Universitetet i Bergen
Det matematisk–naturvitenskapelige fakultet
Institutt for informatikk

BOKMÅL

EKSAMEN I EMNET INF100 Grunnkurs i programmering (Programmering 1)
Fredag 12. desember 2003
Tid: 09:00–14:00

Tillatte hjelpemiddel: Alle trykte og skrevne hjelpemiddel er tillatt.

I hele oppgavesetttet antar vi at klassen `Terminal` (se vedlegg A for de viktigste metodene) er tilgjengelig. Du kan også gjøre bruk av klassen i besvarelsen.

Oppgave 1 (20 %)

I denne oppgaven er det gitt en del program som enten skriver ut eller tegner et tall på skjermen. I hver deloppgave skal du finne ut hvilket tall som skrives eller tegnes. Det er ikke nødvendig med noen begrunnelse av svaret.

a) (4%) Hvilket tall vises på skjermen når `Klient1a` kjøres?

```
public class Klient1a {
  public static void main( String[] args ) {
    int i=1;
    int j=2;
    System.out.println(j++);
    j=i;
  }
}
```

b) (4%) Hvilket tall vises på skjermen når `Klient1b` kjøres?

```
public class Klient1b {
  public static void main( String[] args ) {
    int i=0;
    int j=2;
    int k=0;
    boolean fortsett = i<j;
    while (fortsett && k<2) {
      i++;
      j--;
      k++;
    }
    System.out.println(j);
  }
}
```

c) (4%) Hvilket tall vises på skjermen når `Klient1c` kjøres?

```
public class Oppg1c {

  private int felt0;

  public Oppg1c(int f) {
    felt0 = f;
  }

  public int metode0(int i) {
    if (i<=felt0)
      return i;
    else if (i<=2*felt0)
      return felt0;
    return 0;
  }
}

public class Klient1c {
  public static void main( String[] args ) {
    Oppg1c oppg1c = new Oppg1c(4);
    System.out.println(oppg1c.metode0(5));
  }
}
```

d) (4%) Hvilket tall vises på skjermen når `Klient1d` kjøres?

```
public class Oppg1d {

  private int felt0;
  private static int felt1;

  public Oppg1d(int f0, int f1) {
    felt0 = f0;
    felt1 = f1;
  }

  public int metode0() {
    return felt0+felt1;
  }
}

public class Klient1d {
  public static void main( String[] args ) {
    Oppg1d obj1 = new Oppg1d(1,0);
    Oppg1d obj2 = new Oppg1d(3,obj1.metode0());
    System.out.println(obj1.metode0());
  }
}
```

e) (4%) Hvilket tall vises på skjermen når `Klient1e` kjøres?

```
public class Oppg1e {

  private int bredde;
  private int høyde;
  private char tegn;

  public Oppg1e(int bredde, char tegn) {
    this.bredde = bredde;
    this.høyde = 2*bredde - 1;
    this.tegn = tegn;
  }
```

```
  public void metode0() {
    for (int i=1; i<=høyde; ++i)
      if (i==1 || i==bredde || i==høyde)
        metode1(' ',tegn);
      else
        metode1(tegn,' ');
  }

  public void metode1(char c1, char c2) {
    for (int i=1; i<=bredde; ++i)
      if (i==1 || i==bredde)
        System.out.print(c1);
      else
        System.out.print(c2);
    System.out.println();
  }

}


public class Klient1e {
  public static void main( String[] args )
  {
    System.out.print("Dette er et program som tegner et mønster");
    System.out.println(" som forestiller et tall på skjermen.");
    System.out.print("Du bestemmer selv hvilken bredde mønsteret skal ha,");
    System.out.println(" og hvilket tegn som skal inngå i mønsteret.");
    int bredde = 0;
    while (bredde < 3 || bredde > 10) {
      System.out.print("\tGi bredde (mellom 3 og 10): ");
      bredde = Terminal.lesInt();
    }
    System.out.print("\tGi tegn (f.eks '*'): ");
    char tegn = Terminal.lesString().charAt(0);
    Oppg1e oppg1e = new Oppg1e(bredde, tegn);
    oppg1e.metode0();
  }
}
```

Oppgave 2 (38%)

Gitt klassen `Terning` i Vedlegg B (s. 10). Klassen representerer terninger som kan brukes i ulike terningspill. Den har en metode `kast` for å kaste terningen på nytt, og en metode `hentVerdi` som uten å kaste terningen leser av og returnerer verdien vi fikk ved siste kast. Klassen benytter metoden `Math.random()` som returnerer et tilfeldig flyttall mellom 0 og 1.

