Løsningsforslag ukeoppg. 6: 28. sep - 4. okt (INF1000 - Høst 2011)

Løsningsforslag til oppgave 7, 8, og 9 mangler

Klasser og objekter *(kap. 8.1 - 8.14 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
Få et første innblikk i hvordan man programmerer med klasser og objekter.

🔑 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

0. **KursRegister.java:**
(a) Studér følgende program. Finn ut hva som er klassene, objektene, og pekerne i programmet, og hvordan disse er brukt. Undersøk gangen i programmet når det kjøres. Hvorfor skriver det ut forskjellig informasjon om to kurs når det bare er én System.out.println i programmet? Begge klassene lagres i én fil, kalt `KursRegister.java`.

```
class Kurs {
    // Objektvariabler:
    String kode;
    int studiepoeng;

    // Objekt-metode (dvs. uten "static"):
    void skrivUt() {
        System.out.println("Kurs med kode: " + kode
                           + ", og studiepoeng: " + studiepoeng);
    }
}

class KursRegister {

    // Klasse-metode (dvs. med "static"):
    public static void main(String[] args) {

        Kurs inf, mat; // pekere (variabler som kan peke på Kurs-objekter)

        inf = new Kurs(); // Lager et objekt av klassen Kurs
        inf.kode = "INF1000"; // Setter verdier i objektet...
        inf.studiepoeng = 10;
        inf.skrivUt();

        mat = new Kurs(); // Lager et objekt til av klassen Kurs
        mat.kode = "MAT1001";
        mat.skrivUt();
    }
}
```

```
KJØREEKSEMPEL:
> java KursRegister
Kurs med kode: INF1000, og studiepoeng: 10
Kurs med kode: MAT1001, og studiepoeng: 0
```

Programmet skriver ut informasjon om to forskjellige kurs fordi metoden skrivUt() blir kalt to ganger fra klassen KursRegister, og den kalles via forskjellig Kurs-objekt hver gang, ved hjelp av disse setningene i koden ovenfor:

```
        inf.skrivUt();
        ...
        mat.skrivUt();
```

(b) **Mange kurs:** Endre programmet slik at kursene lagres i en **array** av Kurs-pekere, i stedet for enkelt-pekerne `inf` og `mat`. F.eks. skal første element i arrayen være `kurs[0]`, og koden til denne (`kurs[0].kode`) skal være `"INF1000"`. Lag en løkke som oppretter tre kurs-objekter, setter verdien 10 som studiepoeng i alle, og setter kode til kursene ved å ta ett og ett navn fra følgende tilleggsarray:

```
String[] koder = { "INF1000", "MAT1001", "INF1080" };
```

Løkken skal til slutt kalle metoden `skrivUt()` for å skrive ut info om kursene som opprettes.

```
class Kurs {
    // Objektvariabler:
    String kode;
    int studiepoeng;

    // Objekt-metode (dvs. uten "static"):
    void skrivUt() {
        System.out.println("Kurs med kode: " + kode
                           + ", og studiepoeng: " + studiepoeng);
    }
}

class KursRegister {

    // Klasse-metode (dvs. med "static"):
    public static void main(String[] args) {

        Kurs[] kurs = new Kurs[3]; // Array av Kurs-pekere
        String[] koder = { "INF1000", "MAT1001", "INF1080" };

        for (int i = 0; i < kurs.length; i++) {
            kurs[i] = new Kurs(); // Lager et objekt av klassen Kurs
            kurs[i].studiepoeng = 10;  // Setter verdier i objektene...
            kurs[i].kode = koder[i];
            kurs[i].skrivUt();
        }
    }
}

KJØREEKSEMPEL:
> java KursRegister
Kurs med kode: INF1000, og studiepoeng: 10
Kurs med kode: MAT1001, og studiepoeng: 10
Kurs med kode: INF1080, og studiepoeng: 10
```

