Løsningsforslag for eksamen i IN1010 og INF1010

Stein Gjessing Dag Langmyhr

7. juni 2019 (oppdatert 24. mai 2022)

Generelle kommentarer

- Dette er et løsnings*forslag* og ikke en fasit. Det er bare én av mange ulike måter å løse oppgaven på.
- I koden er det lagt inn noen utskrifter for å teste koden; de er markert, og det er ikke forventet at dere skriver slikt på eksamen.

Oppgave 1: Klassen Tidspunkt

Tidspunkt.java

```
class Tidspunkt implements Comparable<Tidspunkt> {
    int aar, maned, dag, time, minutt, sekund;

    Tidspunkt(int a, int m, int d, int t, int min, int s) {
        aar = a;  maned = m;  dag = d;
        time = t;  minutt = min;  sekund = s;
    }

    @Override
    public int compareTo (Tidspunkt t) {
        // Sortering etter vanlig oppfatning av tid:
        // tidligere tidspunkter kommer først.

        if (aar != t.aar) return aar - t.aar;
        if (maned != t.maned) return maned - t.maned;
        if (dag != t.dag) return dag - t.dag;
        if (time != t.time) return time - t.time;
        if (minutt != t.minutt) return minutt - t.minutt;
        return sekund - t.sekund;
    }

    // Ikke bedt om i oppgaveteksten:
    @Override
    public String toString() {
        return String.format("%d.%d.%d %d:%02d:%02d",
                             dag, maned, aar, time, minutt, sekund);
    }
}
```

Oppgave 2: Klassene Hund og Kull

Hund.java

```
class Hund implements Comparable<Hund> {
    String navn;
    Kull mittKull;
    Tidspunkt minFodselstid;
    Hund neste = null;

    Hund(Kull k, String navn, Tidspunkt fodt) {
        this.navn = navn;
        mittKull = k;
        minFodselstid = fodt;
    }

    @Override
    public int compareTo(Hund h) {
        return minFodselstid.compareTo(h.minFodselstid);
    }

    public Hund mor() {
        return mittKull.mor;
    }

    public Hund far() {
        return mittKull.far;
    }

    public boolean erHelsosken(Hund h) {
        return mor()!=null && far()!=null && mor()==h.mor() && far()==h.far();
    }

    public boolean erHalvsosken(Hund h) {
        return (mor()!=null && mor()==h.mor() || far()!=null && far()==h.far()) &&
            ! erHelsosken(h);
    }

    public Hund finnEldsteKjenteOpphav() {
        if (mor() == null)
            return far()==null ? this : far().finnEldsteKjenteOpphav();

        if (far() == null)
            return mor().finnEldsteKjenteOpphav();

        Hund morsEldsteOpphav = mor().finnEldsteKjenteOpphav();
        Hund farsEldsteOpphav = far().finnEldsteKjenteOpphav();
        if (morsEldsteOpphav.compareTo(farsEldsteOpphav) < 0)
            return morsEldsteOpphav;
        else
            return farsEldsteOpphav;
    }
}
```

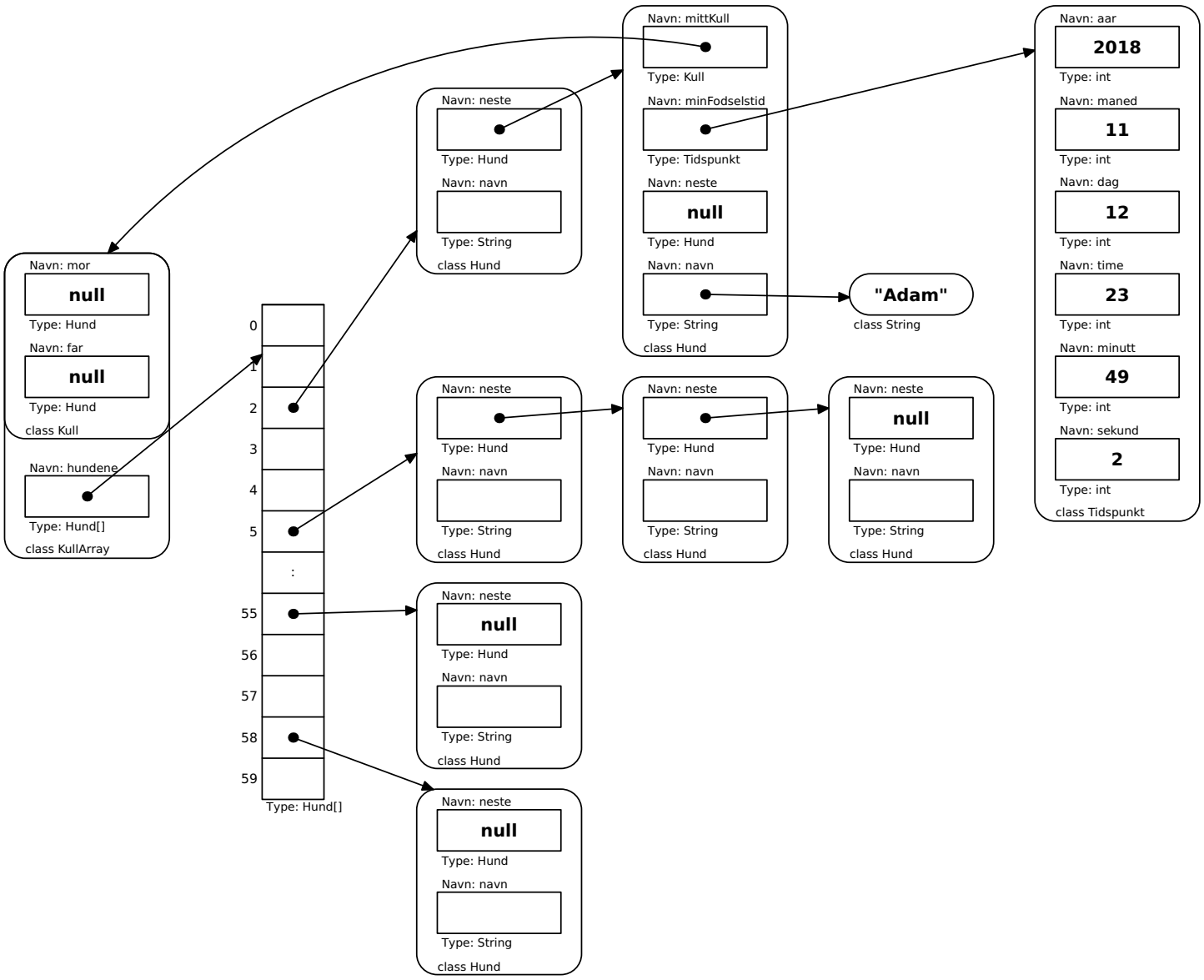
Kull.java

```
import java.util.Iterator;

abstract class Kull implements Iterable<Hund> {
  Hund mor, far;

  Kull (Hund mor, Hund far) {
    this.mor = mor;
    this.far = far;
  }

  public abstract void settInn(Hund h);
  public abstract Iterator<Hund> iterator();
}
```