a) (15%) Vi vil ha et Java-program som gjør det mulig for brukeren å spille følgende spill: Brukeren spiller alene, og skal prøve å få flest mulig poeng. Ved spillets start har brukeren 0 poeng. Spillet foregår i flere omganger, og brukeren kaster terningen en gang i hver omgang. Før en omgang eventuelt starter skal brukeren få spørsmål om han vil spille en omgang til. Hvis han svarer ja, og han i denne omgangen får en verdi som er minst like stor som verdien han fikk i omgangen før, økes poengsummen med verdien fra denne omgangen. Får han en verdi som er mindre enn verdien han fikk i omgangen før, settes poengsummen til 0, og spillet avsluttes. Svarer han nei på spørsmålet går brukeren ut av spillet med den poengsummen han har samlet opp.

Lag et slikt program ved å bruke klassen `Terning`. Hvis du ikke klarer å bruke klassen, kan du skrive programmet uten bruk av klassen (i så fall blir maksimal uttelling på deloppgaven 12%).

Mulig brukerdialog, eksempel 1:

```
Du har 0 poeng. Flere omganger(j/n)? j
Poeng i omgangen: 4
Du har 4 poeng. Flere omganger(j/n)? j
Poeng i omgangen: 5
Du har 9 poeng. Flere omganger(j/n)? j
Poeng i omgangen: 2
Spillet avsluttes med 0 poeng.
```

Mulig brukerdialog, eksempel 2:

```
Du har 0 poeng. Flere omganger(j/n)? j
Poeng i omgangen: 3
Du har 3 poeng. Flere omganger(j/n)? j
Poeng i omgangen: 5
Du har 8 poeng. Flere omganger(j/n)? j
Poeng i omgangen: 5
Du har 13 poeng. Flere omganger(j/n)? n
Spillet avsluttes med 13 poeng.
```

b) (15%) I mange terningspill spilles det med mer enn en terning. Vi har derfor bruk for en klasse, `FlereTerninger` (skrevet i Java), som representerer et sett terninger. Klassen skal ha følgende konstruktører/metoder:

- En konstruktør, `public FlereTerninger(int antall)`, som oppretter et sett med et antall terninger. Antallet gis som parameter til konstruktøren.
- En metode `public int kast()` som kaster alle terningene på nytt, og som returnerer total verdi som terningene viser.
- En metode `public int hentVerdi(int i)` som henter ut verdien som terning `i` viser i siste kast.
- En metode `public int hentVerdi()` som henter ut total verdi som terningene viser til sammen i siste kast.

Merk at de to sistnevnte metodene bare henter ut verdiene uten å kaste terningene på nytt. Det behøver ikke å være mulig å legge til nye terninger til et sett etter at det er opprettet.

Lag en slik klasse ved å gjøre bruk av klassen `Terning`. Hvis du ikke får til å bruke klassen `Terning`, kan du lage den uten (i så fall blir maksimal uttelling på deloppgaven 12%).

c) (8%) Bruk klassen `FlereTerninger` til å lage et Java-program som lar brukeren spille samme typen spill med samme regler som spillet i oppgave a, men med den forskjell at det spilles med flere terninger. I hver omgang kastes alle terningene, og verdien i omgangen er lik summen av verdiene til alle terningene. Før spillet starter skal bruker bli bedt om å oppgi antall terninger han vil spille med (samme antall i hver omgang). Hvis du ikke fikk til oppgave b, kan du likevel gå ut fra at du har tilgjengelig en klasse `FlereTerninger` som passer til beskrivelsen i oppgave b.

Oppgave 3 (42 %)

Vi skal lage en klasse (i Java), `Kommune`, som representerer kommuner. En kommune er kjennetegnet med et navn (tegnstreng), et antall innbyggere i fast arbeid, et antall arbeidssøkende innbyggere og antall andre innbyggere (omfatter skoleelever, studenter, pensjonister etc.) som ikke har fast arbeid, men som heller ikke søker arbeid.

