UiO : Universitetet i Oslo

Ukeoppgaver 7: 11. - 15. okt (INF1000 - Høst 2010)

Mer om klasser og objekter *(kap. 8.17 - 8.18),* ***UML*** *(kap. 12.1 - 12.9)*

Mål

Få mer øvelse i bruk av klasser og objekter, en innføring i UML-klassediagrammer, og tips til Oblig 3.

Oppgaver til teoritimen

1. **UML og behandling av obliger:**
   (a) Lag et **UML-klassediagram** for et lite program som kan brukes til å holde orden på obliger og studenter i én INF1000-gruppe. Bruk tre klasser: Student, Obligbesvarelse, og Gruppe. Navngi forholdene og skriv antall på begge sider av disse. Plassér følgende variabler i klassene der de passer best: `navn` på studentene, og en boolean `erGodkjent` som angir status på leverte obliger. Hva slags dataelementer (variabler, arrayer, eller lignende) ville du lagt til for å holde orden på studentene i gruppen og obligbesvarelsene de leverer? Anta at det bare er én oblig i kurset, men at studentene kan levere to utgaver av obligen hvis den første ikke blir godkjent.
   Tips: Se eksemplet på side 236 i læreboka.

   (b) Skriv et **program** for systemet beskrevet i del (a), og lag en testmetode som legger inn 3 studenter i gruppen, og obligbesvarelser for to av dem. Registrér én av disse besvarelsene som godkjent. (Anta som i (a) at kurset bare har én oblig, og at hver student kan levere opptil 2 besvarelser på den). Til slutt i testmetoden kaller du en annen metode i klassen Gruppe, kalt `skrivUt()`, som går gjennom alle studentene som er registrert i gruppen, og for hver av de skriver ut antall obliger de leverte (0, 1, eller 2), og om de fikk godkjent.

   (c) **Utvidelser:** Diskutér hvordan man kan utvide programmet til å kunne håndtere flere oblignr. i kurset, og enda flere besvarelser fra samme student på en gitt oblig. Evt. også flere grupper.

   (d) **Konstruktør og this:** Lag en *konstruktør* i klassen Student, med minst én parameter, navn, og bruk nøkkelordet this for å skille mellom parameteren "navn" og objektvariabelen "navn" i konstruktøren. Skriv kode som benytter denne konstruktøren. *En konstruktør er en metode med samme navn som klassen den er i, og som blir utført automatisk når man oppretter objekter av klassen vha. nøkkelordet* `new`. Husk å endre alle setninger i programmet som inneholdt uttrykket "`new Klassenavn()`", slik at de benytter den nye konstruktøren (dvs. at du legger til en parameter i parentesene etter `new Klassenavn...`).

2. **Lesing/skriving av datafil:**
   Utvid programmet fra oppgave 1. (b) ovenfor med lesing og skriving av dataene til en datafil. Finn på en passende måte å formatere dataene på slik at du kan putte alt i en og samme fil.

