INF1000 Prøveeksamen
Oppgave 7 og 9

Høst 2015

Siri Moe Jensen

Oppgave 7 (47 poeng)

Herr Glum lager en julekalender til barna sine hvert år. Kalenderen inneholder en gave for hver dag i desember, frem til og med julaften 24.12. Det går på omgang mellom barna hvem som får lov å åpne dagens gave – når alle har åpnet en gave hver, er det førstemann sin tur igjen. Nå ønsker herr Glum seg et program som kan hjelpe ham med holde rede på hvilke barn som får hvilke gaver, og hvor mye gavene hvert barn får, har kostet.

Programmet skal kunne lese inn data om gavene fra en fil som herr Glum oppdaterer etter hvert som han handler inn gaver i tiden før desember. Filen inneholder 48 linjer, 2 for hver gave: Én linje med navn på gaven (en tekststreng), deretter én linje med prisen (et heltall). Du skal hjelpe ham å skrive dette programmet i Java.

a) Skriv en klasse **Gave** med to variabler som forteller hva som er i gaven, og hvor mye den har kostet. Klassen skal ha en konstruktør med parametere som angir verdier for objektvariablene. Foruten konstruktøren skal klassens grensesnitt omfatte tre metoder: En som returnerer gavepris; en som returnerer gavenavn; og en metode **toString** som returnerer gavens navn og verdi som en **String**.

7a)
Skriv en klasse **Gave** med to variabler som forteller hva som er i gaven, og hvor mye den har kostet. Klassen skal ha en konstruktør med parametere som angir verdier for objektvariablene. Foruten konstruktøren skal klassens grensesnitt omfatte tre metoder: En som returnerer gavepris; en som returnerer gavenavn; og en metode **toString** som returnerer gavens navn og verdi som en **String**.

```
class Gave {
  private String innh;
  private int kr;

  public Gave (String hva, int kr) {
    innh = hva;
    this.kr = kr;
  }

  public int getPris () {
    return kr;
  }

  public String toString () {
    return (innh + " " + kr);
  }

  public String getInnhold () {
    return innh;
  }
}
```

- private!
- ved eksamen:
  - legger ikke vekt på eksplisitt public

7b)
Klassen **Barn** skal ha en datarepresentasjon for barnets navn, alle gavene barnet har åpnet, og totalverdien av gaver barnet har åpnet. Grensesnittet til klassen skal være en konstruktør med barnets navn som parameter, en metode for avlesing av totalverdien av alle gaver barnet har mottatt, en metode **apneGave** som legger til en ny gave og oppdaterer totalverdien av gaver barnet har fått, og en metode **skrivBarn** som skriver ut på terminal barnets navn, en linje for hver av barnets gaver og til slutt totalverdien av barnets gaver. Skriv klassen **Barn** med alt innhold.

- samling gaver
  - ukjent antall
  - ikke oppslag?
  - liste opp

```
class Barn {
  private String navn;
  private int totKost;
  private ArrayList <Gave> fatt = new ArrayList <>();

  public Barn (String hvem) {
    navn = hvem;
    totKost = 0;
  }

 // class Barn fortsetter…
```

7b) forts
..Grensesnittet til klassen skal være en konstruktør med barnets navn som parameter, en metode for avlesing av totalverdien av alle gaver barnet har mottatt, en metode **apneGave** som legger til en ny gave og oppdaterer totalverdien av gaver barnet har fått, …..

```
// ..class Barn fortsetter

  public String getNavn () {
    return navn;
  }

  public int getTotKost () {
    return totKost;
  }

  public void apneGave (Gave ny) {
    fatt.add(ny);
    totKost += ny.getPris();
  }

// class Barn forsetter…
```

7b) forts
.. og en metode **skrivBarn** som skriver ut på terminal barnets navn, en linje for hver av barnets gaver og til slutt totalverdien av barnets gaver...

