

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

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

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;
    }
}
```

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, deklartert 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:
Elin's kontonr: 720020, saldo: 8000
Kontonr. 530010 tilhører Nils

```

(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--;
        }
    }
}

```

(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.

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...
```

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

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 innparametre: 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

```

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 linje(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, Eli 21, Nils 19, og Anna 19.