(c) **Legg til foreleser:** Anta at hvert kurs har én foreleser og at vi bare ønsker å lagre *navnet* på foreleseren i hvert kurs. Utvid programmet med objektvariabelen `String foreleser` i klassen Kurs, og slik at verdien i denne initialiseres med følgende foreleser-navn for de tre kurs, henholdsvis "Ragnhild K.", "Erik L.", og "Roger A." (bruk gjerne en tilleggsarray slik som i del (b) ovenfor). Utvid også metoden skrivUt() slik at det får tak i og skriver ut navnet på foreleseren i hvert kurs.

```
class Kurs {
    // Objektvariabler:
    String kode;
    int studiepoeng;
    String foreleser;

    // Objekt-metode (dvs. uten "static"):
    void skrivUt() {
        System.out.println("Kurs med kode: " + kode
                           + ", og studiepoeng: " + studiepoeng
                           + ". Foreleser: " + foreleser);
    }
}

class KursRegister {

    // Klasse-metode (dvs. med "static"):
    public static void main(String[] args) {

        Kurs[] kurs = new Kurs[3]; // Array av Kurs-pekere
        String[] koder = { "INF1000", "MAT1001", "INF1080" };
        String[] forelesere = { "Ragnhild K.", "Erik L.", "Roger A." };

        for (int i = 0; i < kurs.length; i++) {
            kurs[i] = new Kurs(); // Lager et objekt av klassen Kurs
            kurs[i].studiepoeng = 10;  // Setter verdier i objektene...
            kurs[i].kode = koder[i];
            kurs[i].foreleser = forelesere[i];
            kurs[i].skrivUt();
        }
    }
}

KJØREEKSEMPEL:
> java KursRegister
Kurs med kode: INF1000, og studiepoeng: 10. Foreleser: Ragnhild K.
Kurs med kode: MAT1001, og studiepoeng: 10. Foreleser: Erik L.
Kurs med kode: INF1080, og studiepoeng: 10. Foreleser: Roger A.
```

(d) **Legge til studenter:** Anta at hvert kurs har maks. 200 studenter. Hvordan kan vi legge til informasjon

om studentene i kursene? Undersøk følgende to muligheter: Legge til en array `String[] studenter;` vs. å lage en ny klasse Student med en objektvariabel `navn`. Hvilken av de to variantene vil være mest hensiktsmessig hvis vi vil lagre enda mer informasjon om hver student i systemet (i tillegg til navn)?

> Det er bedre å lage en klasse **Student**, og så lagre informasjon om studentene i objekter av denne klassen, f.eks. ved hjelp av en array `Student[] studenter = new Student[200];` deklarert i klassen Kurs. Fordelen er at man da kan samle forskjellig type informasjon om en student i ett og samme student-objekt.

1. **Oppgave 1** i kapittel 8 (side 176)
   Lag et program hvor `main` ligger i en klasse Prog1, og skriv en annen klasse ABC med et heltall `int i`. Begge klassene skal være i samme fil (`Prog1.java`).

   (a) Deklarer en peker `pek` til ABC-klassen i main.
   (b) Lag et objekt av klassen ABC, og la `pek` peke på det objektet.
   (c) Sett verdien av `i` til 14 i dette objektet.
   (d) Skriv ut på skjermen verdien av `i` vha. en setning i `main`.
   (e) Deklarer en metode `dobbelt()` i ABC som dobler verdien av `i`.
   (f) Kall denne metoden fra `main` to ganger, og skriv så ut fra `main` verdien av `i` i objektet.

```
// Løsningsforslag:
// Følgende legges i fila Prog1.java (både klasse Prog1 og klasse ABC).
class Prog1 {

    public static void main(String[] args) {
        ABC pek; // (a) Deklarerer en peker til ABC

        pek = new ABC(); // (b) Oppretter et objekt av ABC

        pek.i = 14; // (c) Setter en verdi i objektet

        System.out.println("Verdien av i = " + pek.i); // (d)

        // (f):
        pek.dobbelt();
        pek.dobbelt();
        System.out.println("Verdien av i = " + pek.i);
    }
}

class ABC {
    // Variabler:
    int i;

    // Metoder: (e)
    void dobbelt() {
        i = i * 2;
    }
}

KJØREEKSEMPEL:
> javac Prog1.java
> java Prog1
Verdien av i = 14
Verdien av i = 56
```

