Klasser

Noen småting vi ikke har tatt før.

Hvordan tenker en programmerer

Gitt et problem, hvordan tenker en programmerer for å finne fram til en løsning?

- Eksempler
- Oppslagstabeller (klassen HashMap)

Når er objekter «like»?

Likhet mellom tallverdier sjekkes med == og !=. Kan vi sjekke pekere på samme måte?

```
String s1 = "";
String s2 = s1 + s1;
if (s1 == s2) { ... }
```

sjekker om `s1` og `s2` er *samme objekt*. Testen over vil derfor aldri slå til.

Semantisk likhet

Ofte er vi mer interessert i om to objekter er «like» i en eller annen betydning, for eksempel Stringer. Metoden equals defineres slik vi ønsker det.

```
  class Bok {
    private String tittel, forfatter, isbn;

    String finnISBN() { return isbn; }

    boolean equals(Bok b) {
      return isbn.equals(b.hentISBN());
    }
}
```

Hvordan tenke når man får et konkret problem?

Svar: Det finnes ingen standardløsning, men man kan lære mye av å høre hvordan andre tenker.

En palindromoppgave

Skriv metoden

```
boolean erPalindrom(String s) { ... }
```

som avgjør om s er et *palindrom*.

Store norske leksikon: «ord som lyder likt, lest både baklengs og forlengs»

Eksempler

Otto regninger

A man, a plan, a canal: Panama

Råd 1

Følg definisjonen; den inneholder ofte en oppskrift.

Les baklengs og forlengs:

```
boolean erPalindrom1(String s) {
    int i1 = 0;
    int i2 = s.length()-1;

    while (i1 < i2) {
        String s1 = s.substring(i1,i1+1);
        String s2 = s.substring(i2,i2+1);
        if (! s1.equals(s2)) return false;
        i1++;  i2--;
    }
    return true;
}
```

Råd 2

Prøv om du kan vri problemet til et annet som er greiere å løse.

Lag baklengsordet:

```
boolean erPalindrom2(String s) {
    String sRev = "";

    for (int i = 0;  i < s.length();  i++) {
        sRev = s.substring(i,i+1) + sRev;
    }
    return s.equals(sRev);
}
```

Råd 3

Sjekk om det finnes biblioteksrutiner du kan bruke.

String kan ikke reversere tekst, men StringBuilder kan:

```
boolean erPalindrom3(String s) {
    StringBuilder sb = new StringBuilder(s);
    return s.equals(sb.reverse().toString());
}
```

Råd 4

Se om du kan redusere problemet litt etter litt.

Sjekk ett tegn først og sist i teksten; hvis OK, kan de fjernes.

```
boolean erPalindrom4(String s) {
    while (s.length() > 1) {
        int len = s.length();
        String sForst = s.substring(0,1);
        String sSist = s.substring(len-1,len);
        if (! sForst.equals(sSist)) return false;

        s = s.substring(1,len-1);
    }
    return true;
}
```

Råd 5

Ta særtilfellene først; det er lettere å løse det generelle problemet.