3. **Vanlige feilmeldinger i Oblig 3:**
   Finn feilene i følgende program og foreslå hvordan de kan rettes. Programmet er en forenklet utgave av det fra oppgave 1. (b), hvor det ikke er noen obliger, men bare håndtering av studentene i én INF1000-gruppe. Programmet består av to klasser: Student, som bare har én objektvariabel: navn; og Gruppe, som har en array med opptil 40 studenter og en *konstruktør* med programkode som illustrerer bruk av klasser og objekter.

```
class Student {
   String navn;
}

class Gruppe {
   Student[] studenter = new Student[40]; // Array med Student-objekter
   int antStudenter; // Antall plasser i arrayen over som er i bruk

   public static void main(String[] args) {
      Gruppe g = new Gruppe();

      skrivAntall();
      /* Spørsmål:
       * a) Hvorfor gir linjen over følgende feilmelding?  Hvordan unngå det?
       *       Gruppe.java:12: non-static method skrivAntall() cannot
       *                       be referenced from a static context
       *           skrivAntall();
       *           ^
       *  Tips: Du kan lese mer om dette på side 199 (og 155) i læreboka.
       *    Problemet skyldes at main-metoden er en "klassemetode", dvs. deklarert
       *    med "static", og derfor kan man ikke referere til "objektvariabler"
       *    eller "objektmetoder" (dvs. variabler og metoder som ikke er deklarert
       *    me nøkkelordet static) direkte fra en static metode uten å gå
       *    via en peker, f.eks. pekeren g ovenfor.  Dette illustrerer
       *    hvorfor vi vanligvis bare har noen få programsetninger i
       *    main-metoden som bare setter programmet i gang, ved å kalle
       *    på andre metoder i programmet.
       */
   }

   Gruppe() {
      // b) Den vanligste feilmeldingen i Java er "cannot find symbol".
      //    Følgende linje gir feilmeldingen vist under.  Hva er feil i dette
      //    tilfellet, og hvordan kan vi rette det?
      antallStudenter = 0;
      /*                Gruppe.java:32: cannot find symbol
       *                symbol  : variable antallStudenter
       *                location: class Gruppe
       *                        antallStudenter = 0;
       *                        ^
       *  Tips: Årsaken til feilmeldingen "cannot find symbol" er alltid
       *    at noe ikke er deklarert riktig (f.eks. at det ikke er deklarert
       *    på riktig sted eller er stavet feil).  Feilen er heldigvis
       *    lett å fikse: se i feilmeldingen hva det var som ikke var
       *    deklarert, og sjekk at du har deklarert det riktig (på riktig
       *    sted i programmet, og stavet riktig).  Du kan finne hva det er
       *    som ikke var riktig deklarert på 2. linje i feilmeldingen, der
       *    det står "symbol".  I eksemplet over er problem-symbolet
       *    "variable antallStudenter".  Legg også merke til linjenummeret
       *    og posisjonen på linja som Java-kompilatoren også gir deg.
       */

      // c) Følgende linje gir "NullPointerException" her.  Hvilken setning
      //    har vi glemt å utføre før dette?  (Husk at studenter[] er en array)
      studenter[antStudenter].navn = "Trine";
      antStudenter++;
      /*  Tips: Feilmeldingen NullPointerException skyldes oftest at man har
       *    forsøkt å bruke prikk-notasjon på en peker som hadde verdien null.
       *    Det er ikke lov (det gir ingen mening å si f.eks. null.navn).
```

```
 *
 *  Slik fikser du NullPointerException-feil:
 *    1. Se i feilmeldingen hvilket linjenummer Java fant feilen i.
 *    2. Se hvilke pekere i den angitte linjen som kan ha vært null
 *       under kjøring av programmet, særlig blandt de som har prikk-
 *       notasjon etter seg, f.eks. hvis linjen inneholder uttrykket:
 *         hyblene[rad][kol].leietager.navn  ...så kan null-pekeren
 *       være hyblene[rad][kol] eller hyblene[rad][kol].leietager
 *    3. Endre programmet slik at pekeren som forårsaket feilen ikke
 *       kan være null når programmet kommer til den aktuelle linjen.
 *       Hvis problemet var en peker med prikknotasjon kan du legge
 *       til en if-test på at pekeren != null før den aktuelle linjen.
 *    Det som står foran prikken må altså være noe som peker på
 *    et allerede-oppretet objekt av riktig klasse (og ikke null), på
 *    det tidspunktet under programutførelsen når setningen blir utført.
