PRØVEEKSAMEN –(rettet versjon 27/11)

Eksamen i : INF1000	Grunnkurs i objektorientert programmering
Eksamensdag :	Onsdag 26 nov. 2003.
Tid for eksamen :	1400-1700
Oppgavesettet er på :	13 sider
Vedlegg :	intet
Tillatte hjelpemidler :	Alle trykte og skrevne

- Les gjennom hele oppgaven før du begynner å løse oppgaven. Kontroller at oppgavesettet er komplett før du begynner å besvare det. Dersom du savner opplysninger i oppgaven, kan du selv legge dine egne forutsetninger til grunn og gjøre rimelige antagelser, så lenge de ikke bryter med oppgavens "ånd". Gjør i såfall rede for forutsetningene og antagelsene du gjør.

- Dine svar skal skrives på disse oppgavearkene, og ikke på separate ark. Dette gjelder både spørsmål med avkrysningssvar og spørsmål hvor du bes om å skrive programkode. I de oppgavene hvor det skal skrives programkode, anbefales det at du først skriver en kladd på eget ark før du fører svaret inn i disse oppgavearkene på avsatt plass.

- Endel av spørsmålene er flervalgsoppgaver. Sett da kryss for ***alle*** riktige alternativer. På disse oppgavene får du poeng etter hvor mange avkrysningbokser som stemmer overens med fasiten (dvs som har kryss når fasitsvaret er kryss og som står tomme når fasitsvaret er at det skal stå tomt).

- Hvis du har satt et kryss i en avkrysningsboks og etterpå finner ut at du *ikke* ønsket et kryss der, kan du skrive "FJERN" like til venstre for avkrysningsboksen.

- Husk å skrive såpass hardt at besvarelsen blir mulig å lese på alle gjennomslagsarkene, men ikke legg andre deler av eksamensoppgaven under når du skriver (Merk: dette punktet kan du se bort fra nå under prøveeksamen – det gjelder under den ordentlige eksamenen 5. desember, hvor du må skrive på gjennomslagspapir).

Oppgave A (forslag til tidsbruk: 45 min – 60 min)

1) Anta at følgende deklarasjoner utføres:

```
int i = 0;
int j = i;
```

Hva gjelder like etterpå**?**

- ☐ Variabelen i har verdien 0
- ☐ Variabelen j har verdien 0
- ☐ Variablene i og j er blitt til samme variabel
- ☐ Vi har **`j > i`**

2) Anta at følgende deklarasjon utføres:

```
int antall = 3;
```

og at vi deretter utfører en av alternativene nedenfor. Kryss av de alternativene som vil føre til at variabelen **`antall`** får verdien 5:

☐ `++antall; ++antall;`
☐ `antall++; antall++;`
☐ `antall += 2;`
☐ `antall += antall + 2;`
☐ `antall = (int) (3.5 + antall/2);`
☐ `antall = 4 + antall/2;`

3) Anta at følgende setninger utføres:

```
int i = 5;
int j = i--;
i = j + i;
```

Hva er verdien til variabelen **`i`** like etter at setningene over er utført?

Svar: .....

4) Anta at følgende kodelinjer utføres:

```
boolean b;
int i = 13;
int j = 12;
int k = 200;
b = k < i*j && i == (j+1) || i == 14;
```

Hva er verdien til variabelen **`b`** rett etterpå?

☐ `true`
☐ `i*j`
☐ `k`
☐ `false`
☐ `13`

5) Vi har følgende programsetninger:

```
int j = 0;
for (int i = 0; j < 777; i++) {
   j = i;
   System.out.println(”Eksamen”);
}
```

Hvor mange ganger skrives ”Eksamen” ut i løkken ovenfor?

Svar: .....

6) Anta at følgende setninger er utført:

```
int [] a = new int[77];
int stInd =0
int i = 0;
int j;
```

Hvilke av alternativene nedenfor finner indeksen til det ***største*** elementet i arrayen a og legger denne indeksen inn i variabelen **`stInd`**:

☐
```
while (a[i] > 0) {
  stInd = a[i++];
}
```

☐
```
for (i = 0; i < a[i].length; i++) {
   if (j < a[i]) {
      j = a[i];
      stInd = i;
   }
}
```

☐
```
while (a[i++] < a.length) {
   if (stInd > i) {
       stInd =a[i];
   }
}
```

☐
```
while (i < a.length) {
   if(a[i++] > a[stInd]) {
      stInd = i-1;
   }
}
```

☐
```
for (i = 0; i < a.length; ++i) {
   if(a[stInd] > a[i]) {
      stInd = a[i];
   }
}
```