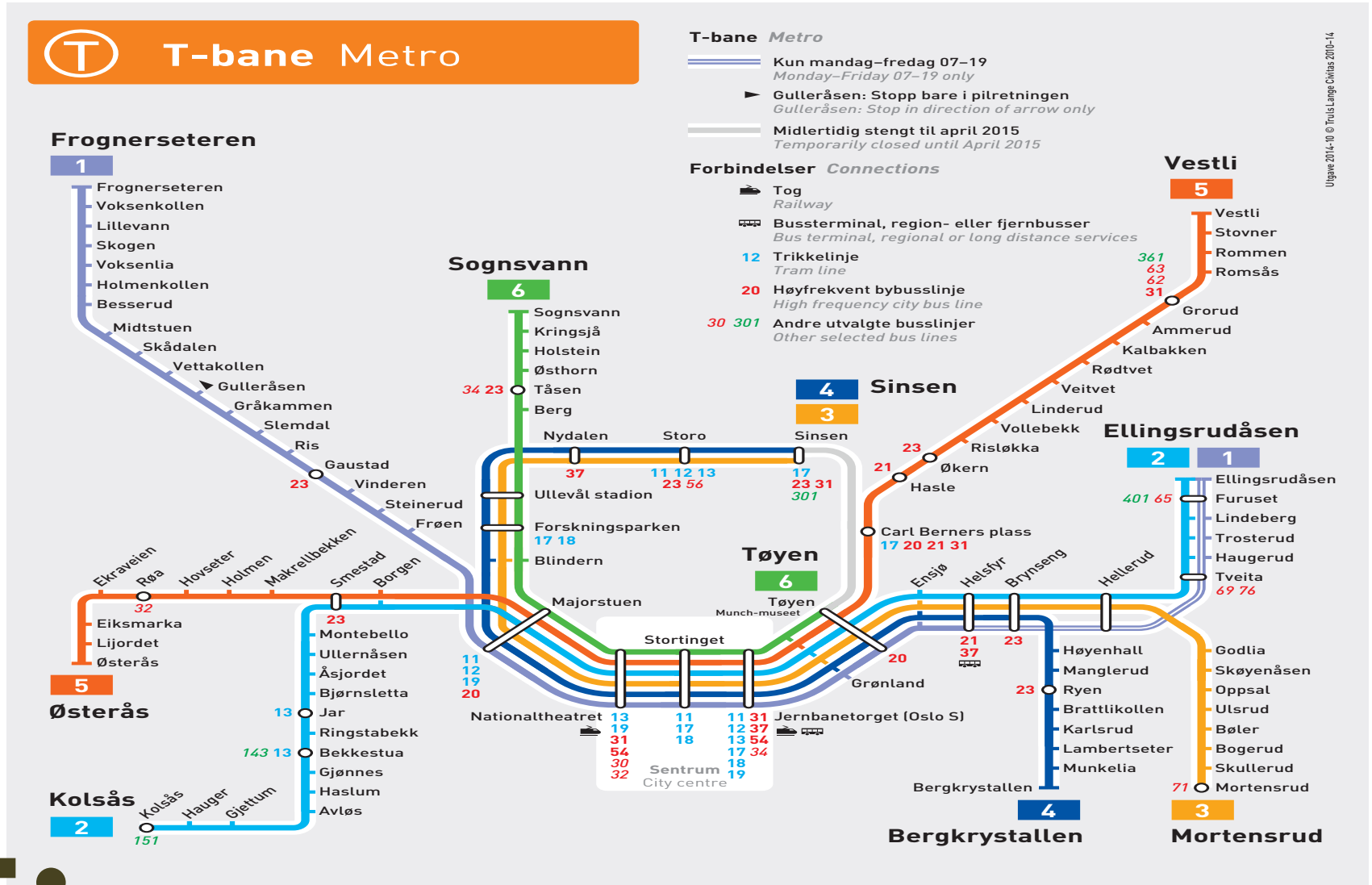
En tekst på 0 eller 1 tegn er alltid et palindrom:

```
boolean erPalindrom5(String s) {
    int len = s.length();
    if (len <= 1) return true;

    String sForst = s.substring(0,1);
    String sSist = s.substring(len-1,len);
    if (! sForst.equals(sSist)) return false;

    return erPalindrom4(s.substring(1,len-1));
}
```

Konklusjonen

Det finnes mange ulike måter å utvikle programmer på.
Med erfaring vil du finne den som passer deg best.

Hvordan komme fra Blindern til Eiksmarka?



