Oppgave 1 - Klasser og objekter

Følgende klasseskjelett er gitt:

```
class Kaningaard {

    private Kanin [] kaniner = new Kanin[100];

    public boolean full() { . . . }
    public boolean tom () { . . . }
    public Kanin finnEn(String navn) { . . . }
    public void settInn (Kanin kanin) { . . . }
    public void fjern(String navn) { . . . }
}
```

a. Lag en klasse Kanin. Kaninen har et navn som angis ved opprettelse. I tillegg har klassen en metode, hentNavn(), som returnerer navnet til kaninen.

b. Skriv ferdig klassen Kaningaard ved å fylle ut metodene i klasseskjelettet gitt ovenfor. Du kan gå ut fra at alle kaniner skal ha forskjellige navn. Ta derfor høyde for at det ikke er lov å sette inn en kanin to ganger (altså med samme navn som en i arrayen). Ta også høyde for noen andre problemer som kan oppstå og gi gode feilmeldinger. To eksempler på problemer er dersom man forsøker å sette inn en kanin i en full kaningård, eller når man forsøker å fjerne en kanin som ikke er i listen. Du kan anta at klassen Kanin inneholder metoden hentNavn().

c. Skriv et hovedprogram der du gjør minst tre tester av de forskjellige delene av klassen Kaningaard. Test for eksempel tilfeller som innsetting av kanin ved full kaningård eller fjerning av en ikke-eksisterende kanin.

d. Tegn datastrukturen slik den ser ut etter at du har kjørt hovedprogrammet. Hvorfor kan datastrukturtegninger være nyttige? Diskutér gjerne med en medstudent.

Oppgave 2 - Filinnlesning, objekter og arv

a. Opprett en fil kalt *ord.txt,* som inneholder ett ord på hver linje. Lag et program som skriver ut alle linjene slått sammen til én setning/linje. Legg til et mellomrom etter hvert ord som blir lagt til.

Eksempel på ord.txt

```
Dette
er
en
setning
paa
flere
linjer.
```

b. Utvid programmet i oppgave a, slik at brukeren kan skrive inn navnet på filen som skal leses ved oppstart av programmet.

c. Når brukeren skriver inn feil filnavn, vil det kastes et unntak. Brukeren skal få en passende beskjed når feil filnavn oppgis. Deretter skal programmet avsluttes.
d. For hvert ord som leses inn fra filen skal du nå opprette et Ord-objekt samt kalle på en instansmetode som skriver ut ordet.

Oppgave 3 - Array, ArrayList og løkker

a. Lag et program med en array med plass til 10 heltall. Fyll arrayen med alle sifrene mellom 0 og 10 i stigende rekkefølge. Skriv ut alle verdiene på hver sin linje.
b. Endre programmet slik at du får utskriften i motsatt rekkefølge. Denne gangen skal du bruke en ArrayList i stedet for en array.
c. Endre programmet fra a slik at utskriften nå er tallene i arrayen eller ArrayList-en og summen av alle tallene som er skrevet ut til nå. Denne gangen skal du bruke en “for each”-løkke.
d. Endre programmet slik at tallet blir gjort om til en streng og at tallene blir lagt til i en string som samler alle tallene per iterasjon. Her velger du om du vil ta utgangspunkt i arrayen eller ArrayList-en, og du velger mellom vanlig for-løkke og “for each”-løkke.

Utskrift:

```
> 0
> 01
> 012
> 0123
> 01234
> 012345
> 0123456
> 01234567
> 012345678
> 0123456789
```

e. Lag et program som oppretter en array med plass til 10 strenger. Fyll arrayen med *, hvor plass/indeks 0 har én * og hvor plass/indeks 9 har ti *-er. Det vil si at hver plass inneholder (plassnummer + 1) antall *-er. Skriv ut alle strengene på separate linjer.

Utskriften:

```
> *
> **
> ***
> ****
> *****
> ******
> *******
> ********
> *********
> **********
```

Oppgave 4 - HashMaps
Du har fått i oppdrag av spionbyrået MI6 å lage et program for å holde oversikt over alle agentene og deres kodenavn.

a. Opprett en klasse Agenter. Klassen har en instansvariabel *kodenavnene* som er en HashMap. Denne skal inneholde kodenavn (nøkkel) og ekte navn (verdi) på agentene (begge er String). Konstruktøren til klassen inneholder ingen parametre.

b. Opprett metoden *leggTilAgent* der agentens ekte navn og kodenavn gis som argumenter. Begge er av datatypen String. Legg til disse i hashmap-en *kodenavnene*.

c. Opprett metoden *finnAgent* som tar et kodenavn som argument (String). Dersom kodenavnet finnes i *kodenavnene* skal metoden returnere det ekte navnet. Dersom det ikke finnes, skal metoden returnere null.

d. Opprett metoden *fjernAgent*, som tar et kodenavn som argument og fjerner kodenavn og dets ekte navn fra *kodenavnene*, hvis de finnes.

e. En agent døde på oppdrag:(Noen må derfor ta over kodenavnet til den døde agenten. Skriv metoden *oppdaterKodenavn* som tar et kodenavn og et ekte navn som argumenter (String), og legger det ekte navnet som ny verdi til kodenavnet i *kodenavnene*.

f. Noen klarte å bryte seg inn i *kodenavnene* og vet nå identiteten til alle agentene! Du får derfor jobben med å kryptere alle de ekte navnene i *kodenavnene*. Skriv metoden *krypterKodenavn* som går gjennom *kodenavnene* og oppdaterer de ekte navnene (verdiene) slik at annenhver bokstav erstattes med tegnet *. F.eks. blir navnet James Bond til J*m*s*B*n*.

g. Lag en metode *skrivUt* som skriver ut alle kodenavn og ekte navn i *kodenavnene* på en fin måte.

h. Skriv til slutt et hovedprogram! Opprett et objekt av klassen Agenter, og legg til fire agenter i *kodenavnene*. Test så de ulike metodene og sjekk at de fungerer som de skal.