Java Oefeningen – Reeks 3 (extra)

Vraag 4

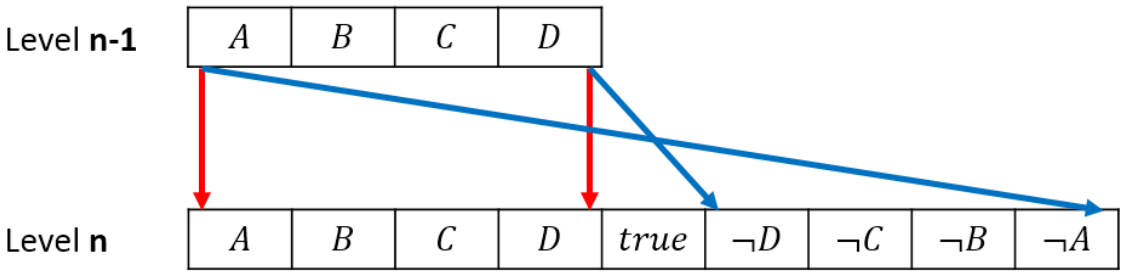
Voor het eerste deel van deze vraag is de opdracht om methodes te maken die een lijst van booleans teruggeven volgens een bepaald patroon. Maak de klasse **Drakencurve** (een **JPanel)**. Implementeer de volgende methodes in **Drakencurve**:

1. Schrijf een methode **List<Boolean> blockcurve(int n)** die een lijst van booleans teruggeeft van lengte n. Deze lijst bevat alternerend twee **trues** en twee **falses**:

$$\underbrace{\{\boldsymbol{true}, \boldsymbol{true}, \boldsymbol{false}, \boldsymbol{false}, \boldsymbol{true}, \boldsymbol{true}, \boldsymbol{false}, \boldsymbol{false}, \boldsymbol{true}, \boldsymbol{true}, \boldsymbol{false}, \ldots\}}_{\boldsymbol{n\ elementen}}$$

2. Maak vervolgens methode **List<Boolean> dragoncurve(int n)**. Deze methode gaat een lijst van booleans creëren volgens een specifiek algoritme (de betekenis hiervan wordt duidelijk in deelvraag 3). Om de lijst van niveau **n** op te bouwen onderneem je de volgende stappen:

    a. Neem de lijst van niveau **n-1** (niveau 0 is een lege lijst)
    b. Voeg een **true** toe aan de lijst
    c. Voeg een gespiegelde kopie toe van de lijst van niveau **n-1** toe, waarvan je van elk element de negatie neemt (**true** wordt **false** en vice versa).

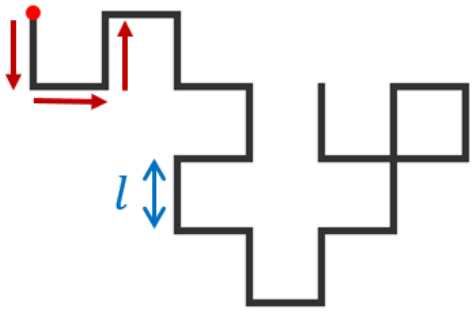


Hieronder een voorbeeld van de eerste 4 niveaus:

1. **true**
2. true **true** false
3. true true false **true** true false false
4. true true false true true false false **true** true true false false true false false

Tip: maak een procedure gegeven een booleanse lijst, een nieuwe lijst maakt van een hoger niveau, zoals aangegeven in Figuur 1.

De methodes uit deelvragen 1 en 2 gaan we gebruiken om een applicatie schrijven die “rechthoekige curves” kan tekenen. Deze curves bestaan uit een opeenvolging van lijnstukken van gelijke lengte, die allen met elkaar hoeken van 90° vormen. Een voorbeeld is hieronder getekend:

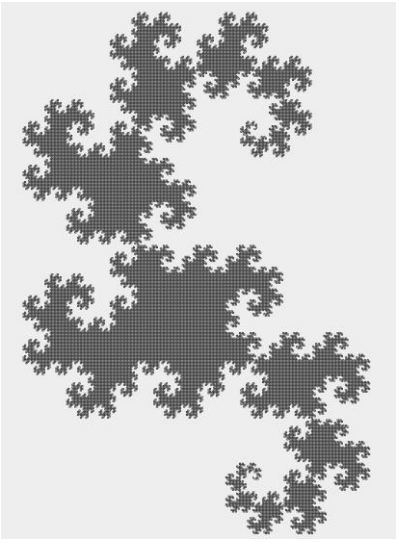


Figuur 1: voorbeeld van de rechthoekige curve

De curve begint in een gekozen punt (gericht naar beneden) met een lijnstuk van lengte l. Na elk lijnstuk slaagt de curve naar links of naar rechts af; dit wordt voorgesteld door een boolean (links = **`true`**, rechts = **`false`**). Een volledige curve kan dus beschreven worden als een lijst van booleans. Voor de bovenstaande curve komt dit overeen met:

```
{true, true, false, false, true, false, false, true, true,
false, true, true, false, true, false, true, true, true,
false, false}
```

3. Maak een methode **drawcurve(Graphics g, List<Boolean> curve, int x, int y, int l)**. Deze tekent een rechthoekige curve met als startpunt de coördinaten (**x**,**y**) en lijnstuk-lengte **l**. Gegeven een lijst **curve**, wordt de rechthoekige curve getekend met de corresponderende draaiingen op het Graphics-object **g**. Test de methode (apart) uit met lijsten gegenereerd in **blockcurve** en **dragoncurve:**
    - Voor **blockcurve:** gebruik $n = 127$, $(x, y) = (250{,}384)$ en $l = 8$.
    - Voor **dragoncurve:** gebruik $n = 16$, $(x, y) = (550{,}200)$ en $l = 2$.

