Faculteit der Exacte Wetenschappen — toets Datastructuren

Vrije Universiteit — 17 mei 1999

tijdsduur : 1.5 uur

UITWERKINGEN

==========

Opgave 1.

a)
```
public Object clone (Object rhs) {
    X src  = (X) rhs;
    X dest = new X();

    dest.string = new String(src.string);
    dest.i = src.i;

    return dest;
}
```

b) LIFO = last in first out (b.v. stack)

FIFO = first in first out (b.v. queue)

c) 1234

Y1234Y

ZtestZ

ZYZtestZYZ

ZtestZ

ZYZtestZYZ

ZtestZ

ZYZtestZYZ

d) Een abstracte methode is een methode waarvan alleen de heading gegeven is.

Verschillen tussen een interface en een abstracte class zijn:

- In een abstracte class kunnen ook niet-abstracte methodes staan, in een interface kunnen alleen abstracte methodes staan.
- In een class die een abstracte class extends hoeven de abstracte methodes uit die abstracte class niet geimplementeerd te worden, in een class die een interface implements moeten alle abstracte methodes uit die interface geimplementeerd worden.

Omdat nog niet alle methodes van een abstracte class bekend zijn, kunnen er ook nog geen instanties van zo'n abstracte class gemaakt worden waar die methodes in aanwezig zijn.

♀Opgave 2.

a)
```
private Knoop wortel;
private int aantalElementen;
```

b)
```
Bag () {
    init();
}
```

c)
```
Bag (Bag b) {
    wortel = kopie(b.wortel);
    aantalElementen = b.aantalElementen;
}

private Knoop kopie (Knoop w) {
    if (w == null) {
            return null;
    }
    return new Knoop(w.data, w.aantal, kopie(w.left), kopie(w.right));
}
```

d)
```
Bag include (Data d, int aantal) {
    if (aantal(d) == 0) {
            aantalElementen += 1;
    }
    wortel = include(wortel, d, aantal);
    return this;
```

```
}

private Knoop include (Knoop w, Data d, int a) {
    if (w == null) {
            return new Knoop(d, a);
    }

    if (d.compareTo(w.data) < 0) {
            w.left = include(w.left, d, a);
    } else
    if (d.compareTo(w.data) > 0) {
            w.right = include(w.right, d, a);
    } else { // d.compareTo(w.data) == 0
            w.aantal += a;
    }
    return w;
}

int size () {
    return aantalElementen;
}

int aantal (Data d) {
    return aantal (wortel, d);
}
```

```
private int aantal (Knoop w, data d) {
    if (w == null) {
            return 0;
    }

    if (d.compareTo(w.data) < 0) {
            return aantal(w.left, d);
    } else
    if (d.compareTo(w.data) > 0) {
            return aantal(w.right, d);
    } else { // d.compareTo(w.data) == 0
            return w.aantal;
    }
}
```