Avdeling for allmenne fag

EKSAMEN

852 OBJEKTORIENTERT PROGRAMMERING

08.12.2004

Tid:	3 timar, 9.00 – 12.00
Målform:	Nynorsk/bokmål
Sidetal:	forside + 3 + 3
Hjelpemiddel:	Alle trykte og skrevne
Merknader:	Ingen
Vedlegg:	Ingen

Eksamensresultata vert offentleggjort på Arena høgskole. Passordet til Arena har alle studentar fått i eige brev ved semesterstart. Du går inn på Arena høgskole gjennom HiTs nettsider.

Eksamen 852 Objektorientert Programmering 08.12.2004 – Nynorsk

Råd og retningsliner. Les oppgåveteksten godt før du går i gang med å løyse oppgåva. Deloppgåvene er uavhengige av kvarandre i den meining at om du ikkje får til éi oppgåve, kan du likevel gjere neste, som om den fyrste var løyst. Fordél tida godt på alle oppgåvene. Om du meiner ei oppgåve er upresis, så skriv din eigen presisering.

Oppgåve 1 (45%)
Du skal delta i eit utviklingsprosjekt for ein bank. Banken skal bytte ut all sin gamle programvare som er flikka på sidan 60-talet, alt skal designast på nytt. Det endelege systemet skal sjølvsagt bli eit databasesystem, men det skal fyrst utviklast ein prototyp i rein Java.

1a (15%)
Ein bank held orden på konti av ulike slag. Felles eigenskapar skal beskrivast i class Konto. Den skal ha instansvariable for å halde på saldo (innestående beløp på konto), innehavar av kontoen (ein instans av class Innehavar), rentesatsen på kontoen i prosent, og eit 11-sifra kontonummer. Den skal ha konstruktørmetode med parametre tilsvarande desse fire instansvariablane, get-metode for kvar av dei fire instansvariablane, og set-metode for innehavar og rente. Du treng ikkje sjekke inndata på nokon av metodane. Programmer ferdig class Konto med utgangspunkt i denne skissa:

```
public class Konto {
  // instansvariable her...
  public Konto(float saldo, Innehavar innehavar,
    float rente, String kontonr) {...}
  public float getSaldo() { ...}
  public Innehavar getInnehavar() {...}
  public float getRente() {...}
  public String getKontonr() {...}
  public void setInnehavar(Innehavar innehavar) {...}
  public void setRente(float rente) {...}
}
```

1b (15%)
Ein innskotskonto er ein konto der saldo ikkje kan vere negativ. Ein kan gjere innskot (sette pengar inn på kontoen) og uttak (ta pengar ut av kontoen), men pass på at det ikkje vert negativ saldo. Pass også på at det berre vert gjort innskot og uttak av positive beløp, ein skal ikkje kunne kamuflere uttak som eit negativt innskot. Ulovlege operasjonar skal kaste unntak. Programmer class InnskotsKonto som subklasse av Konto med konstruktørmetode og metodar for innskot og uttak.

1c (15%)
Bankar baserer stadig meir av inntektene sine på gebyr. Klassa GebyrKonto skal vere subklasse av InnskotsKonto, og ha ein ekstra instansvariabel "antalTransaksjonar" som skal nyttast til å telle opp kor mange transaksjonar (innskot og uttak) som er gjort. Du må kanskje reimplementere nokre metodar for å få denne funksjonaliteten. Ein metode getGebyr skal returnere totalgebyret på kontoen, når kvar transaksjon kostar 2 kr. Programmer ferdig klassa GebyrKonto.

Oppgåve 2 (30%)
Ein **transaksjon** er ein flytting av eit visst beløp pengar på ein bestemt dato. Det kan vere a) kontantinnskot på konto, b) kontantuttak frå konto, eller c) overføring mellom to konti. Dersom vi skal representere ein transaksjon som eit objekt, ser vi at ved innskot treng vi ein instansvariabel til å representere til-konto – kontoen vi set pengar inn på. Ved uttak treng vi ein instansvariabel til å halde på frå-konto – kontoen vi tek pengar frå. Ved overføring treng vi både til-konto og frå-konto. I tillegg treng vi instansvariable for dato (anta at du har class Dato som kan nyttast her) og beløp.

2a (15%)
Teikn eit klassehierarki som i det minste inneheld class Object og dei tre transaksjonsklassene antyda over, berre vurdert ut i frå IS-A relasjonen. Før deretter opp kva som må deklarerast av instansvariable i kvar klasse, og diskuter ut frå dette *kort* eventuelle problem du ser ved å nytte eit reint klassehierarki her. Kontonummer kan også her representerast som String.

