3e deeltentamen Imperatief Programmeren

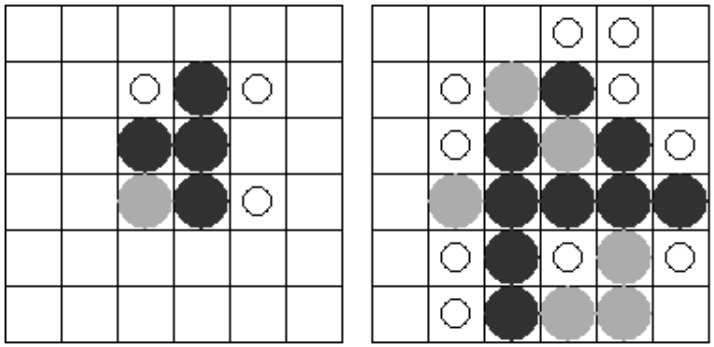
30 oktober 2009, 15.30-17.30 uur

Opgave 1

Bij het spel “reversi” leggen twee spelers om de beurt een gekleurde steen op een veld van een rechthoekig speelbord.
Een steen mag alleen maar worden neergelegd op een veld als

- het veld nog leeg is, en
- met deze zet een rij van een of meer stenen van de andere kleur wordt ingesloten tussen de nieuwe steen en een al op het bord liggende steen van de eigen kleur.

De ingesloten stenen kunnen in acht mogelijke richtingen naast de nieuwe steen liggen: horizontaal, verticaal of diagonaal. Stenen insluiten in meerdere richtingen mag ook.
In de figuur is voor twee voorbeelden met open cirkels aangegeven op welke velden de speler met de lichte stenen mag zetten.



Als gevolg van een zet veranderen alle ingesloten stenen van kleur.

In een programma wordt de situatie opgeslagen in een twee-dimensionale array:

```
int bord[][] = new int[6][6];
```

Lege velden zijn gecodeerd met 0, gevulde velden met 1 of -1 voor de twee kleuren.

De opgave: Schrijf een methode

```
boolean mag (int kleur, int x, int y)
```

die controleert of speler `kleur` een steen op veld `(x,y)` mag zetten.
(De zet wordt dus nog niet uitgevoerd!).
Hint: het is toegestaan (en handig) om een extra methode te schrijven die het insluiten in één van de 8 richtingen controleert.

Opgave 2

Van onderstaande zes onderdelen mag je er één overslaan.
De andere vijf tellen elk voor 2 punten.
Maak je ze toch alle zes, dan tellen ze elk voor 1.67 punten
(als je er eentje fout beantwoordt, had je hem dus beter kunnen overslaan!)

a. De Java GUI-library ‘Swing’ is een *lichtgewicht* GUI-toolkit.
- Wat is het verschil met een zwaargewicht toolkit, zoals AWT?
- Noem een voordeel en een nadeel van zo’n lichtgewicht toolkit.

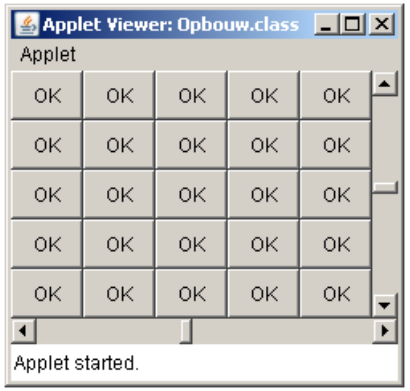
b. Een object is een groepje variabelen dat door methoden onder handen genomen wordt. Als het type van een object een klasse uit een standaard-package is, is het voor de programmeur niet altijd bekend uit welke variabelen zo’n object bestaat, maar uit het gedrag dat het object vertoont kun je het soms toch afleiden.
Geef twee voorbeelden van een variabelen die blijkbaar onderdeel uitmaken van een `Graphics`-object.
Geef ook twee voorbeelden van eigenschappen die in een `Graphics`-object *niet*, maar in een `Graphics2D`-object *wel* worden bijgehouden.

c. Beschrijf (kort) de syntax en de semantiek van de `switch`-opdracht.

d. Java heeft (sinds versie 5) een mechanisme voor “automatische boxing en unboxing”. Wat is dat, en wanneer wordt dat gedaan?

e. Wat wordt er in de standaardklasse `MouseAdapter` gedefinieerd?
Hoe kun je deze klasse in een programma gebruiken?

f. Hoe kun je de userinterface in de afbeelding hiernaast opbouwen?
(Je hoeft dit niet helemaal uit te programmeren; beschrijf in woorden welke objecten er nodig zijn om deze schermopbouw te realiseren).



