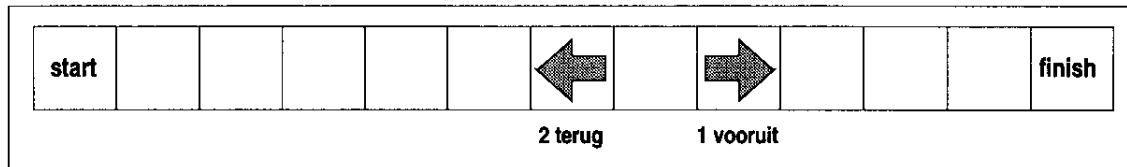


TENTAMEN  
**Imperatief Programmeren**  
19 april 2006

**Opgave 1**



In deze opgave bekijken we een versimpelde versie van het aloude *Ganzebordspel*. Een speler begint het spel met het plaatsen van zijn pion op het vakje *start*. Een beurt bestaat uit het werpen met één dobbelsteen. Het aantal ogen geeft het aantal plaatsen dat richting het vakje *finish* mag worden opgeschoven. Komt de pion uit op een vakje met een pijl, dan moet de pion terug of vooruit geplaatst worden, overeenkomstig de vermelde instructie op het bord.

Komt de pion uit op het vakje *finish*, dan is het spel uitgespeeld.

Komt de pion op het vakje *finish*, maar is er nog niet overeenkomstig het aantal ogen van de worp verplaatst, dan moet er worden doorgeteld. De pion wordt het resterende aantal plaatsen naar links gezet. Bijvoorbeeld: als de pion op het vakje voor de *finish* staat en er wordt een 6 gegooid, dan komt de pion te staan op het vakje tussen de twee pijlen.

De bedoeling is een programma te schrijven dat een schatting maakt van de verwachtingswaarde van het aantal beurten dat nodig is om het spel te spelen.

- [3 pt]  1. Schrijf een (aanzet voor) een klasse `MijnGans` die de gebruiker vraagt hoeveel spelletjes er gesimuleerd moeten worden.
  
- [6 pt]  2. Voorzie de klasse `MijnGans` van een (statische) methode `beurtCnt` die één spel simuleert en oplevert in hoeveel beurten de *finish* is bereikt.  
NB: de klasse `Math` heeft een methode `random` die een willekeurige waarde  $x$  oplevert die voldoet aan  $0 \leq x < 1$ . Gebruik dit om het werpen met de dobbelsteen te simuleren.
  
- [6 pt]  3. Breid de klasse uit met statements zodat het opgegeven aantal spelletjes wordt gesimuleerd en het gemiddelde aantal beurten per spel wordt berekend.

lees verder

## Opgave 2 Koekalender

Deze opgave gaat over een koekalender. Het is voor een melkveehouder uitermate belangrijk om van elke koe te weten in welk stadium van de lactatieperiode zij zich bevindt. Onder de lactatieperiode verstaan we hier de periode tussen twee opeenvolgende momenten waarop een kalf wordt geboren. Nadat een koe een kalf heeft gekregen begint ze met het produceren van melk. Met tussenpozen van ongeveer drie weken wordt ze tochtig. Op zo'n moment kan ze worden geïnsemineerd. Na enige tijd moet dan worden gecontroleerd of ze inderdaad drachtig is. Is dat daadwerkelijk het geval, dan kan de verwachte afkalfdatum worden berekend. Enkele weken voor de verwachte afkalfdatum wordt ze niet meer gemolken (ze wordt drooggezet).

Bij de administratie van dit hele proces spelen kalenderdata een belangrijke rol. Daarom richten we eerst de aandacht op een klasse om kalenderdata te representeren:

```
class KalenderDatum {
    int dag; // nummer van de dag
    int maand; // nummer van de maand: 1 = januari,..., 12 = december
    int jaar; // jaartal, voluit

    KalenderDatum () {
        dag = 0; maand = 0; jaar = 0;
    } // constructor

    KalenderDatum (int a, int b, int c) {
        dag = a; maand = b; jaar = c;
    } // constructor

} // KalenderDatum
```