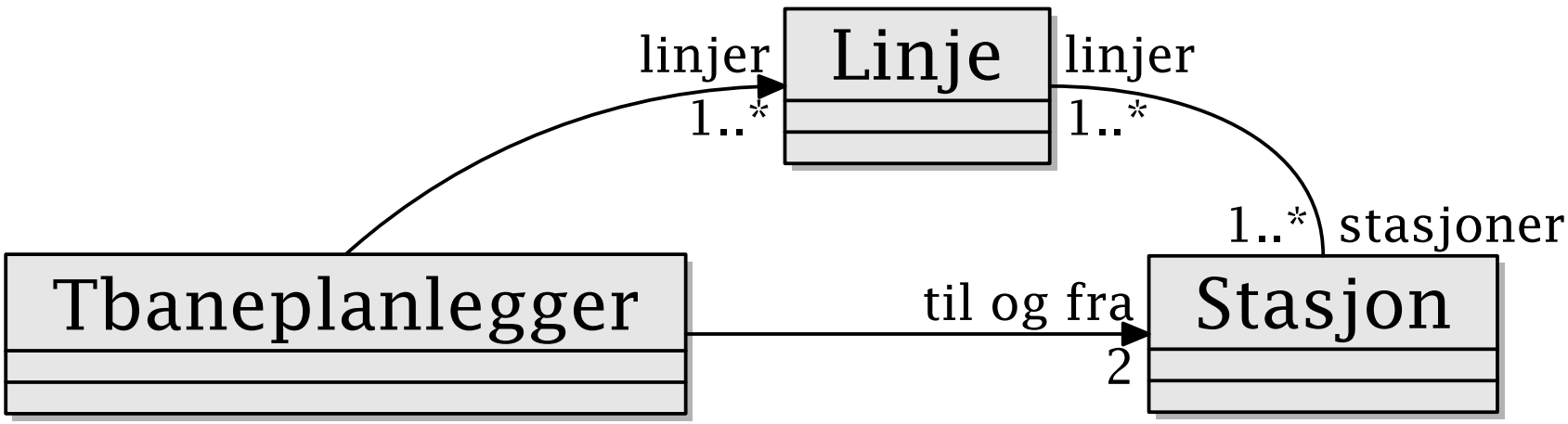
Fase 0: Problemet

Skriv et program som kan fortelle om hvordan man kommer fra en gitt T-banestasjon til en annen:

```
$ java Tbaneplanlegger Blindern Eiksmarka
Du kan komme fra Blindern til Eiksmarka ved bytte paa Majorstuen
```

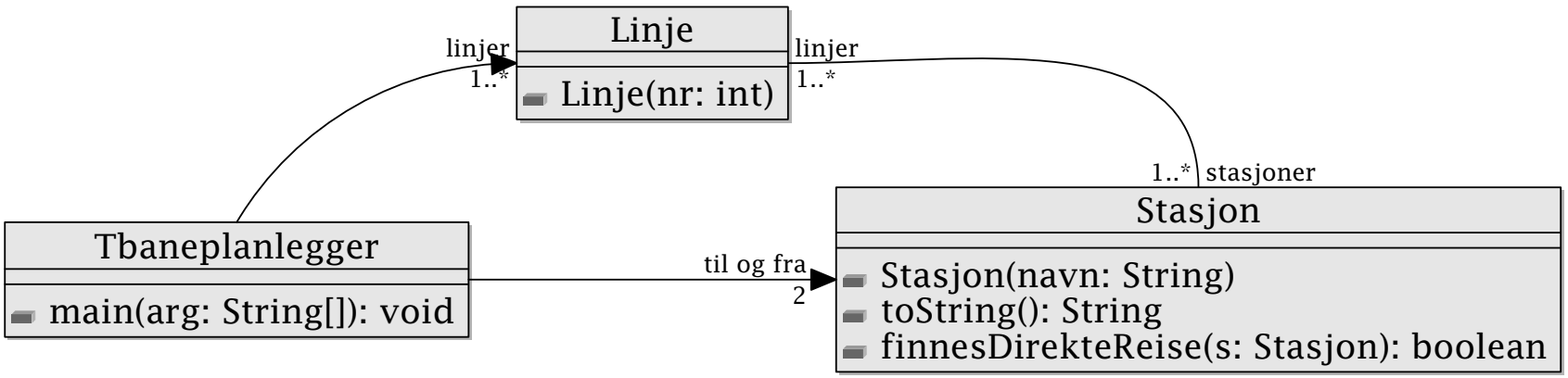
Fase 1: Hvilke klasser trengs?

