Løsningsforslag ukeoppg. 9: 19. - 25. okt (INF1000 - Høst 2011)

HashMap, innstikksortering, javadoc *(kap. 9.1 - 9.11, m.m. i "Rett på Java"* ***3. utg.****)*

> ***NB!*** *Legg merke til at disse er* ***løsningsforslag****. Løsningene dine trenger ikke å være like med disse forslag for å være riktige. Det er vanlig i programmering at samme oppgave kan løses på mange vidt forskjellige måter, og alle fremgangsmåter er ok i INF1000 så lenge de leder fram til riktig resultat og oppfyller kravene som står i oppgaveteksten.*

Mål
Forstå forskjellene mellom arrayer og HashMaper, og anvendelsesområdene for disse datastrukturene, og forstå hvordan innstikksortering fungerer.

🔑 Oppgave merket med nøkkelsymbol er plukket ut som spesielt representativ for de viktigste temaene fra ukens forelesning, og alle bør ha som minimumsmål å løse denne selvstendig.

Oppgaver

For de som ønsker å lese en annerledes og fin forklaring av pensum anbefaler vi Marit Nybakkens notater.

1. **HashMap: Hva skrives ut?** *(Se oversikten på side 190 i læreboka)*

```
import java.util.*;
class Personer {
    public static void main(String[] args) {
        HashMap <String, Person> register = new HashMap <String, Person> ();

        Person p1 = new Person("Ida", 19);
        Person p2 = new Person("Lars", 21);

        register.put(p1.navn, p1);
        register.put(p2.navn, p2);

// a)
        Person p = register.get("Ida");
        System.out.println(p.navn + p.alder);

// b)
        for (String s : register.keySet()) {
            System.out.println(s);
        }
// c)
        p1.alder = 24;
        for (Person p3 : register.values()) {
            System.out.println(p3.navn + ":" + p3.alder);
        }
// d)
        if (register.containsValue(p2) && ! register.containsKey("Elin")) {
            System.out.println(true);
        }
// e)
        register.remove("Lars");
        System.out.println(register.size() + " - " + register.isEmpty());
// f)
        System.out.println(register.remove("Ida") == null);
        System.out.println(register.remove("Ida") == null);
    }
}

class Person {
    String navn;
    int alder;

    Person(String navn, int alder) {
        this.navn = navn;
        this.alder = alder;
    }
}
```

```
a)  Ida19
b)  Lars
    Ida
c)  Lars:21
    Ida:24
d)  true
e)  1 - false
f)  false
    true
```

2. **Bank.java: Array vs. HashMap**
(a) Følgende program viser et enkelt banksystem med en array `kontoer[]`, og metoder for å finne en konto vha. navn til eieren og vha. kontonummer. Skriv om programmet slik at det bruker en HashMap i stedet for arrayen `kontoer[]`. I første omgang lager vi én HashMap, med personnavn som nøkkel og et Konto-objekt som verdi, deklarert slik:

```
HashMap<String, Konto> kontoFraNavn = new HashMap<String, Konto>();
```