- spesialisert for-løkke
- bruker toString

```
// ..class Barn fortsetter

 public void skrivBarn () {
    System.out.println (navn + ":");

    for (Gave g: fatt) {
      System.out.println (g);
    }

    System.out.println("Totalverdi gaver for "
                       + navn + ": " + totKost);
    System.out.println ();
  }
}
```

Vi skal i de senere oppgavene utvide programmet, som skal inneholde en klasse **Julekalender** med blant annet følgende innhold:

```
private Gave [] kalender;                  // pekere til en gave for hver dag
private Barn [] apnere;                    // pekere til hvert av barna i familien
private int nesteApner;                    // holder rede på hvem sin tur det er til å åpne
private int dag;                           // hvilken dag skal åpnes neste gang
Julekalender (String[] barneNavn, String filnavn)   // Konstruktør
private void lesGavefil (String filnavn)            // leser inn gaver med pris fra fil
void nyDag ()                              // åpner en ny gave og oppdaterer datastrukturen
void gaveOversikt ();                      // Skriver en oversikt over barna, deres åpnede gaver
                                           // og totalpris per barn på skjermen
```

- vet før bruk hvor mange gaver og barn som skal håndteres
- aksesseres sekvensielt
- => array eller ArrayList, her oppgitt som array
- lesGavefil oppgitt som private. Kun ved oppstart => i konstruktør

…. Filen inneholder 48 linjer, 2 for hver gave: Én linje med navn på gaven (en tekststreng), deretter én linje med prisen (et heltall)….

7c)
Skriv metoden **lesGavefil** i klassen **Julekalender**, som leser inn alle gavene fra filen oppgitt i parameteren og legger disse inn i kalenderen. Filformatet er beskrevet ovenfor.

```
private void lesGavefil (String filnavn) throws Exception {
   File fil = new File(filnavn);
   Scanner gavefil = new Scanner (fil);

   for (int i=0; i<24; i++) {
     String gnavn = gavefil.nextLine();
     int gpris = Integer.parseInt (gavefil.nextLine());
     kalender[i] = new Gave (gnavn, gpris);
   }
 }
```

- kaster unntak – må gjøres også i metoder som kaller denne

7d)
Skriv konstruktøren til klassen **Julekalender**. Konstruktøren skal opprette objekter for alle barna i familien og opprette selve julekalenderen med gaver.

```
class Julekalender {
  private Gave[] kalender = new Gave[24];
  private Barn[] apnere;
  private int nesteApner;
  private int dag;

  public Julekalender (String[] barneNavn, String filnavn)
throws Exception {

    lesGavefil (filnavn);
    apnere = new Barn [barneNavn.length];
    for (int i=0; i<apnere.length; i++) {
      apnere[i] = new Barn (barneNavn[i]);
    }
    nesteApner = 0;
    dag = 0;
  }
```

- Fast antall gaver, varierende antall barn
- Får array med barnenavn, trenger objekter
- Indeksering fra 0, <> dato

7e)
Skriv metoden **nyDag** i klassen **Julekalender**.
Husk å oppdatere nesteApner.

```
public void nyDag () {
   Barn dagens = apnere[nesteApner];
   dagens.apneGave(kalender[dag]);
   dag++;
   nesteApner++;
   if (nesteApner == apnere.length) {
     nesteApner = 0;
   }
 }
```

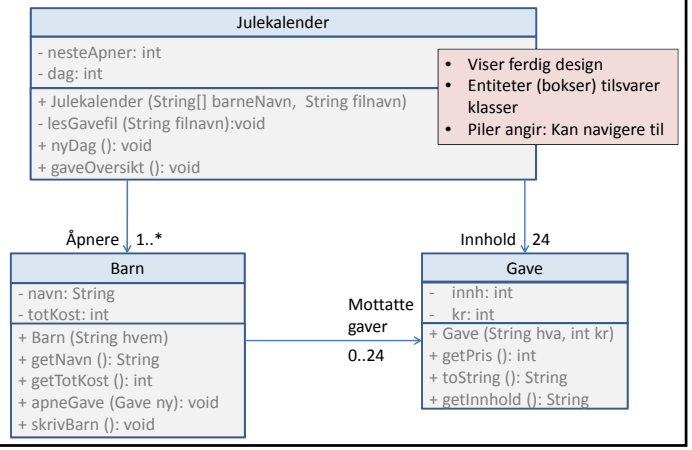
- Hvilket barn skal åpne?
- kall apneGave for dette barnet, som oppdaterer barnet
- klar for ny gave og nytt barn