2. **Oppgave 2** i kapittel 8 (side 176)
   Fjern klassen ABC fra fila `Prog1.java` og legg den på en egen fil, kalt `ABC.java`, i samme mappe.
   (a) Kompilér så `ABC.java` og `Prog1.java` hver for seg.
   (b) Kjør nå `Prog1.java` og se at du får det samme resultat som i Oppgave 1.

> Klassen Prog1 fra forrige oppgave legges alene i filen med navn Prog1.java, og klassen ABC legges i en egen fil med navn ABC.java
> (a) ...så kompilerer man (med javac) hver av disse to filene for seg, og
> (b) ...så kjører man bare Prog1.java (med "java Prog1") til slutt.
>
> Legg merke til at resultatet av dette, med klassene fordelt på to filer, blir akkurat det samme som da man hadde begge klassene samlet i én fil:

```
KJØREEKSEMPEL:
> javac Prog1.java
> javac ABC.java
```

```
> java Prog1
Verdien av i = 14
Verdien av i = 56
```

3. **Oppgave 3** i kapittel 8 (side 176)
(a) Deklarer en objektvariabel `double x` i klassen Prog1.
(b) Lag en objektmetode `double settX(double y)` i klassen Prog1 som setter verdien av `x` til parameterens verdi, og som returnerer den gamle verdien av `x` før den fikk den nye verdien.
(c) Lag en ny metode som tester `settX()` med 10 000 kall i en løkke.

```
class Prog1 {

    double x; // (a)

    public static void main(String[] args) {

        // Hvis man vil legge koden i main()-metoden, som er static
        // så trenger man en peker til denne samme klassen (Prog1)
        // for å kunne bruke globale variabler som x, som ikke er static.
        Prog1 peker = new Prog1();

        // (b):
        peker.x = 123;
        System.out.println("Verdi av x før kall på settX() = " + peker.x);
        double gammelVerdi = peker.settX(456);
        System.out.println("Verdi av x etter kall på settX() = " + peker.x);
        System.out.println("Gammel verdi av x, returnert av settX() = "
                           + gammelVerdi);
        // (c):
        peker.testSettX();
    }

    double settX(double y) { // (b)
        double tmp = x;
        x = y;
        return tmp;
    }

    void testSettX() { // (c)
        double gammelVerdi = 0;
        System.out.println("\nTester settX() 10000 ganger, ved å sette inn "
                           + "verdiene 1 - 10000...");
        for (int i = 1; i <= 10000; i++) {
            gammelVerdi = settX(i);
        }
        System.out.println("Verdi av x etter 10000 kall på settX(): " + x);
        System.out.println("Gammel verdi av x, returnert av siste kall"
                           + "på settX() = " + gammelVerdi);
    }
}

KJØREEKSEMPEL:
> java Prog1
Verdi av x før kall på settX() = 123.0
Verdi av x etter kall på settX() = 456.0
Gammel verdi av x, returnert av settX() = 123.0

Tester settX() 10000 ganger, ved å sette inn verdiene 1 - 10000...
Verdi av x etter 10000 kall på settX(): 10000.0
Gammel verdi av x, returnert av siste kallpå settX() = 9999.0
```