2b (15%)
Det er ikkje nok å ta vare på saldoen til ein konto. Ein bank må også ta vare på historia til kontoen, alle transaksjonar som har skjedd. Det skal utviklast ei ny klasse for å ta vare på alle transaksjonane. Vi kunne t.d. lage denne som subklasse av Vector, men skal i staden nytte ein instansvariabel av type Vector. Altså:

```
class Transaksjonar {
  protected Vector transaksjonsVektor;

  public Transaksjonar(){...}
  public void nyTransaksjon(float beløp, String fråKonto,
    String tilKonto){...}
  public float saldo(String konto){...}
}
```

Her er class Transaksjon ei enkel klasse, det er ikkje nytta klassehierarki. Ved innskot/uttak er den unyttige String'en null.

```
class Transaksjon {
  // instansvariable...
  public Transaksjon(float beløp, Dato dato, String fråKonto,
    String tilKonto) throws IllegalArgumentException {...}
  public float getBeløp(){...}
  public Dato getDato(){...}
  public String getFråKonto(){...}
  public String getTilKonto(){...}
}
```

Lag den parameterlause konstruktørmetoden i class Transaksjonar.
Lag metoden nyTransaksjon som opprettar instans av class Transaksjon og legg denne inn på slutten av vektoren. Dato-klassa har parameterlaus konstruktør som lagar rett dato.
Lag saldo-metoden i class Transaksjonar. Den skal gå igjennom alle transaksjonar, sjå etter transaksjonar som gjeld det oppgitte kontonummer, og rekne ut saldoen på denne kontoen.
Du skal ***ikkje*** lage class Transaksjon.

Oppgåve 3 (25%)
I bankanes betalingssentral (BBS) har dei totaloversikt over alle bankar og alle konti. Ein bank er identifisert ved eit firesifra nummer, dei fire fyrste sifra av eit 11-sifra kontonummer.

```
class Bank {
  private String bankNummer;  // 4 siffer
  private Vector konti;  // Kan innehalde instansar av
                         // InnskotsKonto og GebyrKonto.
  ...
}

class BBS {
  private Vector bankar;  // inneheld Bank-objekt
  private Transaksjonar transaksjonar;

  public void transaksjon(float beløp, String fråKonto,
    String tilKonto) throws IllegalArgumentException {...}
  ...
}
```

Lag metoden **transaksjon** i class BBS. Her er fråKonto og tilKonto 11-sifra kontonummer, evt. null, og beløpet skal vere positivt. Metoden skal oppdatere transaksjonsvektoren, og dessutan finne fram dei rette kontoane i dei rette bankane og oppdatere saldoane der. Lag gjerne hjelpemetodar i klasser du måtte ynskje. Test inndata så godt du kan. Du skal ikkje introdusere nye instansvariablar.

Lykke til!

Eksamen 852 Objektorientert Programmering 08.12.2004 – Bokmål

Råd og retningslinjer. Les oppgaveteksten godt før du går i gang med å løse oppgava. Deloppgavene er uavhengige av hverandre i den forstand at om du ikke får til én oppgave, kan du likevel gjøre neste, som om den første var løst. Fordél tida godt på alle oppgavene. Om du mener en oppgave er upresis, så skriv din egen presisering.

Oppgave 1 (45%)
Du skal delta i et utviklingsprosjekt for en bank. Banken skal bytte ut all sin gamle programvare som er flikket på siden 60-tallet, alt skal designes på nytt. Det endelige systemet skal selvsagt bli et databasesystem, men det skal først utvikles en prototyp i ren Java.

1a (15%)
En bank holder orden på konti av ulike slag. Felles egenskaper skal beskrives i class Konto. Den skal ha instansvariable for å holde på saldo (innestående beløp på konto), innehaver av kontoen (en instans av class Innehaver), rentesatsen på kontoen i prosent, og et 11-sifret kontonummer. Den skal ha konstruktørmetode med parametre tilsvarende disse fire instansvariablene, get-metode for hver av de fire instansvariablene, og set-metode for innehaver og rente. Du trenger ikke sjekke inndata på noen av metodene. Programmer ferdig class Konto med utgangspunkt i denne skissa:

```
public class Konto {
  // instansvariable her...
  public Konto(float saldo, Innehaver innehaver,
    float rente, String kontonr) {...}
  public float getSaldo() { ...}
  public Innehaver getInnehaver() {...}
  public float getRente() {...}
  public String getKontonr() {...}
  public void setInnehaver(Innehaver innehaver) {...}
  public void setRente(float rente) {...}
}
```

1b (15%)
En innskuddskonto er en konto der saldo ikke kan være negativ. Man kan gjøre innskudd (sette penger inn på kontoen) og uttak (ta penger ut av kontoen), men pass på at det ikke blir negativ saldo. Pass også på at det bare blir gjort innskudd og uttak av positive beløp, man skal ikke kunne kamuflere uttak som et negativt innskudd. Ulovlige operasjoner skal kaste unntak. Programmer class InnskuddsKonto som subklasse av Konto med konstruktørmetode og metoder for innskudd og uttak.