7f)
Skriv metoden **gaveOversikt** i klassen **Julekalender**.

```
public void gaveOversikt () {
   for (int i=0; i<apnere.length; i++) {
     apnere[i].skrivBarn ();
     System.out.println();
   }
 }
```

- Skal skrive ut hvert barn, og dets gaver
- hvert barn skriver ut seg selv

7g)
Tegn et UML klasse-diagram som viser programmets klasser med attributter og metoder, og deres relasjoner.



7f)
Vi skal nå utvide klassen Julekalender med historikk fra tidligere år. I første omgang er hensikten å redusere sjansen for at et barn får samme gave flere år på rad. Du kan anta at gavenavn er entydige. Klassen Julekalender er utvidet med objektvariabel og metoder som vist nedenfor. Du skal skrive metoden **avvergetLike** som beskrevet nedenfor, og utvide metoden **nyDag** med kall på **avvergetLike.** Dersom **avvergetLike** returnerer **false**, skal du skrive ut en beskjed på terminalen . Beskriv eventuelle andre behov for endringer i metoder du allerede har skrevet.

- Denne oppgaven viser eksempler på noen utfordringer som kan komme
- bruk av kjente ting på nye måter
- les nøye – ikke implementere mer enn bedt om
- Bruk piler og kommentarer i marg, ikke skriv (for mye) på nytt
- design preget av utvidelse
- "i første omgang" – kan komme flere

7f) forts.

/* HashMap **historikk** inneholder Gave-objekter for alle gaver som har vært delt ut tidligere år. Nøkkel er en String bestående av navn på barnet som fikk den, konkatenert (sammensatt) med navnet på gaven.*/

```
private HashMap<String,Gave> historikk = new HashMap<>();
```

/* Metoden **lesHistorikk** leser inn all historikk om tidligere utdelte gaver fra filen **Historikk.txt**, og bygger opp HashMap'en historikk som beskrevet over. *Denne metoden skal du ikke skrive selv.* */

```
void lesHistorikk(String filnavn) throws Exception {}
```

/* Metoden avvergetLike prøver å avverge at en barn får en gave det har fått i tidligere år. Den kalles hver dag før metoden apneGave. Dersom gaven og barnet som står for tur sammenfaller med noe som er gitt tidligere år, byttes gaven som står for tur med gaven for neste dag i kalenderen. I de tilfeller denne strategien ikke kan hindre at et barn får en gave det har fått før, skal metoden returnere **false**. Dersom barnet ikke har fått gaven som står for tur tidligere, eller du klarer å avverge det ved å bytte, skal metoden returnere **true**: */

```
private boolean avvergetLike () {}
```

/* Metoden **skrivHistorikk** lagrer årets gaver på fil sammen med tidligere historikk. Metoden skal *ikke* skrives av deg: */

```
void skrivHistorikk (String filnavn) {}
```

- Nøkkel sammensatt av flere elementer
- Gjenbruk av gaveobjekter i ny sammenheng
- Endringer i besvarte metoder

7f) forts.

Tillegg i klassen Julekalender: Ny objektvariabel og leser historikk i konstruktør

```
class Julekalender {
  private Gave[] kalender = new Gave[24];
  private Barn[] apnere;
  private int nesteApner;
  private int dag;
  private HashMap<String, Gave> historikk = new HashMap<>();

  public Julekalender (String[] barneNavn, String filnavn)
                                          throws Exception {
    lesHistorikk ("Historikk.txt");
    lesGavefil (filnavn);
    apnere = new Barn [barneNavn.length];
    for (int i=0; i<apnere.length; i++) {
      apnere[i] = new Barn (barneNavn[i]);
    }
    nesteApner = 0;
    dag = 0;
  }
```

