Software Ontwikkeling
Labo 3: Inheritance & Polymorphism

Algemene afspraken labo's

a) op het einde van het labo worden de uitgewerkte oplossingen op de dropbox van minerva geplaatst.

b) de niet uitgewerkte oplossingen worden zelfstandig uitgewerkt tegen het volgende labo en ingediend op het einde van het volgende labo (tenzij anders vermeld wordt).

c) noteer ook telkens de tijd die je aan elke oefening gewerkt hebt in de code van de oplossing. Dit zal voor een klein deel bijdragen aan de finale punten.

Oefeningen

1. **Maak een lichtpunt (of hergebruik dit uit het vorige labo)**

   Het lichtpunt moet kunnen aan en uit geschakeld worden.
   Maak verschillende implementaties die extra functionaliteit toevoegen op de volgende manier:

   a. Dimbare lichten – kunnen ingesteld worden op een percentage (naast aan-uit)
      Maak verschillende implementaties van deze dimbare lichten die allemaal een verschillende uitwerking hebben wanneer die aangestuurd wordt (maak dit duidelijk in de boodschap die uitgeschreven wordt)
   b. RGB-leds waarbij je zowel kan dimmen als met sturing de kleur kan aanpassen
   c. Een verlichtingspunt dat op zijn beurt uit verschilende aparte lichtpunten bestaat en waarbij alle punten tegelijk of apart kunnen aangestuurd worden.

   Toon de functionaliteit van deze lampen aan met een korte boodschap die uitgeschreven wordt op de console wanneer de lamp wordt geschakeld of aangestuurd.
   Toon in de main-methode de functionaliteit van al deze lampen op de volgende manier:

   a. Maak alle lichtpunten in een aparte variabele aan
   b. Schakel rechtstreeks deze lichtpunten aan met alle specifieke methodes.
   c. Maak een array of lijst van lichtpunten aan en stop daar alle voorheen gemaakte lichtpunten in.
   d. Loop over deze lijst en schakel elk lichtpunt aan en uit
   e. Loop opnieuw over deze lijst en zet de kleur van alle RGB-leds op rood. Dit hoeft niet op de samengestelde lampen.

2. Binnenhuis-decoratie

Bij het uitwerken van een tekenprogramma voor binnenhuisdecoratie, worden de volgende concepten aangemaakt:

- Huis, Verdieping, Trap, Kamer, Keuken, Living, Slaapkamer, Traphal, hoge keukenkast, lage keukenkast, kookeiland, tafel, 3-zit, 2-zit, enkele zetel, bankstel, bed, kleerkast, televisie, tegels, houten vloer, tapijt en verlichting.
- Elke kamer heeft een kleur op de vier verschillende muren en plafond, en een vloertype.
- Elke keuken, living of slaapkamer houdt alleen maar deze meubels die daar logisch in passen. Het staat jullie vrij hier nog extra meubilair hier aan toe te voegen.
- Tapijt en stenen vloeren hebben een kleur
- Alle zetels hebben een type bekleding (string of enumeration)
- Alle meubels hebben een type hout (string of enumeration)

Maak een **structuurmodel** van de verschillende concepten in dit probleem. Zorg ervoor dat elk element zichzelf kan afprinten. Zorg er daarnaast voor dat er ook een methode is met de naam draw() die in staat zou kunnen zijn om alles uit te tekenen. Uittekenen zelf is niet nodig (en nogal complex), maar laat bij het aanroepen van deze methode een boodschap op het scherm verschijnen.

Let op: Deze opdracht wordt volledig op papier uitgevoerd en ingediend

3. Dobbelstenen

Schrijf een klasse Dice, die een dobbelsteen voorstelt. Het aantal zijden van de dobbelsteen is variabel en wordt meegegeven als parameter van de constructor. Er wordt ook een methode voorzien om met de dobbelsteen te gooien, die het aantal ogen als integer teruggeeft. Elk vlak evenveel kans om gekozen te worden (uniforme verdeling). De nummering van een vlak begint altijd bij 1 oog, en loopt zo op tot het aantal vlakken van de dobbelsteen.

Test deze Dice klasse uit vooraleer je verder gaat (schrijf bevoorbeeld code die een dobbelsteen maakt en er 10x mee werpt).

We voorzien nu ook een paar types dobbelstenen die een afwijkend gedrag hebben. Om het afwijkende gedrag te implementeren maken we gebruik van overerving.

> Hernoem de klasse Dice naar NormalDice en maak een nieuwe, **abstracte klasse Dice** met een **abstract methode** voor de worp. Zorg ervoor dat NormalDice van Dice overerft.
>
> OF
>
> Blijf werken met je originele Dice-klasse en leidt van deze klasse de 2 onderstaande klassen af.

De 2 types dobbelsteen met afwijkende gedrag zijn:

- Een magische dobbelsteen: het aantal vlakken is bij deze dobbelsteen niet constant. Als het maximum aantal ogen wordt geworpen verdubbelt het aantal vlakken voor de volgende drie worpen. Daarna gedraagt de dobbelsteen zich terug normaal.
- Een 'supermagische' dobbelsteen: deze dobbelsteen verdubbelt het aantal vlakken indient het huidige maximum aantal ogen wordt geworpen. Wanneer 1 oog gegooid wordt halveert het aantal vlakken. Het aantal vlakken wordt echter nooit minder dan 2.

Test ook deze klassen uit. Speel ook eens met het gebruik van polymorfisme (gebruik een Dice-referentie om naar instanties van eender welke dobbelsteen-klasse te verwijzen).

Om de 3 klassen makkelijker te kunnen gebruiken voorzien we nu ook een DiceFactory. Deze klasse bevat 1 methode, CreateDice, die een dobbelsteen teruggeeft. DiceType is een enumeratie die aangeeft wel soort dobbelsteen er wordt gevraagd.

```
Dice CreateDice(DiceType type, int sides)
```

Merk op: gelijk welke soort dobbelsteen er gevraagd wordt, er wordt altijd een Dice-referentie teruggegeven. Als de code die de dobbelsteen gebruikt een worp uitvoert, wordt aan de hand van polymorfisme de correcte implementie (van NormalDice, Dice, MagicDice of SuperMagicDice) opgeroepen.

Tot slot schrijven we een console applicatie om de DiceFactory uit te testen:

Als hoofdmenu:

1. Nieuw Spel.
2. Gooi.
3. Stoppen.

Bij de keuze “nieuw spel” krijg je een submenu:

1. Gewone dobbelsteen.
2. Magische dobbelsteen.
3. Super magische dobbelsteen.
4. Random dobbelsteen.

en vervolgens nog:

Geef aantal vlakken:

Bij de keuze Gooi krijg het je het aantal ogen van de worp te zien.

Werk bij het verwerken van de menukeuzes met Enumerations, zodat je bijna overal met de namen van de menu-opties kunt werken in je code in plaats van met de numerieke waarde.

Voor het implementeren van de Random dobbelsteen-keuze heb je de optie: ofwel kies je random een type dobbelsteen in de code voor de applicatie, ofwel geef je Random mee als type bij de methode CreateDice van de DiceFactory, waarbij de DiceFactory dan kiest welk soort dobbelsteen er gemaakt wordt.