INF110 Ukeoppgaver: Ibsen.Java
NB! Kun tekst. En Java-kode versjon finnes også!

```
    import inf101.*;

    public class Ibsen {
        public static void main(String[] args) {
            String ord;
            int antallOrd = 0;

            String sep = ". :,;-?!\"";
            Inn innfil = new Inn(args[0]);
            IbsenTre tre = new IbsenTre();

            innfil.skipSep(sep);
            while (!innfil.endOfFile()) {
                ord = innfil.inString(sep).toLowerCase();
                tre.settInn(new IbsenElem(ord));
                antallOrd++;
                innfil.skipSep(sep);
            }

            System.out.println("Totalt antall ord: " + antallOrd);
            System.out.println("Antall ulike ord: " + tre.size());

            IbsenFrekTre frekTre = new IbsenFrekTre();
            frekTre.innsetting(tre.getRot());

            frekTre.skrivInnfiks();
        }
    }


//  Klasse for å ta vare på ordene og telle opp antall forekomster:

    class IbsenElem implements Comparable {
        String ord; int antall;

        IbsenElem (String s) {
            ord = s; antall = 1;
        }

        public int compareTo(Object x) {
            IbsenElem e = (IbsenElem) x;
            return ord.compareTo(e.ord);
        }

        public String toString() {
            return (ord + " " + antall);
        }
    }
```

```
//  Vanlig node i binært søketre:

    class BinNode {
        Comparable element;
        BinNode venstre;
        BinNode hoyre;

        BinNode(Comparable x) {
            element = x;
        }
    }

//  Vanlig binært søketre der "noe gjøres" ved like elementer.
// (Fordi klassen er public, skulle den ha ligget på en egen fil.
//  Fjern ordet "public" på neste linje hvis du vil kompilere filen.)

    public class BinSokeTre {
        protected BinNode rot;
        protected int antallNoder = 0;

        private void settInn(Comparable x, BinNode n) {
            int i = sammenlign(x, n.element);
            if (i < 0) {
                if (n.venstre == null) {
                    n.venstre = new BinNode(x);
                    antallNoder++;
                } else {
                    settInn(x, n.venstre);
                }
            } else if (i > 0) {
                if (n.hoyre == null) {
                    n.hoyre = new BinNode(x);
                    antallNoder++;
                } else {
                    settInn(x, n.hoyre);
                }
            } else {
                oppdater(n, x);
            }
        }

        public void settInn(Comparable x) {
            if (rot == null) {
                rot = new BinNode(x);
                antallNoder++;
            } else {
                settInn(x, rot);
            }
        }

        protected void oppdater(BinNode n, Comparable x) {
            // Default: Ingenting gjøres for like elementer.
        }

        protected int sammenlign(Comparable n1, Comparable n2) {
            return n1.compareTo(n2);
        }
```

```
    public void skrivInnfiks() {
        innfiks(rot);
    }

    private void innfiks(BinNode n) {
        if (n != null) {
            innfiks(n.venstre);
            System.out.println(n.element.toString());
            innfiks(n.hoyre);
        }
    }

    public int size() {
        return antallNoder;
    }

    public BinNode getRot() {
        return rot;
    }

}   // slutt class BinSokeTre
```

```
class IbsenTre extends BinSokeTre {

    protected void oppdater(BinNode n, Comparable e) {
        IbsenElem ie = (IbsenElem) n.element;
        ie.antall++;
    }

}


class IbsenFrekTre extends BinSokeTre  {

    protected int sammenlign(Comparable c1, Comparable c2) {

        IbsenElem e1 = (IbsenElem) c1;
        IbsenElem e2 = (IbsenElem) c2;

        return e1.antall - e2.antall;
    }

    protected void oppdater(BinNode n, Comparable e) {
        if (n.venstre == null) {
            n.venstre = new BinNode(e);
        } else {
            BinNode b = new BinNode(e);
            b.venstre = n.venstre;
            n.venstre = b;
        }
        antallNoder++;
    }

    public void innsetting(BinNode n) {
        if (n != null) {
            settInn(n.element);
            innsetting(n.venstre);
            innsetting(n.hoyre);
        }
    }

}
```