1c (15%)
Banker baserer stadig mer av inntektene sine på gebyr. Klassa GebyrKonto skal være subklasse av InnskuddsKonto, og ha en ekstra instansvariabel "antallTransaksjoner" som skal brukes til å telle opp hvor mange transaksjoner (innskudd og uttak) som er gjort. Du må kanskje reimplementere noen metoder for å få denne funksjonaliteten. En metode getGebyr skal returnere totalgebyret på kontoen, når hver transaksjon koster 2 kr. Programmer ferdig klassa GebyrKonto.

Oppgave 2 (30%)
En **transaksjon** er en flytting av et visst beløp penger på en bestemt dato. Det kan være a) kontantinnskudd på konto, b) kontantuttak fra konto, eller c) overføring mellom to konti. Hvis vi skal representere en transaksjon som et objekt, ser vi at ved innskudd trenger vi en instansvariabel til å representere til-konto – kontoen vi setter penger inn på. Ved uttak trenger vi en instansvariabel til å holde på fra-konto – kontoen vi tar penger fra. Ved overføring trenger vi både til-konto og fra-konto. I tillegg trenger vi instansvariable for dato (anta at du har class Dato som kan brukes her) og beløp.

2a (15%)
Tegn et klassehierarki som i det minste inneholder class Object og de tre transaksjonsklassene antydet over, bare vurdert ut i fra IS-A relasjonen. Før deretter opp hva som må deklareres av instansvariable i hver klasse, og diskuter ut fra dette *kort* eventuelle problemer du ser ved å bruke et rent klassehierarki her. Kontonummer kan også her representeres som String.

2b (15%)
Det er ikke nok å ta vare på saldoen til en konto. En bank må også ta vare på historia til kontoen, alle transaksjoner som har skjedd. Det skal utvikles en ny klasse for å ta vare på alle transaksjonene. Vi kunne f.eks. lage denne som subklasse av Vector, men skal i stedet bruke en instansvariabel av type Vector. Altså:

```
class Transaksjoner {
  protected Vector transaksjonsVektor;

  public Transaksjoner(){...}
  public void nyTransaksjon(float beløp, String fraKonto,
    String tilKonto){...}
  public float saldo(String konto){...}
}
```

Her er class Transaksjon en enkel klasse, det er ikke brukt klassehierarki. Ved innskudd/uttak er den unyttige String'en null.

```
class Transaksjon {
  // instansvariable...
  public Transaksjon(float beløp, Dato dato, String fraKonto,
    String tilKonto) throws IllegalArgumentException {...}
  public float getBeløp(){...}
  public Dato getDato(){...}
  public String getFraKonto(){...}
  public String getTilKonto(){...}
}
```

Lag den parameterløse konstruktørmetoden i class Transaksjoner.
Lag metoden nyTransaksjon som oppretter instans av class Transaksjon og legger denne inn på slutten av vektoren. Dato-klassa har parameterløs konstruktør som lager riktig dato.
Lag saldo-metoden i class Transaksjoner. Den skal gå igjennom alle transaksjoner, se etter transaksjoner som gjelder det oppgitte kontonummer, og regne ut saldoen på denne kontoen.
Du skal ***ikke*** lage class Transaksjon.

Oppgave 3 (25%)
I bankenes betalingssentral (BBS) har de totaloversikt over alle banker og alle konti. En bank er identifisert ved et firesifret nummer, de fire første sifrene av et 11-sifret kontonummer.

```
class Bank {
  private String bankNummer;  // 4 siffer
  private Vector konti;  // Kan inneholde instanser av
                         // InnskuddsKonto og GebyrKonto.
  ...
}

class BBS {
  private Vector banker;  // inneholder Bank-objekter
  private Transaksjoner transaksjoner;

  public void transaksjon(float beløp, String fraKonto,
    String tilKonto) throws IllegalArgumentException {...}
  ...
}
```

Lag metoden **transaksjon** i class BBS. Her er fraKonto og tilKonto 11-sifret kontonummer, evt. null, og beløpet skal være positivt. Metoden skal oppdatere transaksjonsvektoren, og dessuten finne fram de rette kontoene i de rette bankene og oppdatere saldoene der. Lag gjerne hjelpemetoder i klasser du måtte ønske. Test inndata så godt du kan. Du skal ikke introdusere nye instansvariabler.

Lykke til!