Obligatorisk oppgave 2 INF1010 2013

Versjon 1.2 — 29.1.13 Endelig versjon. Bruker equals() istedet for == for sammenligning av tekststrenger.

Viktig for å løse denne oppgaven: pekertyper, metoder og metodekall, signatur og returverdier (særlig av pekertyper) for metoder, innkapsling (private/public), klasser og konstruktører. Nytt i denne obligen er generiske klasser. Les Stein Gjessings notat Enkle generiske klasser i Java*.*

Personklassen fra oblig 1 forenkles:

```
class Person {
    private String navn;
    private Person bestevenn;

    // konstruktør

    public String hentNavn()
    // returnerer navnet til denne

    public void blirVennMed(Person p)
    // denne blir venn med p

    public Person hentBestevenn()
    // returnerer bestevennen til denne

    public String minBesteVennHeter()
    // returnerer navnet til dennes bestevenn
    // returnerer "ingen" hvis ingen

    public void skrivUtMeg() {
        System.out.println(navn +
            "␣er␣venn␣med␣" + minBesteVennHeter());
    }

    public void skrivUtAllt() {
        skrivUtMeg();
        if (hentBestevenn() != null) hentBestevenn().skrivUtAllt();
    }
}
```

Oppgave a. Skriv ferdig klassen Person**.** Utskriftsmetodene trenger du ikke skrive selv.

For å teste at personklassen er korrekt skrevet, skal du lage et testprogram i en egen klasse som oppretter de fire personene som pekes på av variablene jeg, emil, lisa og ramzi. Etter at personene er opprettet, skal du lage en vennskapsstuktur og teste den ved å utføre kallene:

```
jeg.blirVennMed(lisa);
lisa.blirVennMed(ramzi);
ramzi.blirVennMed(emil);
jeg.skrivUtAllt();
```

Utskriften skal da se slik ut:

```
Ego er venn med Lisa
Lisa er venn med Ramzi
Ramzi er venn med Emil
Emil er venn med ingen
```

Oppgave b. Skriv testklassen. Ikke gå videre før du har fått til dette.

Du og dine tre venner samler på bøker. Du er særlig glad i poesibøker. Lisa som liker mordmysterier er bare interessert i krimbøker. Emil som er en kjent skiskytter samler kun på sportsbøker og Ramzi som elsker mat samler bare på kokebøker. En bok representeres i programmet som et objekt av klassen Bok.

I denne oppgaven har en bok kun én egenskap, kategorien boka er klassifisert i, tatt vare på i variabelen String kat. Verdien til kat (kategori) er synlig utenfor bokobjektet gjennom metoden

```
public String kategori()
```

Oppgave c. Skriv klassen Bok. Konstruktøren skal ha verdien til kat som parameter.

Et personobjekt har en privat variabel bokKat som forteller hva slags bøker denne personen er interesert og en metode

```
private boolean mittInteressefelt(Bok b)
```

for å sjekke dette.

Programmer denne metoden i Person med nødvendige skjulte attributter for å holde rede på den enkeltes interessefelt. Skriv også om konstruktøren slik at uttrykket

```
Person emil = new Person("Emil", "sports");
```

oppretter personobjektet som representerer Emil. Emils mittInteressefelt returnerer da true hvis og bare hvis bokKat.equals(b.kategori()) returnerer true.

Så skal du skrive et testprogram som oppretter og tilbyr disse bøkene:

```
vilNoenHaBoka(new Bok("Java"));
vilNoenHaBoka(new Bok("krim"));
vilNoenHaBoka(new Bok("historie"));
vilNoenHaBoka(new Bok("sports"));
vilNoenHaBoka(new Bok("poesi"));
vilNoenHaBoka(new Bok("sports"));
vilNoenHaBoka(new Bok("poesi"));
vilNoenHaBoka(new Bok("baby"));
vilNoenHaBoka(new Bok("poesi"));
```

Metoden vilNoenHaBoka spør de fire personene etter tur om de vil ha boka. Spør først Emil, så Ramzi, deretter Lisa og til slutt Ego om de vil beholde boka. Dette gjøres ved å kalle på personklassemetoden f.eks. slik:

```
Bok bok = jeg.vilJegHaBoka(b);
```

Etter kallet vil bok==null bety at personen beholdt boka (og man trenger ikke spørre de andre).

For å ta vare på bøker har alle personene en beholder for bøker av type Beholder<Bok>. Beholderen skal hete mittBibliotek. Klassen Beholder<T> kan du lage helt lik klassen StorBeholder<T> på side 8 i Stein Gjessings

notat *Enkle generiske klasser i Java*. Utvid beholderklassen med en public int hentAntall() som returnerer antall objekter i beholderen.

Oppgave d. Skriv klassen Beholder<T>.

I personklassen utvider vi skrivUtMeg:

```
public void skrivUtMeg() {
    System.out.print(navn + "␣er␣venn␣med␣"  + minBesteVennHeter());
    System.out.println(navn + "␣liker␣" + bokKat
        + "bøker␣og␣har␣" + mittBibliotek.hentAntall() + "␣av␣dem.");
}
```

Oppgave e. Skriv ferdig programmet. Skriv vilNoenHaBoka i testklassen.

I Person skriver du om konstruktøren (med bokkategori). Legg til en peker til en bokbeholder som skal hete mittBibliotek og pekervariabelen bokKat. Programmer metodene mittInteressefelt og vilJegHaBoka.

Med vilNoenHaBoka-kallene ovenfor etterfulgt av et kall på jeg.skrivUtAllt, skal programmet gi utskriften:

```
Ego er venn med Lisa
Ego liker poesibøker og har 3 av dem.
Lisa er venn med Ramzi
Lisa liker krimbøker og har 1 av dem.
Ramzi er venn med Emil
Ramzi liker matbøker og har 0 av dem.
Emil er venn med ingen
Emil liker sportsbøker og har 2 av dem.
```

Obligatorisk oppgave 2 slutt.

Alternativ måte å lage vilJegHaBoka på: I stedet for at vilNoenHaBoka spør alle, kaller den på jeg.vilJegHaBoka(). Lag da vilJegHaBoka slik at før den sjekker om boka er interessant for meg, spør bestevennen om hun/han vil ha boka ved å kalle på vilJegHaBoka i objektet som hentBestevenn returnerer. Hvis bestevennen har en bestevenn vil dennes vilJegHaBoka bli kalt osv. Slik fortsetter det til metoden er kalt i Emil som ikke har noen bestevenn. Returverdien skal være boka hvis personen ikke er interessert i den, null hvis personen selv eller bestevennen har tatt vare på den. Når jeg gjør dette kallet, er det altså (p.g.a. datastrukturen) Lisa som får tilbud om boka. Lisa bør så i sin tur først spørre Ramzi om han vil ha den, og Ramzi spør Emil (som ikke har noen bestevenn). Hvis Emil ikke vil ha boka, kommer den (gjennom returkallet) tilbake til Ramzi, som vurderer om han vil beholde den. Slik fortsetter det inntil vi er tilbake i jeg-objektet som, hvis boka kommer i retur, vurderer om den skal beholdes. Programmert slik, blir vilJegHaBoka en *rekursiv metode*. Vi kommer tilbake til det senere i INF1010, men vi tar det med her som en utfordring til dem som har lyst.