7f) forts.

```
private boolean avvergetLike () {
   String barn = apnere[nesteApner].getNavn();
   String innhold = kalender[dag].getInnhold();
   if (historikk.containsKey(barn+innhold)) {    // Bytt Gave
     if (dag == 23) {
       return false;
     }
     Gave tmp = kalender[dag];
     kalender[dag] = kalender[dag+1];
     kalender[dag+1] = tmp;
     innhold = kalender[dag].getInnhold();
     if (historikk.containsKey(barn+innhold)) {
       return false;
     }
   }
   return true;
```

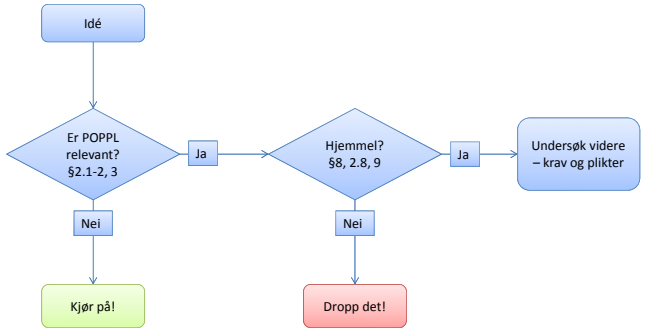
- Skal starte en ny dag
- Finner barn og gave som står for tur
- Skal gi beskjed om resultat
- Hvilke tilfeller klarer vi ikke å fikse?
  - julaften
  - den vi byttet til er også dublett

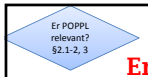
7f) forts.

Tillegg i metoden **nyDag**

```
public void nyDag () {
   if (!avvergetLike()) {
     System.out.println ("Dublett i kalenderluke " + (dag+1)
                         + ": " + kalender[dag].getInnhold());
   }
   Barn dagens = apnere[nesteApner];
   dagens.apneGave(kalender[dag]);
   dag++;
   nesteApner++;
   if (nesteApner == apnere.length) {
     nesteApner = 0;
   }
 }
```

Lov om behandling av personopplysninger (Personopplysningsloven, Poppl)





Definisjoner og virkeområde I: Er systemet/ dataene berørt av Poppl?

§ 2. *Definisjoner*

I denne loven forstås med:

1) personopplysning: opplysninger og vurderinger som kan knyttes til en enkeltperson,

2) behandling av personopplysninger: enhver bruk av personopplysninger, som f.eks. innsamling, registrering, sammenstilling, lagring og utlevering eller en kombinasjon av slike bruksmåter,
...

Utvalg og understrekninger gjort for forelesning INF1000

Oppgave 9 (6 poeng)

Er dette en personopplysning? Begrunn med henvisning til Personopplysningsloven:
Mann, født 1990. Mangler fast bopæl. Straffedømt for narkotikabruk.
Nei, informasjonen kan ikke knyttes til en enkeltperson (§ 2.1)

Nevn tre sentrale hensyn til informasjonssikkerhet som ifølge Personopplysningsloven skal ivaretas ved behandling av personopplysninger.
Konfidensialitet, integritet, tilgjengelighet (§13 Informasjonssikkerhet)

Som ledd i arbeidet mot forsikringssvindel har norske forsikringsselskaper fått konsesjon for lagring av følgende opplysninger om sine skadeoppgjør i et felles, sentralt skaderegister: Forsikringstakers fødselsnummer, saksnummer, bransjekode, selskap, skadetype, dato, saksbehandlers initialer og skadeland.
Konsesjonen omfatter ikke tillatelse til registrering av sensitive personopplysninger. Diskuter om noen av opplysningene konsesjonen omfatter kan komme til å omfatte sensitive personopplysninger slik at spesiell varsomhet bør utvises ved registrering av disse i det felles registeret.
«skadetype» kan tenkes å gi indikasjoner på helseforhold. (§2.8)
<poeng her for alle eksempler på sensitive personopplysninger som KAN bli aktuelle>