Hvilke fordeler og ulemper får vi av å bruke HashMap her? Hva kan variabelen `antKontoer` erstattes med i programmet? (Anta foreløpig at personnavnene er unike og at hver person bare kan ha én konto i banken.)

```
class Konto {
    int nr; // kontonummer
    String navn; // eier
    int saldo;

    Konto(int nr, String navn, int saldo) {
        this.nr = nr;
        this.navn = navn;
        this.saldo = saldo;
    }

    void settInn(int innskudd) {
        saldo = saldo + innskudd;
    }
}

class Bank {
    Konto[] kontoer = new Konto[1000];
    int antKontoer = 0;

    public static void main(String[] args) {
        Bank b = new Bank();
    }

    Bank() {
        åpneNyttKonto(530010, "Nils", 4000);
        åpneNyttKonto(720020, "Elin", 8000);
        åpneNyttKonto(910030, "Tina", 9000);

        Konto k = finnKontoFraNavn("Elin");
        System.out.println("Elins kontonr: " + k.nr + ", saldo: " + k.saldo);

        k = finnKontoFraNr(530010);
        System.out.println("Kontonr. " + k.nr + " tilhører " + k.navn);
    }

    void åpneNyttKonto(int nr, String navn, int saldo) {
        Konto k = new Konto(nr, navn, saldo);
        kontoer[antKontoer] = k;
        antKontoer++;
    }

    Konto finnKontoFraNavn(String navn) {
        for (int i = 0; i < antKontoer; i++) {
            if (kontoer[i].navn.equals(navn)) {
                return kontoer[i];
            }
        }
        return null;
    }

    Konto finnKontoFraNr(int kontonr) {
        for (int i = 0; i < antKontoer; i++) {
            if (kontoer[i].nr == kontonr) {
                return kontoer[i];
            }
        }
        return null;
    }
}

KJØREEKSEMPEL:
Elins kontonr: 720020, saldo: 8000
Kontonr. 530010 tilhører Nils
```

```
import java.util.*;

class Konto {
    int nr; // kontonummer
    String navn; // eier
    int saldo;

    Konto(int nr, String navn, int saldo) {
```

```
        this.nr = nr;
        this.navn = navn;
        this.saldo = saldo;
    }

    void settInn(int innskudd) {
        saldo = saldo + innskudd;
    }
}

class Bank2 {
    HashMap <String, Konto> kontoFraNavn = new HashMap <String, Konto>();
    HashMap <String, Konto> kontoer = new HashMap <String, Konto>();

    public static void main(String[] args) {
        Bank2 b = new Bank2();
    }

    Bank2() {
        åpneNyttKonto(530010, "Nils", 4000);
        åpneNyttKonto(720020, "Elin", 8000);
        åpneNyttKonto(910030, "Tina", 9000);

        Konto k = finnKontoFraNavn("Elin");
        System.out.println("Elins kontonr: " + k.nr + ", saldo: " + k.saldo);

        k = finnKontoFraNr(530010);
        System.out.println("Kontonr. " + k.nr + " tilhører " + k.navn);

        System.out.println("Antall kontoer: " + kontoer.size());
        avsluttKonto(k);
        System.out.println("Fins kontoen nå?: " + kontoer.containsValue(k));
        System.out.println("Antall kontoer nå: " + kontoer.size());
    }

    void åpneNyttKonto(int nr, String navn, int saldo) {
        Konto k = new Konto(nr, navn, saldo);
        kontoFraNavn.put(navn, k);
        kontoer.put("" + nr, k);
    }

    Konto finnKontoFraNavn(String navn) {
        return kontoFraNavn.get(navn);
    }

    Konto finnKontoFraNr(int kontonr) {
        return kontoer.get("" + kontonr);
    }

    void avsluttKonto(Konto k) {
        kontoFraNavn.remove("" + k.nr);
        kontoer.remove("" + k.nr);
    }
}

KJØREEKSEMPEL:
Elins kontonr: 720020, saldo: 8000
Kontonr. 530010 tilhører Nils
Antall kontoer: 3
Fins kontoen nå?: false
Antall kontoer nå: 2
```

(b) Lag en HashMap til, kalt `kontoer`, hvor du bruker som nøkkel kontonummeret konvertert til `String`, og fortsatt Konto-objektene som verdi. Vis at metoden `finnKontoFraNr()` blir enklere nå. Videre tenk deg at vi skal ha en metode for å fjerne en konto. Følgende kode viser hvordan det kan gjøres med arrayer. Hvor mange programsetninger trengs det når vi bruker én HashMap i stedet? Og med to?

```
    void avsluttKonto(Konto k) {
        // Fjerner en konto ved å finne indeksen til kontoen i arrayen
        // kontoer[] og flytte alle kontoene med høyere indeks en plass ned.
        boolean funnet = false;
        for (int i = 0; i < antKontoer && !funnet; i++) {
            if (kontoer[i] == k) {
                funnet = true;
                for (int j = i; j < antKontoer - 1; j++) {
                    kontoer[j] = kontoer[j + 1];
                }
                antKontoer--;
            }
        }
    }
```

(Løsningsforslaget står i del (a) over)

(c) Disse oppgavene har begrensningen at personnavnene må være unike og at hver person bare kan ha

én konto i banken. Hvordan ville man unngått disse begrensninger i et mer avansert system? Hvilke fordeler og ulemper ser du av å bruke HashMap-er i stedet for 2D-arryaer i Oblig 3? (foreslå mulige nøkkel/verdi-kombinasjoner).
Hint: *Se avsnitt 9.11 på side 191 i læreboka for forskjellene mellom arrayer og HashMap-er.*