Oppgave 3: Klassene KullListe og KullArray

KullListe.java

```
import java.util.Iterator;

class KullListe extends Kull {
    Hund forste = null;

    KullListe (Hund mor, Hund far) {
        super(mor,far);
    }

    public void settInn(Hund h) {
        if (forste==null) {
            // Legg inn i tom liste:
            forste = h;
            return;
        }

        // Skal den nye hunden inn først?
        if (h.compareTo(forste) > 0) {
            h.neste = forste;
            forste = h;
            return;
        }

        Hund p = forste;
        while (true) {
            if (p.neste == null) {
                // Plasser sist i listen:
                p.neste = h;
                break;
            }  else if (h.compareTo(p.neste) > 0) {
                // Inn her:
                h.neste = p.neste;
                p.neste = h;
                break;
            } else {
                // Let videre:
                p = p.neste;
            }
        }
    }

    @Override
    public Iterator<Hund> iterator() {
        return new HundeIterator();
    }

    class HundeIterator implements Iterator<Hund> {
        private Hund denne = forste;

        public boolean hasNext() {
            return denne != null;
        }

        public Hund next() {
            Hund svar = denne;
            denne = denne.neste;
            return svar;
        }
    }
}
```


Navn: mittKull
Type: Kull
Navn: minFodselstid
Type: Tidspunkt
Navn: neste
null
Type: Hund
Navn: navn
Type: String
class Hund
Navn: aar
2018
Type: int
Navn: maned
11
Type: int
Navn: dag
12
Type: int
Navn: time
23
Type: int
Navn: minutt
49
Type: int
Navn: sekund
2
Type: int
class Tidspunkt
"Adam"
class String
Navn: mor
null
Type: Hund
Navn: far
null
Type: Hund
class Kull
Navn: hundene
Type: Hund[]
class KullArray
0
1
2
3
4
5
:
55
56
57
58
59
Type: Hund[]
Navn: neste
Type: Hund
Navn: navn
Type: String
class Hund
Navn: neste
Type: Hund
Navn: navn
Type: String
class Hund
Navn: neste
null
Type: Hund
Navn: navn
Type: String
class Hund
Navn: neste
null
Type: Hund
Navn: navn
Type: String
class Hund
Navn: neste
null
Type: Hund
Navn: navn
Type: String
class Hund

KullArray.java

```
import java.util.Iterator;

class KullArray extends Kull {
    private Hund[] hundene = new Hund[60];

    KullArray (Hund mor, Hund far) {
        super(mor,far);
    }

    public void settInn(Hund h) {
        int sek = h.minFodselstid.sekund;
        h.neste = hundene[sek];
        hundene[sek] = h;
    }

    public void skrivAlle() {
        for (int i = 0;  i < 60;  i++) {
            Hund h = hundene[i];
            while (h != null) {
                System.out.println(h.navn);
                h = h.neste;
            }
        }
    }

    // Alternativ løsning:
    public void skrivAlle2() {
        for (Hund h: this)
            System.out.println(h.navn);
    }

    // Det spørres ikke etter denne iteratoren.
    @Override
    public Iterator<Hund> iterator() {
        return new HundeIterator();
    }

    class HundeIterator implements Iterator<Hund> {
        int pos;
        Hund denne;

        HundeIterator() {
            pos = 0;  denne = null;
            while (true) {
                if (pos >= 60) break;
                if (hundene[pos] != null) {
                    denne = hundene[pos];  break;
                }
                pos++;
            }
        }

        @Override
        public boolean hasNext() {
            return denne != null;
        }

        @Override
        public Hund next() {
            Hund svar = denne;
            if (denne.neste != null) {
                denne = denne.neste;
            } else {
                pos++;
                while (pos < 60 && hundene[pos] == null) pos++;
                denne = pos < 60 ? hundene[pos] : null;
            }
            return svar;
        }
    }
}
```