Informatica Instituut, Faculteit Wiskunde en Informatica, UU.
In electronische vorm beschikbaar gemaakt door de TEX van A–Es^2.
Het college IP werd in 1999/2000 gegeven door drs. J. D. Fokker.

Imperatief Programmeren (IP)
9 mei 2000

Opgave 1

a) Kan het keyword **this** worden gebruikt in een methode die *static* is, maar niet *public*? Licht het antwoord (kort) toe.

b) Stel dat er de volgende klassen zijn:

```
class Hoog {\ldots}
class Midden extends Hoog {\ldots}
class Laag extends Midden {\ldots}
```

en dat de volgende declaraties en toekenningen zijn gedaan:

```
Hoog h = new Hoog();
Laag l = new Laag();
Midden m;
```

Van de volgende twee toekenningen is er slechts éń toegestaan, de ander is fout.

```
m = h;
m = 1;
```

Welke toekenning is fout, en waarom?

c) Schrijf een methode *vaakste* met als parameter een array van gehele getallen. Je mag zonder controla aannemen dat de array minstens één getal bevat, en dat de getallen niet kleiner dan 0 en niet groter dan 100 zijn. De methode moet opleveren welk getal het vaakst voorkomt in de array.

d) Beschrijf hoe een object van de klasse *StringTokenizer* kunt gebruiken, en in welke situatie dat handig is.

Opgave 2

Schrijf een statische methode *maximumTotaal* met de volgende specificaties. De methode heeft als parameter een array die één of meer namen van files bevat.
Op elke regel van elke file staat een geheel, niet-negatief getal.
De methode moet als resultaat de naam opleveren van de file waarin de som van alle gatallen het grootste is.
Als er meerdere files zijn met een evengroot totaal, mag je er daarvan één kiezen.
Als een file helemaal geen regels bevat kun je voor die file 0 als totaal hanteren.
Als een file niet bestaat of niet leesbaar is, moet die file buiten beschouwing blijven, maar moeten wel de andere files bekeken worden.
Je mag zonder controle aannemen dat er op elke regel van de files inderdaad een getal staat.

Opgave 3

Schrijf een applet met de volgende specificaties.
Boven in beeld is een drietal knoppen zichtbaar, met als opschrift respectievelijk "schoon", "positie" en "beweeg". De hele rest van het window kan door de gebruiker overal worden aangeklikt. Iedere keer als de gebruiker een punt van het window aanklikt, verschijnt gecentreerd op die plaats een vierkantje van 5×5 beeldpunten. Maximaal mogen honderd punten worden aangeklikt.
Als de gebruiker toch meer punten probeert aan te klikken, gebeurt er niets. Als de gebruiker op de knop "positie" drukt, wordt bij elke tot nu toe aangeklikt punt de positie getoond (als x en y-coördinaat in het Java-coördinatenstelsel). Als de gebruiker nogmaals op de knop "positie" drukt, verdwijnen de getallen weer; bij een derde druk op de knop verschijnen ze weer, enzovoort. Als de gebruiker op de knop "schoon" drukt, verwijen allen punten, en kan de gebruiker beginnen met het aanklikken van nieuwe punten.
Als de gebruiker op de knop "beweeg" drukt, beginnen de punten met een snelheid van 10 beeldpunten per seconde naar de dichtstbijzijnde zijmuur te kruipen. De punten in de linkerhelft van het window kruipen dus naar links, de punten in de rechterhelft naar rechts. Ook de punten die daarna nog worden aangeklikt, beginnen meteen te bewegen. Als de punten de muur hebben bereikt, blijven ze stilstaan: ze blijven dus nog net zichtbaar. Als de gebruiker nogmaals op "beweeg" drukt, stopt de beweging; bij een derde druk op de knop gaan de punten weer bewegen, enzovoort.

Opgave 4

In deze opgave verplaatsen we ons in de schrijver van het packge *java.util*: je moet zelf (een deel van) de klasse *stack* schrijven. De bestaande klassen *Stack* en *Vector* mag je daarbij **niet** gebruiken.
De klasse Stack kent vier methodes: *Stack, push, pop* en *empty*; zie het overzicht.
(In het echt zijn er wel meer methodes, maar die laten we hier buiten beschouwing.)
In een stack kunnen objecten worden opgeslagen öp een stapel": met de methode *push* leg je iets bovenop de stapel, en met de methode *pop* kun je het object van de stapel pakken dat het laatst bovenop de stapel is gelegd (en daarbij tevens van de stapel verwijderen). De methode *Stack* creëert een lege stapel, en de methode *empty* kijkt of een stapel op dit moment leeg is.

a) Definieer de klasse *Stack*, waarbij je de objecten "achter de schermen" in een array opslaat. In eerste instantie kun je in deze array ruimte maken voor 10 elementen, maar als de gebruiker *push* aanroept op een moment dat er al 10 elementen op de stapel liggen, moet je de beschikbare ruimte verdubbelen. Als ook die ruimte op is, moet ook de ruimte weer worden verdubbeld, zodat er ruimte komt voor 40 objecten, enzovoort.

b) Schrijf een statische methode *test* waarin de klasse *Stack* als volgt uitgeprobeerd wordt: er wordt een nieuwe stack gemaakt, daarop worden de strings "aap", "noot", en "mies" neergezet, daarna wordt het bovenste element van stack gepakt, en er wordt getest of dat een string is die de letters "mies" bevat. De uitkomst van de vergelijking wordt als waarheidswaarde opgeleverd als resultaat van de methode *test*. (Als de test goed verloopt, zal de methode dus de waarde *true* opleveren.)