I hver av deloppgavene a-g nedenfor spør vi etter feltvariable, konstruktører eller metoder til klassen. Les gjennom alle deloppgavene før du begynner å svare. Hver konstruktør/metode må påse at feltvariablene ikke tildeles meningsløse verdier.

a) (3%) Foreslå feltvariable for klassen.

b) (3%) Lag en konstruktør som oppretter en kommune. Kommunenavn og antall innbyggere av de ulike kategoriene gis som parametre.

c) (3%) Lag fire metoder som henter ut henholdsvis kommunenavnet, antall innbyggere med fast arbeid, antall arbeidssøkende innbyggere og antall andre innbyggere.

d) (3%) Lag en metode som henter ut totalt antall innbyggere, og en metode som henter ut arbeidsløsheten (antall arbeidssøkende som prosent av antallet med fast arbeid og arbeidssøkende til sammen).

e) (3%) Lag tre metoder som gir nye verdier til henholdsvis antall innbyggere med fast arbeid, antall arbeidssøkende innbyggere og antall andre innbyggere. De nye verdiene gis inn som parametre.

f) (3%) Lag en metode som skriver feltvariablenes verdier til en tekstfil. Metoden skal ha en filstrøm av klassen `PrintWriter` som parameter, og vi antar at strømmen er koblet til filen det skal skrives til.

g) (3%) Lag en metode som leser feltvariablenes verdier fra en tekstfil. Metoden skal ha en filstrøm av klassen `BufferedReader` som parameter, og vi antar at strømmen er koblet til filen det skal leses fra.

Et *fylke* er kjennetegnet av et navn og kommunene det består av, og i denne oppgaven skal vi lage en klasse for et fylke. Vi vil lage klassen slik at antall kommuner ikke behøver å være det samme i alle fylker. Innenfor ett fylke kan antall kommuner også forandre seg over tid.

I hver av deloppgavene h-n nedenfor spør vi etter feltvariable, konstruktører eller metoder til klassen. Les gjennom alle deloppgavene før du begynner å svare. Hver konstruktør/metode må påse at feltvariablene ikke tildeles meningsløse verdier.

h) (3%) Foreslå feltvariable for klassen.

i) (3%) Lag en konstruktør som oppretter et fylke. Navnet på fylket og antallet kommuner det skal settes av plass til gis som parametre.

j) (3%) Lag en metode som henter ut referansen til en kommune med oppgitt tabellindeks (metoden skal returnere `null` hvis indeksen er ugyldig). Tabellindeksen til kommunen gis som parameter.

k) (3%) Lag en metode som legger en ny kommune til fylket. Kommunenavnet og de tre innbyggerantallene til kommunen gis som parametre.

l) (3%) Lag en metode som fjerner en kommune fra fylket. Tabellindeksen til kommunen som skal fjernes gis som parameter.

m) (3%) Lag en metode som skriver antall kommuner i fylket samt feltvariabelverdier for hver kommune til en tekstfil. Navnet på tekstfilen gis som en `String`-parameter.

n) (3%) Lag en metode som slår sammen de to minste kommunene i fylket til en ny kommune. Vi ser bort fra geografiske hensyn når vi skal finne hvilke kommuner som skal slåes sammen, kun størrelse i form av antall innbyggere teller. Den nye kommunen skal få navn sammensatt av de to sammenslåtte kommunene sine navn (det største av de nevnt først). Eksempel: Fylket med navn "Hordaland" består av 26 kommuner, hvor "Fedje" (nest minst) og "Modalen" (aller minst) er de med færrest innbyggere. Etter sammenslåing har "Hordaland" bare 25 kommuner, nemlig de 24 som ikke er berørt av sammenslåingen samt en ny kommune med navn "Fedje/Modalen".

Vedlegg A: Nyttige metoder som finnes i klassen Terminal

```
/** Leser en int-verdi fra terminalen. */
 public static int lesInt()

/** Leser en double-verdi fra terminalen. */
 public static double lesDouble()

/** Leser en linje fra terminalen og returnerer
  * denne som et objekt av typen String */
 public static String lesString()
```

Vedlegg B: Klassen `Terning`

```
/** Klasse som representerer en terning.
  * Når vi kaster terningen får vi en tilfeldig verdi
  * mellom 1 og ANTALL_SIDER */
public class Terning {

  /** Antall sider terningen har */
  private static final int ANTALL_SIDER = 6;

  /** Verdien vi fikk sist vi kastet terningen */
  private int verdi;

  /** Oppretter en terning. Vi setter 1 som siste verdi vi har fått */
  public Terning() {
    verdi = 1;
  }

  /** Returnerer verdien vi fikk sist vi kastet terningen */
  public int hentVerdi() {
    return verdi;
  }

  /** Kaster terningen på nytt, og returnerer verdien vi får */
  public int kast() {
    verdi = 1 + (int)(Math.random()*ANTALL_SIDER);
    return verdi;
  }

}
```