Obligatorisk oppgave 1 INF1010 2013

Versjon 1.6 — 23.1.13 Rettet en uklarhet...

Du skal skrive et javaprogram med fire personobjekter. Disse fire objektene representerer inf1010-studentene Emil, Lisa, Ramzi og deg selv (bruk gjerne ditt eget navn). Objektene skal alle være objekter/instanser av klassen Person. En person har et navn og noen pekere som kan peke på andre personobjekter slik at vi kan lage et forenklet «personnettverk».

```
class Person {

    private String navn;
    private Person [] kjenner;
    private Person likerikke; // = uvenn
    private Person forelsketi;

    Person(String n, int lengde) {
      ...
    }

    public String hentNavn() { return navn; }

    // Andre metoder som er tilgjengelige
    // utenfra, se nedenfor

}
```

Når et personobjekt opprettes, sender vi med en tekststreng som er navnet til personen objektet representerer og et heltall (int lengde) som bestemmer lengden til kjenner-arrayen som parametre til konstruktøren.

Du skal programmere ferdig klassen ved å fullføre konstruktøren og metodene som skal være tilgjengelig utenfra. Noen få metoder er gitt ferdig programmert. Du må gjerne lage flere hvis du ønsker. Alle metoder som skal være tilgjengelig utenfra skal deklareres public. Trenger du ekstra hjelpevariable i klassen, må du gjerne legge dem til. Disse skal også være deklarert som private.

Nedenfor tenker vi oss virkningen av metodene definert for en konkret person («inne i» personobjektet), nedenfor **i objektet med navnet «Lisa»**. Parameteren p peker også på et personobjekt. Klassen Person skal ha følgende metoder her forklart med signaturer og virkning:

```
public boolean erKjentMed(Person p)
Sann hvis Lisa kjenner p.

public void blirKjentMed(Person p)
Lisa blir kjent med p, bortsett fra hvis p peker
på Lisa (Lisa kan ikke være kjent med seg selv).

public void blirForelsketI(Person p)
Lisa blir forelsket i p, bortsett fra hvis p peker på Lisa

public void blirUvennMed(Person p)
Lisa blir uvenn med p, bortsett fra hvis p peker på Lisa
```

```
public boolean erVennMed(Person p)
sann hvis Lisa kjenner p og ikke er uvenner med p

public void blirVennMed(Person p)
samme virkning som blirKjentMed(p), men hvis Lisa ikke
liker p (likerikke == p) blir likerikke satt til null.

public void skrivUtVenner()
skriver ut navnet på dem Lisa kjenner,
unntatt den hun ikke liker.

public .... hentBestevenn() // returtypen skal du bestemme
returnerer en peker til Lisas bestevenn (se nedenfor)

public .... hentKjenninger()
returnerer en array som peker på alle Lisas kjente

// Disse to metodene trenger du ikke lage selv
public void skrivUtKjenninger() {
    for (Person p: kjenner)
        if ( p!=null) System.out.print(p.hentNavn() + " ");
    System.out.println("");
}

public void skrivUtAlt() {
    System.out.print(navn + " kjenner: ");
    skrivUtKjenninger();
    if (forelsketi != null)
        System.out.println(navn +
            " er forelsket i " + forelsketi.hentNavn());
    if (likerikke != null)
        System.out.println(navn +
            " liker ikke " + likerikke.hentNavn());
}
```

Du skal også skrive en klasse som oppretter datastrukturen. I denne klassen lages de fire objektene, pekt på av variablene jeg, emil, lisa og ramzi. Sørg deretter for at følgende *tilstandspåstander* holder i datastrukturen, ved å kalle på metodene du laget i personklassen:

Tilstandspåstander

- Du kjenner de tre andre og er ikke forelsket i noen og liker alle.
- Emil kjenner deg og Ramzi. Han er forelsket i deg. Han liker ikke Lisa.
- Lisa kjenner Emil og Ramzi. Hun er forelsket i Ramzi og liker ikke deg.
- Ramzi kjenner alle i rekkefølgen Ego, Emil, Lisa. Han liker ikke Emil og er forelsket i Lisa.