In alle opgaven mag je de import-regels weglaten.
Je hoeft ook geen commentaar te schrijven.

Opgave 3

In deze opgave is het *niet* toegestaan om klassen uit package `java.util` (zoals ArrayList, LinkedList en andere Collections) te gebruiken; we gaan namelijk zelf iets dergelijks maken. Je mag wel gewone arrays gebruiken.

Een `WoordenLijst` is een lijst met nul of meer woorden; elk woord is een String.
Met zo'n woordenlijst willen we twee dingen kunnen doen:

- een woord toevoegen
- kijken of een bepaald woord al in de woordenlijst voorkomt

Aan een woordenlijst moet in principe een onbeperkte hoeveelheid woorden kunnen worden toegevoegd. Hetzelfde woord mag meermalen worden toegevoegd.

a. Specificeer deze mogelijkheden van een `WoordenLijst` door middel van een Java-taalconstructie.

We gaan twee verschillende implementaties hiervan maken. Allebei gebruiken ze een array van strings om de woorden in op te slaan. Aan het begin is er in die array ruimte om 10 strings op te slaan. Na 10 keer toevoegen van een woord is de array dus vol. Als er daarna toch een woord wordt toegevoegd moet een nieuwe array worden gemaakt met dubbele capaciteit, waarin de oorspronkelijke strings worden gekopieerd. Als die een tijdje later ook vol is, wordt de capaciteit weer verdubbeld, enz.

b. Zorg ervoor dat het aanmaken en het vergroten van de capaciteit van de arrays maar één keer hoeft te worden opgeschreven, als we straks in opgave c en d de implementaties gaan maken. Gebruik de taalconstructie die hiervoor bedoeld is.

c. Maak een implementatie `BinnenLijst`, die de woorden in volgorde van binnenkomst opslaat. Extra woorden kunnen dus op de eerste vrije plaats worden opgeslagen, en om te kijken of een woord er al is kun je de hele lijst langslopen.

Je kunt twee strings `s` en `t` vergelijken met `s.compareTo(t)`.
Als `s` en `t` gelijk zijn, is de uitkomst 0.
Als `s` alfabetisch voor `t` komt, is de uitkomst negatief, anders positief.

d. Maak een implementatie `SorteerLijst`, die de woorden in alfabetische volgorde opslaat. Bij het toevoegen van woorden moet dus eerst ruimte worden gemaakt op de goede plaats. De methode die kijkt of een woord er al is moet eerder stoppen met zoeken als dat mogelijk is vanwege het op-volgorde-staan.

Stel dat we in een `WoordenLijst` nu ook nog naar het langste woord moeten kunnen zoeken. Dat kan op twee manieren:

(1) Door, op het moment dat iemand om het langste woord vraagt, alle woorden langs te lopen en de langste te zoeken;
(2) Door voortdurend het langste woord te onthouden, en bij het toevoegen van een woord dit zo nodig aan te passen.

e. Maak een subklasse van `SorteerLijst`, die een extra methode `langste` implementeert op manier (2).

Opgave 4

Schrijf een programma, zonder visuele interface, met de volgende specificatie.
Het programma wordt door de gebruiker opgestart vanaf een commandoregel.
De gebruiker specificeert daarbij twee filenamen.
Het programma leest een tekstfile met de eerste filenaam als naam.
Daarna schrijft het een tekstfile met de tweede filenaam als naam
De uitvoer moet elk woord uit de invoer op aparte regels vermelden.
De woorden staan daarbij op (alfabetische, of eigenlijk Unicode) volgorde,
waarbij dubbele woorden slechts éénmaal worden vermeld.

Elk groepje karakters zonder spatie erin beschouwen we als woord.
Je mag ervan uitgaan dat er tussen de woorden precies 1 spatie staat.

Als de gebruiker te weinig of te veel filenamen opgeeft, of als er een fout optreedt bij het lezen of schrijven, krijgt de gebruiker daarvan een korte melding.

Voorbeeld: als de invoer de volgende twee regels bevat:

```
dit IS een *@#$ voorbeeld
van een tekst!
```

dan moet de uitvoer de volgende zeven regels bevatten:

```
*@#$
IS
dit
een
tekst!
van
voorbeeld
```

Hint: het is niet nodig om het algoritme voor het sorteren en ont-dubbelen zelf helemaal te ontwikkelen. Je kunt daarvoor de library gebruiken.
Als je niet weet hoe het sorteren of ont-dubbelen moet gebeuren, dan kun je toch nog een flink deel van de punten krijgen door alle woorden van de invoer (op aparte regels) naar de uitvoer te kopiëren.

Einde tentamen