Faculteit der Exacte Wetenschappen tentamen Datastructuren
Vrije Universiteit 18 december 2003

Studenten die het tweede deeltentamen Datastructuren maken krijgen 2:00 uur de tijd en maken de opgaven 2 t/m 3.

Studenten die het gehele tentamen Datastructuren maken krijgen 3:00 uur de tijd en maken de opgaven 1 t/m 3.

U I T W E R K I N G E N

Opgave 1.

a) De subclass erft geen private velden en constructors.

Nee, het is niet mogelijk om in een subclass te zeggen dat deze subclass een bepaalde methode uit de superclass niet wil erven.

Ja, het mogelijk om in een subclass een methode, die uit de superclass geerfd wordt, te herdefinieren.
Dit wordt gedaan door een methode met dezelfde signature te maken als de methode die geherdefinieerd dient te worden.
N.B. herdefinieren = override
signature = naam methode + parameterlijst + return type + modifiers

Ja, het mogelijk in een subclass om een methode, die uit de superclass geerfd wordt, uit te breiden.
Dit wordt gedaan door een methode met dezelfde signature te maken als de methode die geherdefinieerd dient te worden en een body die buiten de statements van de uitbreiding de oorspronkelijke methode uit de superclass aanroept met super.naamMethode(parameterlijst).

De methode "void doeIets()" uit de superclass kan b.v. als volgt worden uitgebreid:

```
void doeIets () {
    super.doeIets();
    // uitbreidingen
}
```

In een subclass moet de default constructor uit de superclass worden aangeroepen met de aanroep "super()".

b) Objecten die een instantie zijn van een class X hebben niet alleen het type X maar o.a. ook het type van alle superclasses van de class X. Aangezien de class Object per definitie de superclass is van iedere andere class, heeft ieder object het type Object.

Aan de elementen van een array met het type "Object[]" kunnen dus objecten van willekeurige types worden toegekend.

```
static final int MAX_AANTAL_ELEMENTEN = 35;
Object[] generiekeRij = new Object[MAX_AANTAL_ELEMENTEN];
```

♀

c) maak een interface XyzInterface als volgt

```
interface XyzInterface {
   // lege interface
}
```

en breidt de declaraties van de classes X, Y en Z als volgt uit

```
class X implements XyzInterface {
   ....
}

class Y implements XyzInterface {
   ....
}

class Z implements XyzInterface {
   ....
}
```

waarna de variabele xyzRij als volgt gedeclareerd kan worden:

```
static final int MAX_AANTAL_ELEMENTEN = 20;
XyzInterface[] xyzRij = new
XyzInterface[MAX_AANTAL_ELEMENTEN];
```

d) Er is sprake van overloading als twee verschillende methodes dezelfde naam (maar verschillende signature) hebben.

Er is sprake van polymorfisme als bij dezelfde methode objecten van verschillende types als parameter meegegeven kunnen worden.

Een abstract method is een methode zonder body.

Een final method is een methode die niet kan worden veranderd in een subclass.

Een protected method is een methode die alleen door classes in hetzelfde package of door subclasses gebruikt kan worden.

Opgave 2.

a) A B C D E

```
A  0  1  1  0  0
B  1  0  0  1  1
C  0  1  0  0  1
D  1  0  0  0  0
E  0  0  0  0  0
```

♀

b)

```
class LijstKnoop {
   Object data;
   LijstKnoop next;

   LijstKnoop (Object data, LijstKnoop next) {
            this.data = data;
            this.next = next;
   }
}

class Lijst inplements LijstInterface {

   private LijstKnoop first;

   public Lijst () {
            first = null;
   }

   public Lijst insert (Object obj) {
            first = new LijstKnoop(obj, first);
            return this;
   }
}
```

c)

```
GraafKnoop voegKnoopToe (char c) { // c is een hoofdletter
   GraafKnoop graafKnoop = new GraafKnoop(c);
   lijst.insert(graafKnoop);
   return graafKnoop;
}

void voegKantToe (GraafKnoop k1, GraafKnoop k2) {
   k1.buren.insert(k2);
   k2.buren.insert(k1);
}
```

Opgave 3.

a) Een boom is een binaire boom indien voor iedere knoop geldt dat deze knoop twee subbomen bevat.

De hoopvoorwaarden zijn dat voor iedere knoop moet gelden dat de inhoud van deze knoop >= is aan de inhoud van de aanwezig kinderen.

Een complete binaire boom is een boom waarvan alle lagen behalve de onderste vol moeten zijn. In de onderste laag moeten alle aanwezige knopen "links aangeschoven" zijn.

```
boolean hoop (Knoop k) {
   if k == null) {
            return true;
   }

   return (k.links  != null && k.data >= k.links.data)  &&
        (k.rechts != null && k.data >= k.rechts.data) &&
             hoop(k.links)                          &&
             hoop(k.rechts)                           ;
}
```

Nee, een binaire zoekboom, die niet twee gelijken knopen heeft, kan geen hoop zijn.

♀

b) Een queue is een lineaire datastructuur (rij) waar alleen achteraan kan worden toegevoegd en alleen vooraan kan worden weggehaald.

Een priority queue is een queue gesorteerd op prioriteit waarbij elementen met dezelfde prioriteit een queue vormen.

TOEVOEGEN
Voeg het toe te voegen element toe aan de hoop door dit op de onderste laag rechts van het meest rechtse al aanwezige element te zetten. Verwissel dit zojuist toegevoegde element met de parent zolang dit element groter is dan de parent.

VERWIJDEREN
Pak de inhoud van de wortel van de hoop. Vervang de inhoud van de wortel van de hoop door de inhoud X van de meest rechtse knoop van de onderste laag. Verwijder de meest rechtse knoop van de onderste laag. Verwissel de waarde X met de grootste waarde van de twee kinderen indien een van de kinderen groter is dan X.

c)
```
Iterator inOrder (Knoop k) {
   Rij rij = new Rij();
   inOrder(k, rij);
   return rij;
}

private void inOrder (Knoop k, Rij rij) {
   if (k == null) {
            return;
   }

   inOrder(k.links, rij);
```

```
    rij.voegToe(k.data);
    inOrder(k.rechts, rij);
}
```