- [5 pt]  4. Voorzie de klasse `KalenderDatum` van een methode `toString` die de datum oplevert in de vorm `dd-mm-jjjj`, dus twee cijfers voor de dag, twee cijfers voor de maand en vier cijfers voor het jaartal, gescheiden door `--`-tekens. De datum 31 mei 1998 wordt dus gerepresenteerd door de string `31-05-1998`.
- [5 pt]  5. Het komt vaak voor dat kalenderdata met elkaar moeten worden vergeleken. Voorzie de klasse `KalenderDatum` daarom van een methode `compareTo` die voldoet aan de specificatie:

```
int compareTo (KalenderDatum other) {
    /* resultaat is
     * -1 als this in de tijd komt voor other
     * 0 als this en other dezelfde datum zijn
     * +1 als this in de tijd komt na other
     */

} // compareTo
```

lees verder

- [8 pt] □ 6. In de koekalenderadministratie moet steeds vanuit een bekende datum een toekomstige datum worden bepaald: vanuit de afkalfdatum de verwachte tochtigheidsdatum; vanuit de inseminatiedatum de datum waarop de koe droog gezet moet worden etc. Implementeer in de klasse `KalenderDatum` de methode

```
KalenderDatum vooruit (int d) {
    /* precondition: d >= 0
     * resultaat is de datum die hoort bij de
     * dag die d dagen verder ligt dan de
     * dag die door this wordt gerepresenteerd
     */

    } // vooruit
```

Bijvoorbeeld: als `KalenderDatum` vandaag de datum 25 augustus 2004 representeert, dan moet de toekenning

```
KalenderDatum overEenWeek = vandaag.vooruit(7);
```

tot gevolg hebben dat `overEenWeek` de datum 1 september 2004 representeert.

NB: om dit te kunnen doen zul je in de klasse gegevens moeten opnemen over hoeveel dagen er in een maand zitten. Leg uit hoe je dit doet. Je hoeft geen rekening te houden met schrikkeljaren.

Dan nu twee klassen waarmee lactatieperioden en koeien worden gerepresenteerd:

```
class Lactatie {
    KalenderDatum afgekalfd =
        new KalenderDatum ();
    KalenderDatum inseminatie =
        new KalenderDatum ();
    String stierNummer;
} // Lactatie
```

```
class Koe {
    String naam;
    String nummer; // 12 cijfers
    Lactatie [] lact = new Lactatie [25];
    int lactNummer;

    Koe (String nm, String levnum) {
        naam = nm;
        nummer = levnum;
        lactNummer = 0;
    } // constructor

} // Koe
```

Bij een koe worden alle afgesloten lactaties en de actuele lactatie geadministreerd. De variabele `lactNummer` geeft het nummer van de actuele lactatie aan. Voor koeien die voor de eerste keer zijn geïnsemineerd is dit nummer 0; in de lactatie met nummer 0 is geen afkalfdatum geadministreerd. Voor alle lactaties geldt: als de koe nog niet is geïnsemineerd, dan representeert `inseminatie` de 'datum' 00-00-0000. Heeft er wel inseminatie plaatsgevonden, dan registreert `stierNummer` het nummer van de stier waarvan het sperma is gebruikt.

lees verder

- [5 pt]  7. Als een koe afkalft, dan begint er een nieuwe lactatieperiode. Voorzie de klasse Koe van de methode `afkalven` die als parameter een kalenderdatum krijgt (die de datum waarop het kalf is geboren representeert) en die dit gegeven verwerkt.

Neem nu aan dat er een lijst met koeien is:

```
Koe [] koe;  
int koeCnt;
```

De variabele `koeCnt` geeft aan van hoeveel koeien de gegevens zijn geadministreerd. Deze gegevens zijn opgenomen in de eerste `koeCnt` elementen van het array `koe`.

- [7 pt]  8. Geregeld zal de melkveehouder een lijstje willen hebben waarop staat welke koeien hij/zij moet controleren op drachtigheid. Dat gebeurt doorgaans drie weken na de inseminatie. Implementeer de methode

```
void drachtigheidsControleLijst (KalenderDatum vandaag) {  
    /* drukt een lijst af van namen, koenummers en  
     * inseminatiedata van alle koeien die  
     * tussen 23 dagen en 19 dagen voor vandaag zijn  
     * geïnsemineerd  
     */  
  
} // drachtigheidsControleLijst
```

- [7 pt]  9. Het is interessant om te weten hoe de leeftijdsopbouw van de veestapel is. Gevraagd wordt om een inventarisatie te maken van hoeveel koeien zich in een bepaalde lactatie bevinden. Geef een methode die een overzicht hiervan afdruckt. Bijvoorbeeld:

lactNr	aantal koeien
0	9
1	16
2	24
3	33
4	29
5	19
6	18

einde

Maximaal aantal punten: 52