Løsningsforslag for prøveeksamen i IN1010

Dag Langmyhr

våren 2023

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

```
// Oppgave 1

class Tidspunkt implements Comparable<Tidspunkt> {
    int aar, mnd, dag, time, min, sek;

    Tidspunkt(int aar, int mnd, int dag, int time, int min, int sek) {
        this.aar = aar;  this.mnd = mnd;  this.dag = dag;
        this.time = time;  this.min = min;  this.sek = sek;
    }

    @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 (mnd != t.mnd) return mnd - t.mnd;
        if (dag != t.dag) return dag - t.dag;
        if (time != t.time) return time - t.time;
        if (min != t.min) return min - t.min;
        return sek - t.sek;
    }

    // Ikke bedt om i oppgaven:
    @Override
    public String toString() {
        return String.format("%d.%d.%d %d:%02d:%02d",
                             dag, mnd, aar, time, min, sek);
    }
}
```

Oppgave 2: Klassen Hund

Hund.java

```
// Oppgave 2

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

    // invariant: mittKull != null

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

    // Oppgave 2a
    public Hund mor () {
        return mittKull.mor;
    }

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

    // Oppgave 2b
    @Override
    public int compareTo (Hund h) {
        return minFodselstid.compareTo(h.minFodselstid);
    }

    // Oppgave 2c
    public boolean erHelsosken (Hund h) {
        if (mittKull == h.mittKull) {
            // To hunder fra samme kull er alltid helsøsken,
            // selv om vi ikke vet hvem moren og faren er.
            return true;
        }

        if (mor() == null || far() == null) {
            // Hvis en eller begge foreldre er ukjent,
            // kan vi ikke garantere at de to hundene er helsøsken.
            return false;
        }

        // Når vi nå vet at vi kjenner mor og far, kan vi sjekke
        // om de to hundene har samme foreldre.
        return mor()==h.mor() && far()==h.far();
    }

    public boolean erHalvsosken (Hund h) {
        if (erHelsosken(h)) {
            // Hvis de er helsøsken, er de ikke halvsøsken.
            return false;
        }

        if (mor() == null || far() == null) {
            // Hvis en eller begge foreldre er ukjent,
            // kan vi ikke garantere at de to hundene er halvsøsken.
            return false;
        }

        // Ellers må de ha én forelder felles.
        return mor()==h.mor() || far()==h.far();
    }

    // Oppgave 2d
    public Hund finnEldsteKjenteOpphav () {
        if (mor() == null && far() == null)
            return this;
        if (mor() == null)
            return far().finnEldsteKjenteOpphav();
```

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

        // Hunden har både mor og far.
        Hund morsEldsteOpphav = mor().finnEldsteKjenteOpphav();
        Hund farsEldsteOpphav = far().finnEldsteKjenteOpphav();
        if (morsEldsteOpphav.compareTo(farsEldsteOpphav) < 0)
            return morsEldsteOpphav;
        else
            return farsEldsteOpphav;
    }

