Pakken `simset`

Pakken `simset` er en Java-pakke til håndtering af dobbelthægtede lister. Funktionalitet og implementering svarer nøje til en tilsvarende pakke i programmeringssproget Simula.

Hvert medlem i en liste har både en reference til sin efterfølger (`suc`) og sin forgænger (`pred`) i listen. Udover listemedlemmerne, der nedarver fra klassen `Link`, har en liste også et hoved (`Head`), der indeholder generel information om listen (f.eks. om hvilket medlem, der står først i listen,).

En liste er internt organiseret i en komplet cirkulær struktur bestående af et `Head`-objekt og en samling `Link`-objekter. Den fælles del af `Head` og `Link` er erklæret i en fælles overklasse kaldet `Linkage`.

I det følgende beskrives de individuelle klasser og deres metoder kort, idet følgende erklæringer antages at være gyldige.

```
Head hd;
Link lk;
Linkage lg;
```

Klassen `Linkage`:

```
public class Linkage {
    public Link suc();
    public Link pred();
    public Linkage prev;
}
```

`lk.suc()` returnerer en reference til det efterfølgende listemedlem, hvis `lk` er er i en liste, og `lk` ikke er det sidste medlem; ellers `null`.

`hd.suc()` returnerer en reference til det første medlem i listen, hvis listen ikke er tom; ellers `null`.

`lk.pred()` returnerer en reference til det foregående listemedlem, hvis `lk` er er i en liste, og `lk` ikke er det sidste medlem; ellers `null`.

`hd.pred()` returnerer en reference til det sidste medlem i listen, hvis listen ikke er tom; ellers `null`.

`lk.prev()` returnerer `null`, hvis lk ikke er i en liste, en reference til listens hoved, hvis `lk` er det første medlem i en liste; ellers returneres en reference til `lk`'s forgænger i listen.

`hd.prev()` returnerer en reference til `hd`, hvis `hd` er tom; ellers en reference til det sidste medlem i listen.

Klassen `Head`:

```
class Head extends Linkage {
    public Link first();
    public Link last();
    public int cardinal();
    public boolean empty()
    public void clear();
}
```

`hd.first()`	er ækvivalent med `hd.suc()`.
`hd.last()`	er ækvivalent med `hd.pred()`;
`hd.cardinal()`	returnerer antallet af medlemmer i listen (0, hvis `hd` er tom).
`hd.empty()`	returnerer `true`, hvis `hd` refererer en liste uden medlemmer, `false` hvis `hd` refererer en liste med et eller flere medlemmer.
`hd.clear()`	fjerner alle medlemmer fra listen (så den bliver tom).

Klassen `Link`:

```
class Link extends Linkage {
    public void out();
    public void into(Head h);
    public void precede(Linkage x);
    public void follow(Linkage x;
}
```

`lk.out()` fjerner `lk` fra en liste og retablerer `suc`- og `pred`-relationerne imellem dens tidligere forgænger- og efterfølgermedlemmer. Hvis `lk` ikke er medlem af nogen liste, sker der intet.

`lk.into(hd)` Først kaldes `lk.out()`. Hvis `hd` == `null`, så sker der intet; ellers indsættes `lk` som det nye sidste medlem i listen hd.

`lk.precede(lg)` Først kaldes `lk.out()`. Hvis `lg` == `null` eller ikke er i nogen liste, så sker der intet; ellers indsættes `lk` i den samme liste som `lg` som den nye forgænger til `lg` (`lg` kan referere enten et `Head`- eller et `Link`-objekt).

`lk.follow(lg)` Som `precede(lg)`, men i stedet bliver `lk` den nye efterfølger for `lg`.