> I store systemer passer det bedre å ha personnummer som nøkkel eller noe annet som er unikt for hver verdi, i stedet for navn alene. Verdi kan være et Person-objekt, som i sin tur kan ha en objektvariabel av type HashMap som refererer til alle kontoene for personen. Hvis programmet trenger å finne personer ut fra navn så kan man likevel bruke navn som nøkkel i en tilleggs-HashMap, som f.eks. kan ha som verdi en HashMap over personene med det navnet.
>
> **Arrayer** passer best å bruke når maks. antall elementer er kjent på forhånd, når vi trenger en gitt rekkefølge på dataene, og når indeksene er numeriske. **HashMap** er bedre når maks. antall elementer er ukjent, eller man trenger å slå opp i dataene vha. "indekser" som er noe annet enn tall (f.eks. tekster), og når rekkefølgen på dataene ikke er avgjørende. Med HashMap må dataene være objekter (pekere), ikke primitive typer, men det finnes objekt-typer for alle de primitive: Integer, Double, Character, Boolean.
>
> I **Oblig 3** er det mest naturlig å bruke arrayer, men man kan også bruke HashMap: da kan f.eks. hele hybelnavnet være nøkkel. Man trenger likevel lett tilgang til etasje- og rom-nr. så disse kan evt. legges til som objektvariabler i Hybel. En annen fremgangsmåte er å ha Etasje som egen klasse og da kan denne ha en array (eller HashMap) med hyblene i etasjen.

3. **HashMap med akronymer:** *kap. 9, oppg. 2 (side 195)*
(a) Et akronym er en bokstavkombinasjon som uttrykker noe mer fullstending for eksempel FN = Forente nasjoner. Lag et program som leser inn en rekke akronymer og deres tolkning fra fil (du finner en fil med akronymer på bokas hjemmeside: akronymer.txt). Legg akronymene og tolkningene i en HashMap med akronymet som nøkkel.

   Brukeren skal oppgi et akronym til programmet og få tilbake en eller flere tolkninger av akronymet. Det kan maksimalt være 10 tolkninger til hvert akronym. Denne brukerdialogen skal gå i løkke til brukeren svarer "-".

   Tips: Lag en klasse der objekter har følgende attributter: akronymet og en String-array med tolkninger, samt antall tolkninger. Pekere til objekter av denne klassen legges i HashMap med akronymet som nøkkel. Det kan være lurt å lage en metode i denne klassen som legger til en ny tolkning, og som holder orden på hvor mange tolkninger det er i øyeblikket.

```
API     Application Programming Interface
AV      Audio/Video
AV      Authenticity Verification
BASH    Bourne Again Shell [Unix]
EU      Europaunionen
FN      Forente nasjoner
...osv...
```

```
// Basert på løsningsforslaget i lærebokens hjemmeside.
/*
* Løsning på oppgave 2 i kapittel 9 : Akronymer
* Programmet legges i en fil med navnet : AkronymerMain.java
*
* Løsningen bygger på to klasser som begge fins i denne filen
* Programmet forutsetter at alle akronymene står med store bokstaver
* i datafilen, men brukeren trenger ikke taste de inn med store bokstaver.
*/

import java.util.*; // for å få med HashMap
import easyIO.*;    // for å få med fil- og brukerkommunikasjon

class AkronymerMain {
    // initialiserer akronym-registeret
    HashMap<String, Akronym> akronymer = new HashMap<String, Akronym>();

    public static void main(String[] args) {
        // Oppretter et objekt av denne samme klassen for å kunne kalle
        // objektmetoder fra denne klassemetoden vha. pekeren "am".
        AkronymerMain am = new AkronymerMain();
```

```
        // leser inn data fra fil og bygger opp register
        am.byggRegister();

        // går i dialog med bruker
        am.brukerdialog();
    }

    void byggRegister() {
        In fil = new In("akronymer.txt");
        while(fil.hasNext()) {

            // leser akronymet og tolkningen
            String akro = fil.inWord();
            String tolkning = fil.inLine().trim();

            // sjekker om akronymet er registrert tidligere
            if (akronymer.containsKey(akro)){

                // vi skal nå legge til en ny tolkning
                Akronym a = akronymer.get(akro);
                a.add(tolkning);

            } else {
                // vi lager et nytt Akronym-objekt
                Akronym ny = new Akronym(akro,tolkning);

                // vi legger det inn i akronym-registeret
                akronymer.put(akro, ny);
            }
        }
        fil.close();
    }

    void brukerdialog() {
        In tastatur = new In();
        String svar;
        do {
            System.out.print("Skriv inn et akronym (avslutt med -): ");
            svar = tastatur.inLine();
            svar = svar.trim();
            svar = svar.toUpperCase();
            if (akronymer.containsKey(svar)) {
               Akronym a = akronymer.get(svar);
               a.skrivUt();
            } else if (!svar.equals("-")){
               System.out.println(svar + " fins ikke i akronymregisteret.");
            }
        } while (!svar.equals("-"));
    }
}

class Akronym {
    int antall;
    String akronym;
    String[] tolkninger = new String[10];

    Akronym (String a, String t) {
        akronym = a;
        tolkninger[antall] = t;
        antall = 1;
    }

    void add(String t) {
        if (antall < 10) {
            tolkninger[antall] = t;
            antall++;
        } else {
            System.out.println("For mange tolkninger av " + akronym);
        }
    }

    void skrivUt() {
       System.out.println(akronym + " kan bety:");
       for (int i = 0; i < antall; i++) {
           System.out.println("   " + tolkninger[i]);
        }
    }
}
```