4. **Oppgave 4** i kapittel 8 (side 177)
Utvid klassen Konto med get- og set-metoder for tekstene `eier` og `adresse` (dvs. metoder som kan returnere verdien, og sette nye verdier i disse objektvariabler).

```
class KontoEksempel {

    public static void main(String[] args) {

        Konto k1 = new Konto();
        k1.bestemKontonr();

        k1.settInn(500);
        System.out.println("Saldo er: " + k1.saldo);

        k1.taUt(300);
        System.out.println("Saldo er: " + k1.saldo);
    }
}

class Konto {
    int kontonr;
```

```
    int saldo;
    String eier, adresse;
    double rente = 2.5; // 2.5% per år
    static int nummer = 0; // Klassevariabel

    void bestemKontonr() {
        nummer++;
        kontonr = nummer;
    }

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

    boolean taUt(int uttak) {
        if (uttak > saldo) {
            return false;
        }
        saldo = saldo - uttak;
        return true;
    }

    int getSaldo() {
        return saldo;
    }
}
```

KJØREEKSEMPEL:

```
> java KontoEksempel
Saldo er: 500
Saldo er: 200
```

```
class KontoEksempel {

    public static void main(String[] args) {

        Konto k1 = new Konto();
        k1.bestemKontonr();

        k1.settInn(500);
        System.out.println("Saldo er: " + k1.saldo);

        k1.taUt(300);
        System.out.println("Saldo er: " + k1.saldo);

        k1.settEier("Martin");
        k1.settAdresse("Oslo");
        System.out.println("Eier: " + k1.getEier());
        System.out.println("Adresse: " + k1.getAdresse());
    }
}

class Konto {
    int kontonr;
    int saldo;
    String eier, adresse;
    double rente = 2.5; // 2.5% per år
    static int nummer = 0; // Klassevariabel

    void bestemKontonr() {
        nummer++;
        kontonr = nummer;
    }

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

    boolean taUt(int uttak) {
        if (uttak > saldo) {
            return false;
        }
        saldo = saldo - uttak;
        return true;
    }

    int getSaldo() {
        return saldo;
    }

    String getEier() {
        return eier;
    }
    String getAdresse() {
        return adresse;
    }
    void settEier(String eier) {
        this.eier = eier;
    }
    void settAdresse(String adresse) {
        this.adresse = adresse;
    }
}
```

```
KJØREEKSEMPEL:
> java KontoEksempel
Saldo er: 500
Saldo er: 200
Eier: Martin
Adresse: Oslo
```

5. **Oppgave 5** i kapittel 8 (side 177)
Lag en metode `årsoppgjør()` som legger renten til saldo for et år (i programmet fra oppgave 4). Ikke gjør endringer på saldo direkte, men kall `settInn`-metoden fra årsoppgjørmetoden for å løse problemet.

På slutten av main-metoden:

```
        k1.årsoppgjør();
        System.out.println("Saldo etter årsoppgjør: " + k1.getSaldo());
```

I klassen Konto:

```
    void årsoppgjør() {
        settInn((int) (saldo * rente/100.0));
    }

KJØREEKSEMPEL:
> java KontoEksempel
Saldo er: 500
Saldo er: 200
Eier: Martin
Adresse: Oslo
Saldo etter årsoppgjør: 205
```

6. **Oppgave 6** i kapittel 8 (side 177)
Lag to klasser A og B som har pekere til hverandre, og bruk disse pekerne til å kalle en metode i A fra B, og tilsvarende bruk pekeren i A til å kalle en metode i B. Begge disse metodene skal skrive ut en tekst på skjermen. Kjør programmet og se at du får riktig utskrift.

```
// Alle 3 klassene legges i en fil med navn: Oppg6.java
class Oppg6 {

    public static void main(String [] args) {
        A a1 = new A();
        B b1 = new B();
        a1.pekerTilB = b1;
        b1.pekerTilA = a1;

        a1.metA();

        b1.metB();
    }
}

class A {
    B pekerTilB;

    void metA() {
        System.out.println("A: metA()");
        pekerTilB.skrivUt();
    }

    void skrivUt() {
        System.out.println(" A: skrivUt()");
    }
}

class B {
    A pekerTilA;

    void metB() {
        System.out.println("B: metB()");
        pekerTilA.skrivUt();
    }

    void skrivUt() {
        System.out.println(" B: skrivUt()");
    }
}


KJØREEKSEMPEL:
> java Oppg6
A: metA()
 B: skrivUt()
B: metB()
 A: skrivUt()
```