 *
 *    4. Hvis linjen har flere pekere og du ikke finner hvilken som ga
 *       NullPointerException kan du legge inn en testutskrift rett før
 *       linja, og skrive ut én av dem, f.eks.
 *         System.out.println("test1:" + hyblene[rad][kol].leietager);
 *       Med denne fremgangsmåten vil du alltid kunne finne nullpekeren.
 */

// d) Skriv det som mangler for at studenter[1].navn skal bli "Martin".
Student s1 = new Student();
s1 _________ = "Martin";
studenter[antStudenter++] = _________ ;

// e) Hva mangler her for at if-testen skal kunne gi true?
//    Tips: Husk at tekster må sammenlignes med andre tekster.
if (studenter[1].equals("Martin")) {
    System.out.println(true);
}

// f) Hvorfor gir følgende løkke NullPointerException på linjen med
//    System.out..? (anta at eneste kode som er blitt utført når
//    programmet kommer hit er det som står i linjene over, med feilene
//    rettet).  Også, hvordan kan betingelsen i første linje endres for å
//    unngå NullPointerException?  Tips: Tenk arrayplasser og != null.
for (int i = 0; i < studenter.length; i++) {
    Student s2 = studenter[i];
    System.out.println(s2.navn);
}

// g) Finn en annen måte å unngå NullPointerException her uten å endre
//    første linje under, men ved å legge til en if-setning inne i løkka:
for (int i = 0; i < studenter.length; i++) {
    Student s3 = studenter[i];

    System.out.println(s3.navn);
}

// h) Hvorfor gir følgende linje i programkoden denne feilmeldingen:
//    ArrayIndexOutOfBoundsException: 40
studenter[40] = new Student();

// i) Hva vet vi om uttrykket merkert med "___" på neste linje hvis
//    linjen gir feilmeldingen "ArrayIndexOutOfBoundsException: -1":
s1 = studenter[ ___ ];
```

```
        // j) Hva er feil her?  Feilmeldingen dette gir er:
        //                  Gruppe.java:103: incompatible types
                          found   : java.lang.String
        //                  required: Student
        studenter[2] = "Eva";

        // k) Hva er feil her?  Feilmeldingen er: cannot find symbol
        //                  symbol  : constructor Student(java.lang.String)
        Student lars = new Student("Lars");

        // l) Hva er feil her?  Feilmeldingen dette gir er:
        //                  Gruppe.java:114: incompatible types
        //                  found   : int
        //                  required: boolean
        if (antStudenter = 0) {
            system.out.println("Ingen student registrert!");
        }
        // m) Hvorfor klager Java også med: "package system does not exist"?

        // n) Når vi har rettet alle feilene i a) til m) ovenfor, så gir fortsatt
        //    if-setningen under NullPointerException.  Hvordan unngår vi det?
        boolean funnet;
        for (int i = 0; i < studenter.length; i++) {
            Student stud = studenter[i];
            if (stud.navn.equals("Eva")) {
                 funnet = true;
                 System.out.println(stud.navn);
            }
        }

        // o) Løkken ovenfor stopper ikke når navnet "Eva" blir funnet.
        //    Hvordan kan vi få løkken til å stoppe når navnet blir funnet?

        // p) Løkken ovenfor gir ingen melding til bruker når navnet ikke blir
        //    funnet.  Hvordan kan vi programmere utskrift av en slik melding?
        //    Og hvordan kan vi unngå at Java da skal klage om at "variable
        //    funnet might not have been initialized"?

        // q) Anta at navnet til
        studenter[2] er "Eva".  Hvorfor endrer ikke
        //    følgende kode navnet til studenter[2]?  Hvordan kan det ordnes?
        //    (slik at studentobjektet i "ny" overføres til studenter[2]).
        Student ny = new Student();
        Student studPeker;
        ny.navn = "Heidi";
        studPeker = studenter[2];
        studPeker = ny;
    }

    void skrivAntall() {
        System.out.println("Antall studenter: " + antStudenter);
    }

    // r) Hvorfor får vi feilmeldingen: "<identifier> expected" på denne linjen:
    System.out.println("Slutt");
}
```

Flere debuggings-tips:
Både feilmeldingene fra kompilatoren (`javac`) og kjøresystemet til Java (`java`) gir deg vanligvis linjenummeret der feilen oppsto, og navnet til feilen. Disse opplysningene er nyttige for å finne og rette feilen. Start alltid med å rette den