(b) Utvid programmet slik at bruker får anledning til å legge til en tolkning dersom den ikke finnes fra før.

```
         // Linje 68 i løsningen (a) vist ovenfor:
         System.out.println(svar + " fins ikke i akronymregisteret.");
         // For å løse del (b) trenger vi bare legge til følgende to
         // linjer rett etter linje 68:
         System.out.print("LEGG INN TOLKNING: ");
         akronymer.put(svar, new Akronym(svar, tastatur.inLine()));
```

4. **Innstikksortering av kontoer:** *(læreboka side 95)*
Ta utgangspunkt i følgende program (som sorterer en heltalls-array og en String-array); og lag en metode som sorterer arrayen `kontoer[]` fra oppgave nr. 2 (a) ovenfor alfabetisk på personnavn. Etter et kall på metoden skal f.eks. `kontoer[0]` være Elins konto, `kontoer[1]` Nils sin, osv. Metoden skal ha to inn-parametre: `Konto[] kontoer`, og `int antKontoer`; og skal kunne sortere et vilkårlig antall kontoer. Utvid programmet i nr. 2 (a) med et kall på metoden for å sjekke at den fungerer.

```
class Innstikksort {

    /** Sorterer en heltallsarray. */
    public static void sorter(int[] a) {
        for (int k = 0 ; k < a.length - 1; k++) {

            if (a[k] > a[k + 1]) {
                // a[k + 1] står på feil plass, ta den ut:
                int tmp = a[k + 1];
                int i = k;

                // Skyv a[i] mot høyre ett hakk til vi finner
                // riktig plass til tmp:
                while (i >= 0 && a[i] > tmp) {
                    a[i + 1] = a[i];
                    i--;
                }
                // Sett tmp inn på riktig plass:
                a[i + 1] = tmp;
            }
        }
    }

    /** Sorterer en String-array.  Dette er en redigert utgave av
    metoden ovenfor som jobber med String-array i stedet for int-array. */
    public static void sorter(String[] a) {
        for (int k = 0 ; k < a.length - 1; k++) {
            if (a[k].compareTo(a[k + 1]) > 0 ) {
                String tmp = a[k + 1];
                int i = k;
                while (i >= 0 && (a[i].compareTo(tmp) > 0)) {
                    a[i + 1] = a[i];
                    i--;
                }
                a[i + 1] = tmp;
            }
        }
    }
}

/** Test av innstikksortering */
class TestInnstikksort {
    public static void main(String[] args) {

        int[] a = { 3, 12, 8, 1, 10 };
        Innstikksort.sorter(a);
        for (int i = 0; i < a.length; i++) {
            System.out.println("a[" + i + "] = " + a[i]);
        }

        String[] navn = { "Ola", "Kari", "Arne", "Eli" };
        Innstikksort.sorter(navn);
        for (int i = 0; i < navn.length; i++) {
            System.out.println("navn[" + i + "] = " + navn[i]);
        }
    }
}

KJØREEKSEMPEL:
a[0] = 1
a[1] = 3
a[2] = 8
a[3] = 10
a[4] = 12
navn[0] = Arne
navn[1] = Eli
navn[2] = Kari
navn[3] = Ola
```

```
    /** Sorterer en Konto-array: Koden er basert på sorter(String[] a) men
      * erstatter a[] med enten kontoer[] eller kontoer[].navn avhengig av om
      * man trenger objektet eller sorteringsnavn.  a.length -> antKontoer */
    public static void sorter(Konto[] kontoer, int antKontoer) {
        for (int k = 0 ; k < antKontoer - 1; k++) {
            if (kontoer[k].navn.compareTo(kontoer[k + 1].navn) > 0 ) {
                Konto tmp = kontoer[k + 1];
                int i = k;
                while (i >= 0 && (kontoer[i].navn.compareTo(tmp.navn) > 0)) {
                    kontoer[i + 1] = kontoer[i];
```

```
                    i--;
                }
                kontoer[i + 1] = tmp;
            }
        }
    }
```