Kall på skrivUtAlt i alle fire objekter i rekkefølgen jeg, emil, lisa og ramzi, skal da gi utskriften (Ego er navnet på objektet som representerer deg):

```
Ego kjenner: Emil Lisa Ramzi
Emil kjenner: Ego Ramzi
Emil er forelsket i Ego
Emil liker ikke Lisa
Lisa kjenner: Emil Ramzi
Lisa er forelsket i Ramzi
Lisa liker ikke Ego
Ramzi kjenner: Ego Emil Lisa
Ramzi er forelsket i Lisa
Ramzi liker ikke Emil
```

Du trenger ikke alle metodene i Person for å få til denne tilstanden.

Vanligere enn å beskrive én spesiell tilstand i datastrukturen, er det å oppgi generelle regler som gjelder for det datastrukturen beskriver. Slike regler kan formuleres som *invariante* (uforanderlige) tilstandspåstander om datastrukturen. Eksempler på slike påstander er f.eks. at en person ikke kan kjenne en annen person mer enn en gang, eller at man ikke kan være forelsket i eller mislike seg selv osv. Ofte er det regler som gjør at datastrukturen stemmer overens med virkeligheten. Tilstandspåstander er til stor hjelp når vi skal programmere og vi vil komme tilbake til dem i senere oppgaver. **Du skal ikke ta hensyn til slike regler i denne oppgaven.**

Lag en metode i Person som skal hete hentBestevenn(). Denne metoden kan brukes slik for å få tak i en peker til Lisas bestevenn:

```
Person lisasBestevenn = lisa.hentBestevenn();
```

En persons bestevenn er for enkelhets skyld definert til å være det objektet som pekes på av kjennerarrayens indeks 0.

Til slutt skal du lage en metode hentKjenninger som skal returnere en array som peker på alle personer som personen kjenner. Arrayen skal være akkurat så lang at lengden er lik antallet kjenninger, og rekkefølgen skal være den samme.

Følgende programkode skal med datastrukturen ovenfor gi utskriften Ego

```
Person [] lisaKjenner = lisa.hentKjenninger();
System.out.println(lisaKjenner[1].hentBestevenn().hentNavn());
```

Oppsummering

Du skal fullføre alle metodene nevnt over i klassen Person. Du skal skrive en klasse som oppretter datastrukturen med de 4 personobjektene slik at datastrukturen tilfredsstiller tilstandspåstandene. Det hele startes i en klasse med main-metoden som du må skrive. De tre klassene skal du sette sammen til ett program som skal gi utskriften ovenfor når det kjøres. Etter denne utskriften kan du, hvis du vil, teste at andre metoder fungerer slik du mener de skal.

Spørsmål til oppgaven med svar

En student spør: «Skal relasjonen mellom instansene av Person-klassen anses til å være symetrisk? Kan/skal vi gjøre det slik at Ego kjenner Lisa impliserer at Lisa også kjenner Ego? Og Ego er forelsket i Lisa impliserer at Lisa også er forelsket i Ego. Fordi, hvis ikke så vil det at Lisa er uvenn med Ego

ikke implisere at Ego også er uvenn med Lisa, og da vil Ego.erVennMed(Lisa) være lik true, og det vil føre til at Ego.bliVennMed(Lisa) gå an, til tross for deres vennskap som egentlig ikke eksisterer. Kan vi løse oppgaven slik at relasjonene blir symetriske der de bør være, eller er det 'ikke lov'.?»

Svar: Dette henger sammen med det som kalles invariante tilstandspåstander i oppgaven. Hvilke regler (fra virkeligheten) vil vi skal reflekteres i programmet (datamodellen av relasjoner mellom 4 personer). I obligen skal du ikke ta hensyn til dette. Tilstanden i datastrukturen skal være akkurat slik som utskriften sier. At dette er unaturlig eller kunstig, gjør ikke noe.

Hvis du vil ha «symmetriske» relasjoner må du gjerne gjøre det, gjør det da i en kopi av programmet som du ikke leverer. Å formulere slike regler er ikke lett. F.eks. vil utsagn som «Lisa og Emil er venner» vanligvis innebære kjennskap begge veier, mens utsagn som «Emil er forelsket i deg» ikke nødvendigvis betyr at du er forelsket i ham eller kjenner ham. Noen av oss opplever dessverre også at noen vi trodde var en venn, viste seg å ikke være det. Å lage regler som treffer virkeligheten 100% er (nesten?) umulig.

Vi vil i senere obliger lage «vennskapsnettverk» som modellerer virkeligheten bedre.