Dette virker som et passende utganspunkt:



Fase 2: Grensesnittet

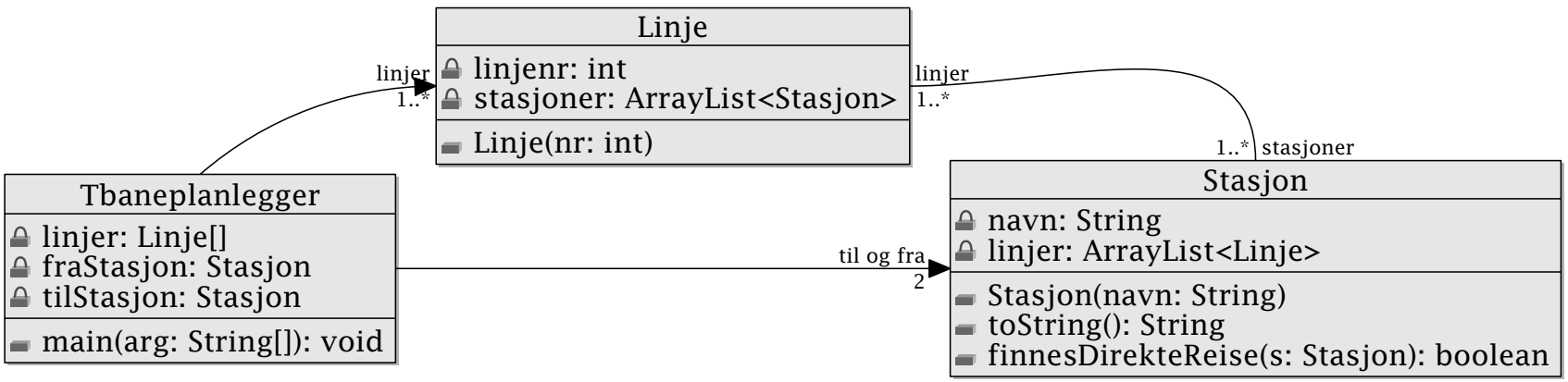
Slikt er alltid litt gjetting i starten, men følgende bør med:



Så kan vi legge til mer etter hvert.

Fase 3: Representasjonen

Vi kan starte med dette utgangspunktet:



Implementasjonen

Hint:

Hvis vi har noen klasser som kan testes alene, bør vi begynne med dem. Hvis ikke, bør vi begynne med hovedprogrammet.