☐
```
for (i = 0; i < a.length; i = i) {
   stInd = stInd + a[i++];
}
```

7) Anta at filen ”ButikkMain.java” inneholder et program med bl.a. klassene **`ButikkMain`**, **`Butikk`** og **`PrisKategori`**. Følgende programsetninger befinner seg i en metode i klassen **`Butikk`**:

```
PrisKategori pk = new PrisKategori();
pk.init(”A”, 12.50);
```

Nedenfor er noen mulige deklarasjoner av metoden **`init`** i klassen **`PrisKategori`**. Vi konsentrerer oss her kun om første linje i metodedeklarasjonen (=metode-hodet), ikke om innholdet i metodene. Kryss av de alternativene som gjør at kallet ovenfor er ***syntaktisk korrekt***:

☐ **`private void init(String kategori, double pris) {...}`**
☐ **`protected void init(String kategori, double pris) {...}`**
☐ **`void init(char kategori, double pris) {...}`**
☐ **`static void init(String kategori, double pris) {...}`**
☐ **`boolean init(String varenavn, double pris) {...}`**

8) Vi har følgende programsetninger:

```
int i = 9;
int j = 8;
int d = 99;
String s = ”9”;

if (j >= i || s.equals(“”+(j+1))) {
     System.out.println(”A”);
} else {
     if (j == i–1) {
          System.out.println(”B”);
     } else {
          System.out.println(” C”);
     }
}
```

Hva blir skrevet ut ovenfor?

- ☐ **A**
- ☐ **B**
- ☐ **C**
- ☐ **AB**
- ☐ **AC**

9) Anta at vi har deklarert følgende

```
class Oppg9{
  static int s = 99;

  static void giVerdi(int k, int p) {
        k++;
        p = s;
        s = k + 2;
     }

  public static void main (String[] args){
          int s = 13;
          int k = 14;
          giVerdi(s-2,13);
          System.out.println(”Nå er s = ” + s);
     }
}
```

Hva skrives ut i den siste setningen i main?

- ☐ **Nå er s = s**
- ☐ **Nå er s = 0**
- ☐ **Nå er s = 13**
- ☐ **Nå er s = 14**
- ☐ **Nå er s = 15**
- ☐ **Nå er s = 16**
- ☐ **Nå er s = k**

10) Anta at vi har følgende program:

```
class Person {
   private static String navn = "";

   Person (String navn) {
      this.navn = navn;
   }

   String fåNavn() {return navn;}
}

class Oppgave {
   public static void main (String [] args) {
     Person s = new Person("Grete");
     Person p = new Person("Jens");
     System.out.println(s.fåNavn() + ” ” + p.fåNavn());
   }
}
```

Hva skriver programmet ovenfor ut på skjermen?

Svar: .....

11) Vi har følgende programkode:

```
class Test {
    int i = 99;

    int a(int i) {this.i = i; return b(i*2)+ 2;}
    int b(int i) {i = c(i+1); return i;}
    int c(int k) {return i+k;}

    public static void main (String [] args) {
        int k = new Test().a(2);
        System.out.println(”Verdi: ” + k);
    }
 }
```

Hva blir utskriften fra programmet ovenfor?

- ☐ **Verdi: 2**
- ☐ **Verdi: 9**
- ☐ **Verdi: 12**
- ☐ **Verdi: 24**
- ☐ **Verdi: 25**
- ☐ Ingenting – programmet vil ikke kompilere

Oppgave B (forslag til tidsbruk: 2t – 2t 15 min)

1) Metoden **`finnMax`** nedenfor skal returnere med verdien av det største elementet i arrayen **`a`**, hvor **`a`** kan antas å inneholde minst en verdi (dvs lengden av **`a`** er positiv). Skriv ferdig metoden.

```
double finnMax (double [] a) {




}
```

2) Petter har tenkt å lage en spørreundersøkelse om hvor fornøyde studentene har vært med første semester på Blindern. For å gjøre en grundig undersøkelse, ønsker han å finne ut om studenter med ulik bakgrunn har ulike oppfatninger om undervisningen. I tillegg til spørsmålene om hvordan de synes undervisningen har vært, tenker han å registrere følgende opplysninger om hver av de som er med på spørreundersøkelsen: brukernavn, adresse, fødselsnummer (11 siffer), telefonnummer, statsborgerskap, yrke og sivil stand (ugift/gift/samboer), forelder/ikke forelder, gjeld til lånekassa og hva man stemte ved siste valg.

Vil et register med ovenstående opplysninger om førstesemester-studentene kreve konsesjon?

- ☐ Nei
- ☐ Ja, på grunn av følgende paragrafer og prinsipper i loven om personvern (det holder å nevne hvorfor dette evt. krever konsesjon –selve § nummerene er ikke viktig):

….....