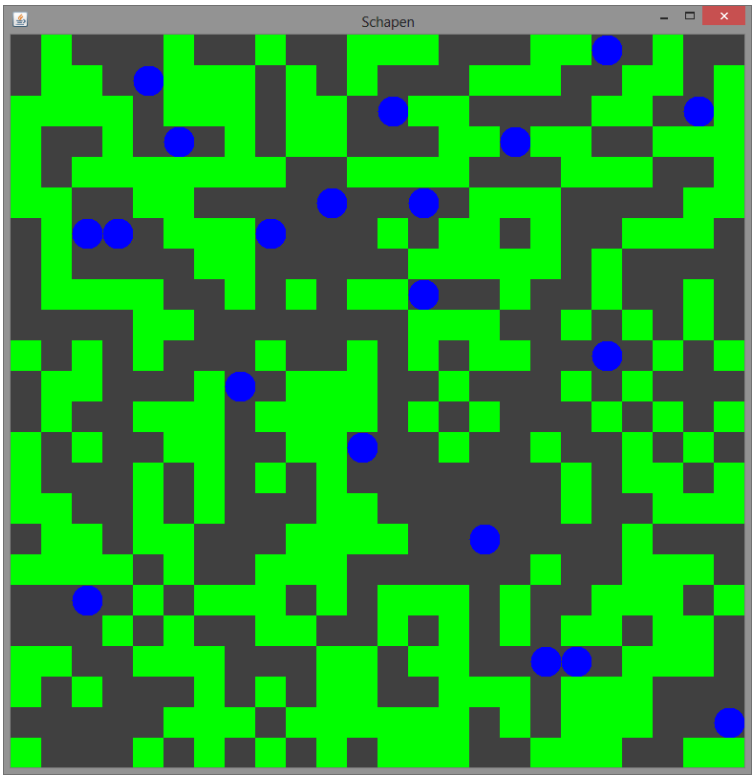


Figuur 3: Drakencurve van niveau 16

Figuur 4: Voorbeeld van een blockcurve

Vraag 5

In deze opgave gaan we een biotoop simuleren. De wereld wordt voorgesteld door een 24x24 grid; op elk vakje (van 32x32 pixels) bevindt er zich al dan geen gras. Op dat grid lopen schapen rond. Schapen zetten elke tijdseenheid een stap in een willekeurige richting. Schapen eten gras waardoor hun energiemeter verhoogt. Wanneer schapen genoeg energie hebben, reproduceren ze; maar schapen verbruiken ook energie, en als die op is sterven ze en worden ze dus verwijderd.



Voorstelling van het biotoop. Groene vakjes zijn begroeid met gras, grijze vakjes niet. Schapen worden voorgesteld als blauwe bolletjes.

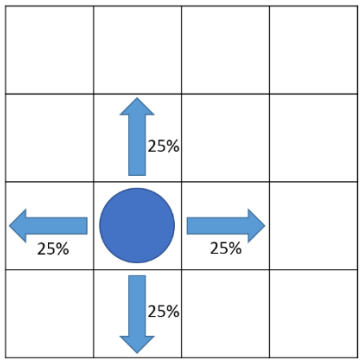
Maak de klasse **Simulation** (een **JPanel)**.

Elk grasveld heeft twee staten: gras of geen gras. Initialiseer het grasveld per vakje met 50% kans op gras. Na elke tijdseenheid is er 1% kans dat er op een leeg grasveld terug gras groeit.

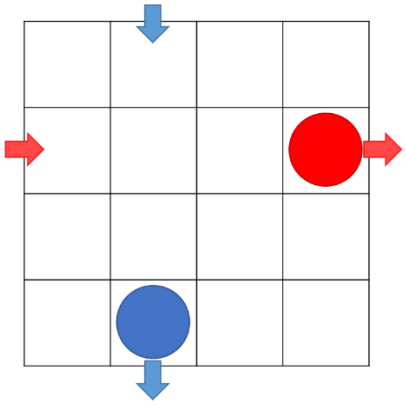
15 schapen worden willekeurig op het veld geplaatst. Schapen zetten elke tijdseenheid een stap in een willekeurige richting (links, boven, rechts of beneden; zie Figuur 2). Het veld is toroidaal: als ze over de rand stappen verschijnen ze aan de tegengestelde rand van het veld, zoals in Pacman (zie Figuur 3).

Schapen hebben een energiemeter die initieel op 10 staat. Deze neemt met 1 punt af per tijdseenheid. Bereikt deze 0, dan gaat het schaap dood en wordt het verwijderd. Als een schaap op een begroeid vakje komt, wordt het gras opgegeten en verandert het in een leeg vakje; het schaap krijgt dan 10 energiepunten bij. Wanneer een schaap meer dan 30 punten heeft, baart het een nieuw schaap op dezelfde positie als het oude schaap. Dit kost het oude schaap 20 punten.

Laat de simulatie draaien met een periode van 100 ms. Teken gras als gevulde rechthoekjes (groen voor begroeid, donkergrijs voor onbegroeid) en schapen als gevulde blauwe cirkels.



Bewegingsmogelijkheden van een schaap met bijgeschreven kansen.



Verplaatsing aan de randen van een torodoidaal veld.