Legger også til følgende testkode på slutten av konstruktøren til Bank:

```
        // Kontosortering:
        System.out.println("Sorterer kontoene på navn...");
        sorter(kontoer, antKontoer);
        for (int i = 0; i < antKontoer; i++) {
            System.out.println("kontoer[" + i + "].navn = " + kontoer[i].navn
                               + ", saldo: " + kontoer[i].saldo);
        }

KJØREEKSEMPEL:
Sorterer kontoene på navn...
kontoer[0].navn = Elin, saldo: 8000
kontoer[1].navn = Nils, saldo: 4000
kontoer[2].navn = Tina, saldo: 9000
```

5. **Mer om HashMap-er:** *(som oppgave 4 i kap. 9, side 197)*
   Finn fram en oppgave du har løst vha. arrayer, og bytt ut noen av arrayene med HashMap-er. Hvilke deler av programmet blir enklere nå, evt. vanskeligere? Bruk gjerne din egen Oblig 2 eller 3.

6. **Kursoppmeldingssystemet:** *(kap. 8.17, side 162-174 og 178-180)*
   Hvis du har lyst å jobbe med Kursoppmeldingssystemet vist på side 162-174 i læreboka (kap. 8.17), så kan du laste ned kildekoden her: Studentregister.java. Det er flere oppgaver i læreboka som tar utgangspunkt i dette programmet: Oppgave 9-13 i kap. 9 (side 176-178). Løsningsforslag til tre av disse oppgavene (9, 10, 12) finner du her.

7. **Javadoc:**
   Du kan finne offisielle eksempler på javadoc-kommentarer her og her. Se på noen av disse eksemplene, og skriv lignende kommentarer i din Oblig 3 eller 4. Javadoc-kommentarer startes med `/**` og avsluttes med `*/`, og plasseres i linjen(e) rett før klassen, metoden, eller objektvariabelen man ønsker å kommentere. Kjør deretter `javadoc`-kommandoen, og åpne til slutt den genererte `index.html`-filen i en browser for å se på resultatet:

   ```
   > javadoc -package Programnavn.java
   > firefox index.html &
   ```

8. **Ukens nøtt: Dobbel-sortert utskrift av HashMap** *(veldig vanskelig!)*
   NB! Denne oppgaven er litt kunstig, vanligvis bruker man ikke HashMap som eneste datastruktur for å lagre data som skal sorteres.
   Lag en metode "`void skrivSortert(HashMap <String, Person> register)`" som skriver ut innholdet i HashMap-en fra oppgave nr. 1 ovenfor sortert på alder. Utskriften skal vise alder og navn for personene, og hvis det er flere med samme alder skal disse skrives ut sortert på navn. For å teste metoden lager du et objekt av klassen Personer og kaller metoden din via dette objektet, etter å ha lagt inn 5 personer i HashMap-en: Ida 19, Lars 21, Elin 21, Nils 19, og Anna 19.

Tibakemelding om dette oppgavesettet kan du skrive i bloggen eller sende på mail til josek *[a]* ifi.uio.no