Repetisjonsoppgaver

7. **Løkker: Gangetabell**
   (a) Lag et program som ber bruker taste inn et tall, og skriver ut gangetabellen for det tallet, ganget med 1, 2, osv. til 12. Hvis bruker taster inn 5 skal resultatet se slik ut:

```
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
...osv...
```

   (b) Endre programmet slik at utskriften av gangetabellen lages i en egen metode. Sett opp programmet med to klasser som vist i oppgave 4 ovenfor.

8. **Tekster: Tallsiffer-oversetting:** *kap. 6, oppg. 7 og 8 (side 118)*
   (a) Lag et program som oversetter fra tallsiffer til tekst slik at f.eks. 3 blir oversatt til "tre". Programmet skal kunne oversette alle 10 sifre (fra 0 til 9). *Hint:* Bruk en array med tekstene "null", "en", "to", osv.
   (b) Lag et program som oversetter fra tekst til tall. Programmet skal be brukeren skrive inn et tall mellom null og ni (med bokstaver), og skrive ut tilsvarende siffer. *Hint:* Bruk arrayen fra del (a).

9. **Array med String-er:**
   Hva blir skrevet ut i følgende program?, og hvorfor?

```
class NavneArray {
   public static void main(String[] args) {

        String[] navn = { "Anne", "Kari", "Ole", null };

//(a)
        System.out.println(navn[1] + navn[navn.length/2]);

//(b)
        for (int i = 0; i < navn.length; i++) {
            // Testen "!= null" sikrer at neste ledd ikke blir null.equals(..)
            if (navn[i] != null
                && (navn[i].equals("Ole") || navn[i].equals("Anne")) ) {
                System.out.println(i);
            }
        }

//(c)
        int indeks = 0;
        boolean funnet = false;
        while (indeks < 4 && !funnet) {
            if (navn[indeks].equals("Kari")) {
                funnet = true;
            }
            indeks++;
        }
        System.out.println(indeks);

//(d)
        String[] andreNavn = { "Per", "Anne", "Ole" };
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 3; j++) {
                // Testen "!= null" sikrer at neste ledd ikke blir null.equals(..)
                if (navn[i] != null && navn[i].equals(andreNavn[j])) {
                    System.out.println(i + " " + j);
                }
            }
        }
   }
}
```

10. **Metode med array som inn-parameter:** *kap. 7, oppg. 3 (side 136)*
    Lag en metode `double gjennomsnitt(int[] a)` som summerer alle elementene i heltallsarrayen `a`, og som returnerer (det aritmetiske) gjennomsnittet av verdiene i `a`.

```
// Dette eksemplet viser en annen måte å sette opp programmet med bare én klasse.
// Vi slipper å bruke to klasser ved å la objekt-pekeren "gj" referere til
```

```
// den samme klassen som den selv og main-metoden er plassert i.
class Gjennomsnitt {

    public static void main(String[] args) {
        Gjennomsnitt gj = new Gjennomsnitt();
        gj.testGjennomsnitt();
    }

    void testGjennomsnitt() {
        int[] a = { 1, 2, 2, 1, 0, 3 };
        double snitt = gjennomsnitt(a); // Argument: arraynavnet uten klammer
        System.out.println("Gjennomsnittet er: " + snitt);
    }

    double gjennomsnitt(int[] a) {
        int sum = 0;
        for (int i = 0; i < a.length; i++) {
            sum += a[i];
        }
        return (double) sum / a.length;
    }
}

KJØREEKSEMPEL:
Gjennomsnittet er: 1.5
```

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