    // Ikke bedt om i oppgaven:
    @Override
    public String toString () {
        return navn + "(f " + minFodselstid + ")";
    }
}
```

Oppgave 3: Klassene Kull, KullListe og KullArray

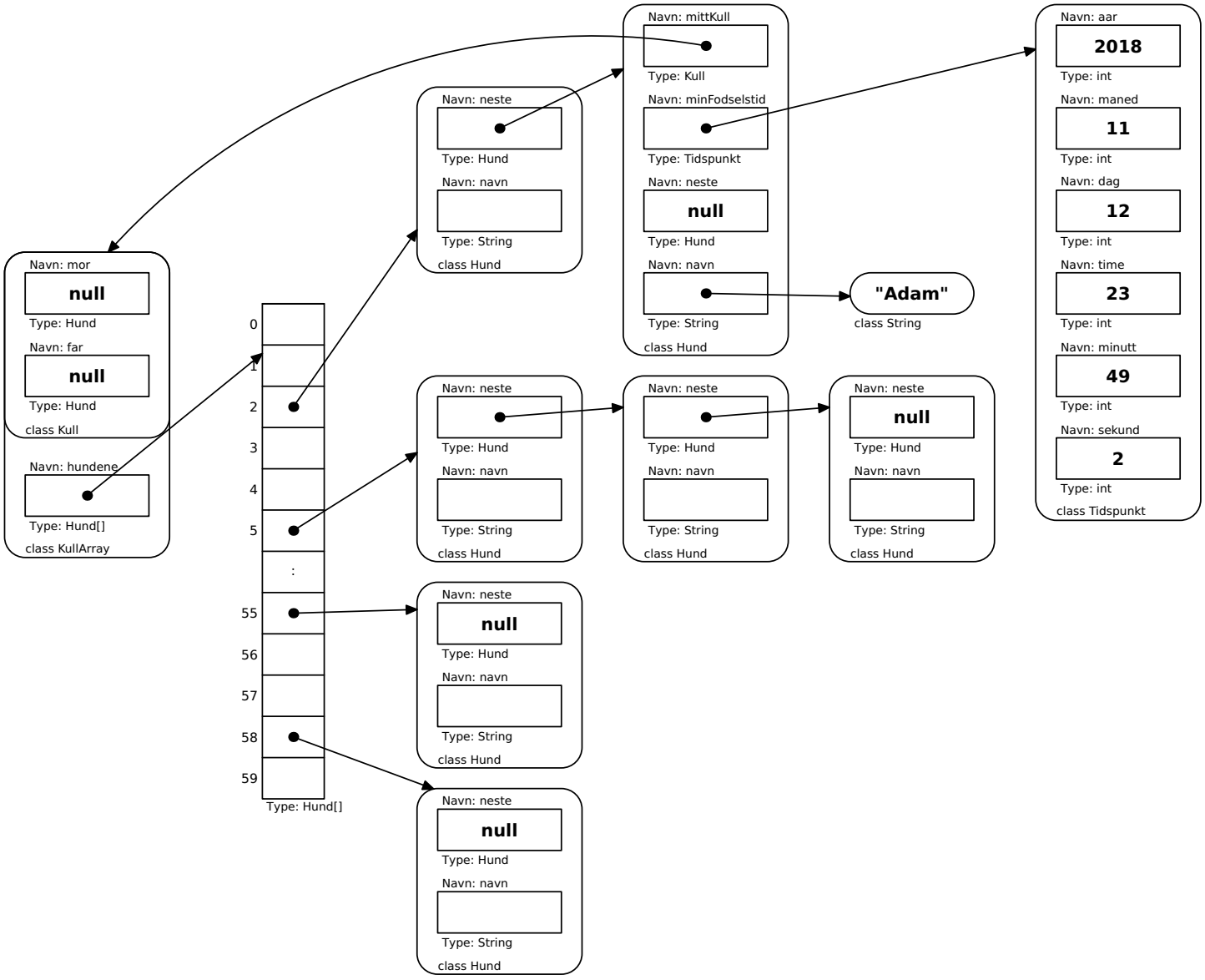
Kull.java

```
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 void skrivUtAlle () {
        for (Hund h: this)
            System.out.println("  " + h);
    }

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

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;
        }
    }
}
```



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.sek;
        h.neste = hundene[sek];
        hundene[sek] = h;
    }

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

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

        HundeIterator () {
            pos = 0;  denne = hundene[0];
            finnNesteHund();
        }

        private void finnNesteHund () {
            while (denne == null) {
                ++pos;
                if (pos >= 60) return;
                denne = hundene[pos];
            }
        }

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

        @Override
        public Hund next () {
            Hund svar = denne;
            denne = denne.neste;
            finnNesteHund();
            return svar;
        }
    }

    // Ikke bedt om i oppgaven:
    public void dumpArray () {
        for (int s = 0;  s < 60;  ++s) {
            if (hundene[s] == null) continue;

            System.out.print(s + ":");
            Hund h = hundene[s];
            while (h != null) {
                System.out.print(" " + h);
                h = h.neste;
            }
            System.out.println();
        }
    }
}
```