan ons spel uitvoeren de klasse **Game** (zie parallel-website).

Oef 4

Nu gaan we items in het spel introduceren. Maak een **abstract** klasse **Item** met twee **final** Strings (naam en beschrijving), welke worden geïnitialiseerd in de constructor. Maak hierin ook een **abstract** methode **void effect(Player)** die het effect van de item uitvoert.

Breid **Item** uit met de klasse **Potion**, met als constructor als inputs een naam en een **int** potency, die bepaalt hoeveel HP de **Potion** regenereert. Voorzie een standaardbeschrijving voor Potion, en geef die mee aan de **super**-constructor. De methode **effect** moet de meegegeven **Player** zijn HP incrementeren (maar niet boven zijn maximum HP). Print ook een bericht uit dat de speler met '**potency'** HP genezen werd.

Voorzie binnen **Player** een nieuw attribuut **inventory** van de klasse Map<Item, Integer>. Deze houdt bij hoeveel de speler van elk object bezit. Maak vervolgens drie bijkomende methodes in **Player**:

- **void additem(Item):** deze methode voegt een Item toe aan de **inventory** van een Player. Als de key voor dat Item binnen de Map al bestaat, incrementeer je de value (= de hoeveelheid) met 1. Anders maak je een nieuwe tupel aan voor dat Item en zet je de value op 1.
- **boolean useitem():** Print eerst alle Items en hun bijbehorende hoeveelheden af. Vervolgens kan de gebruiker de naam van het Item dat hij/zij wil gebruiken invoeren. Als de invoer overeenkomt met de naam van een Item uit het **inventory**, voer je het effect uit en decrementeer je de hoeveelheid van dat object met 1. Als de hoeveelheid gelijk wordt aan 0, haal je de entry uit de Map. Return **true** als het object gevonden werd, return **false** als het niet in de lijst stond.
- **boolean turn(Character):** Overschrijf de methode turn van **Character**. Laat de speler op zijn beurt kiezen tussen een aanval of een item gebruiken uit zijn **inventory** (via **useitem()**). Gebruik hiervoor de methode ***make_choice*** uit **Game**.

Oef 5

Voor het laatste gedeelte gaan we **Equipment** introduceren. Dit zijn objecten die je kunt dragen en je stats een boost geven. **Equipment** is ook een uitbreiding van Item, met als attributen strength, defense, speed, die aanduiden met hoeveel elk van de respectievelijke attributen van het **Character** zal geïncrementeerd worden. **Equipment** heeft ook een attribuut van het type **EquipType**, die 5 verschillende waarden kan aannemen: **HELMET, SHIELD, WEAPON, CHESTPLATE** en **BOOTS**.

Maak het attribuut **wearing** van type **List<Equipment>** in **Player** die de uitrusting voorstelt die de speller momenteel draagt. Overschrijf de getter voor strength in **Player**, en laat deze een nieuwe waarde teruggeven die de som is van de strength van de Player, en alle strength-attributen van alle **Equipment**-objecten in **wearing**. Doe hetzelfde voor de getters van defense en speed.

Implementeer tenslotte de methode **effect(Player)** van **Equipment**. Deze stopt de Equipment in **wearing** van de meegegeven **Player**. Er is echter wel een restrictie: een speler mag geen twee objecten van hetzelfde **EquipType** dragen. Controleer eerst of de speler al een object draagt van dat type, indien ja, haal je deze eerst uit **wearing** en stop je die terug in **inventory**, voor je de nieuwe Equipment gaat dragen.