1. Sett opp datastrukturen
2. Finn en løsning ved å følge datastrukturen

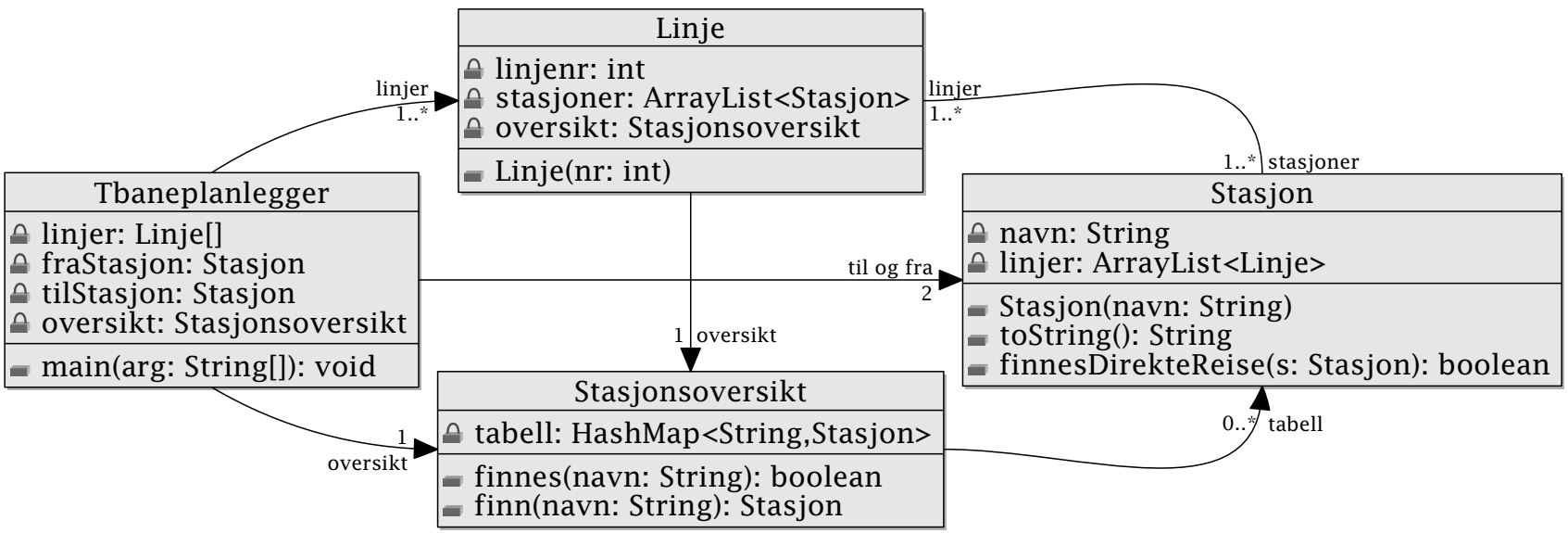
Initiering

Vi har en litt stor struktur med 6 linjer og 100 stasjoner. Alternativene er

1. Skriv Java-kode som bygger opp datastrukturen
2. Les data fra fil.
   - Mindre skriving
   - Mer fleksibelt

Forbedring av opplegget

Vi ser det er vanskelig å finne en Stasjon utifra navnet. Det hadde vært fint å kunne slå opp i en tabell.



Oppslagstabeller

Vi har både arrayer og ArrayList der vi kan slå opp ved å bruke et heltall. Ofte trenger vi å slå opp med en tekst i stedet:

- telefonkatalog
- medlemskartotek

⋮

Derfor har noen laget **HashMap**.

Array

```
C[] a = new C[100];
a[0] = new C();
a[12+3*(7-1)] = a[0];
int len = a.length;

for (int i = 0;  i < a.length;  i++) {
    if (a[i] != null) { a[i].p(); }
}
```

ArrayList

```
import java.util.ArrayList;


        ArrayList<C> al = new ArrayList<>();
        al.add(new C());
        C p = al.get(0);
        if (p == null) { }
        al.set(0, p);  //NB! Kun posisjoner i bruk!
        len = al.size();

        for (int i = 0;  i < al.size();  i++) {
            al.get(i).p();
        }
```

HashMap

```
import java.util.HashMap;


        HashMap<String,C> hm = new HashMap<>();
        hm.put("Min", new C());
        String id = "Din";
        C q = hm.get(id);
        if (q == null) { }
        len = hm.size();

        for (String s: hm.keySet()) {
            hm.get(s).p();
        }
```

Da kan vi skissere hovedprogrammet main:

```
if (arg.length != 2) {
    System.out.println("Usage: java Tbaneplanlegger fra-stasjon til-stasjon");
    System.exit(1);
}

if (! stasjonsliste.finnes(arg[0])) {
    System.out.println("Stasjonen " + arg[0] + " finnes ikke!");
    System.exit(2);
}
Stasjon fraStasjon = stasjonsliste.finn(arg[0]);

if (! stasjonsliste.finnes(arg[1])) {
    System.out.println("Stasjonen " + arg[1] + " finnes ikke!");
    System.exit(2);
}
Stasjon tilStasjon = stasjonsliste.finn(arg[1]);

if (fraStasjon.finnDirekteReise(tilStasjon)) {
    System.out.println("Det er direkte reise fra " + fraStasjon +
        " til " + tilStasjon);
} else {
    Stasjon overgang = fraStasjon.finnReiseMedOvergang(tilStasjon);
    if (overgang != null) {
        System.out.println("Du kan komme fra " + fraStasjon +
            " til " + tilStasjon + " ved bytte paa " + overgang);
    } else {
        System.out.println("Ingen reise fra " + fraStasjon +
            " til " + tilStasjon);
    }
}
```

Forbedringer

Denne løsningen kan forbedres på minst to måter:

- Programmer forteller ikke hvilken linje du skal ta.
- Programmet finner ikke alltid korteste vei, og kan svare:

Dette overlates til dere.