3) Nedenfor finner du et uferdig program:

```
class Tomter {
   public static void main(String[] args) {
       // Opprett en ny tomt med 212 som gårdsnummer, 33 som bruksnummer,
       // og med angitt adresse og eier:
       Tomt s1 = new Tomt(212, 33, "Eriks vei 3", "Per Jensen");
       // Opprett en ny tomt med 342 som gårdsnummer, 22 som bruksnummer, og med
       // adresse og eier begge satt til "IKKE REGISTRERT" :
       Tomt s2 = new Tomt(342, 22);
   }
}

class Tomt {
    String eier;
    String adresse;
    int gårdsnummer;
    int bruksnummer;

    // Her skal du føye til noe
}
```

Skriv den programkoden som må føyes til klassen **`Tomt`** for at instruksjonene i main-metoden skal fungere slik kommentarene i programmet angir:

4) Betrakt følgende uferdige program:

```
class MittRegister {
   public static void main (String [] args) {
        Register reg = new Register();
        reg.ordreløkke();
   }
}

class Register {
   private HashMap biler = new HashMap();

   void ordreløkke() {...}

   void leggInnBil() {
       // Les inn reg.nr, eier og reg.år fra terminal, opprett nytt Bil-objekt og legg
       // objektet inn i HashMapen 'biler'.
   }

   void taUtBil() {
       // Les inn reg.nr fra terminal og fjern objektet med dette reg.nr fra HashMapen
       // 'biler', eller gi feilmelding hvis ingen slik bil finnes i registeret.
   }

   void skrivAlleBiler(String filnavn) {
       // Skriv ut all info om alle biler til en fil med angitt filnavn
   }
}

class Bil
   private String regNr;   // Registreringsnr, f.eks. "AX25333"
   private String eier;    // Navn på eier
   private int regÅr;      // Hvilket år bilen ble registrert

   private static Out skjerm = new Out();

   Bil(String regNr, String eier, int regÅr) {
        this.regNr = regNr;
        this.eier = eier;
        this.regÅr = regÅr;
   }
   void skrivUt() {
        skjerm.outln("Registreringsnr: " + regNr);
        skjerm.outln("Eier: " + eier);
        skjerm.outln("Registringsår: " + regÅr);
   }

   void skrivUt(Out s) {
        s.outln("Registreringsnr: " + regNr);
        s.outln("Eier: " + eier);
        s.outln("Registringsår: " + regÅr);
   }

}
```

Du skal i denne oppgaven skrive ferdig en del av programmet ovenfor. Se neste side.

4-A) Skriv metoden leggInnBil()

4-B) Skriv metoden taUtBil()

4-C) Skriv metoden skrivAlleBiler(String filnavn):

5) Du har følgende problemstilling: Et kurs (beskrevet med en kurskode, antall studenter og antall studiepoeng), har flere studenter (hver beskrevet med navn og studentnummer). Begrepene Kurs og Student representeres her naturlig med hver sin klasse.

5-A) Tegn et UML klassediagram med antall på forholdet mellom disse klassene.

5-B) Nedenfor er en skisse til et lite program som inneholder de to klassene diskutert ovenfor, i tillegg til en klasse med en main-metode:

```
class Studenter {
   public static void main (String [] args) {
     Kurs k = new Kurs("INF1000", 10, "studenter.txt");
   }
}

class Kurs {
     ....
}


class Student {
   ....
}
```

Skriv ferdig klassene Kurs og Student. Kurs-klassen skal i tillegg til de objektvariablene som er nevnt tidligere (kurskode, antall studenter og antall studiepoeng) inneholde en HashMap 'studenter' som holder rede på alle studentene som tar kurset. Idet et Kurs-objekt opprettes skal alle objektvariablene få verdier, og HashMapen 'studenter' skal også fylles opp med

informasjon om alle studentene som følger kurset. Informasjon om studenter skal leses fra filen med navnet som oppgis når Kurs-objektet opprettes. Hver linje i filen inneholder informasjon om en enkelt student og består av (1) et studentnummer; (2) hvor mange ganger studenten har vært meldt opp i kurset før; og (3) studentens navn. Opplysning (2) er ikke relevant for vårt formål og skal ikke tas vare på. Her er et eksempel på starten av en slik fil (antall studenter som det er informasjon om i filen er ikke kjent på forhånd):

Filen studenter.txt:

```
345653      0      Petter Jensen
363322      0      Arne Hansen
366444      2      Kari Olsen
842234      0      Varg Veum
        .... osv ....
```

Merk: det er ikke meningen at du skal lage noen ordreløkke eller på annen måte utvide programmets funksjonalitet ut over det som det direkte bes om ovenfor.

6) Anta at vi har behov for å lagre mange objekter av klassen Person, med personens fødselsnummer (en tekststreng) som nøkkel. Vi kunne brukt en HashMap til dette formål, men her skal vi se hvordan vi kunne laget vår egen variant av klassen HashMap ved bruk av arrayer.

```
class PersonTabell {
   private String [] nøkler;
   private Person [] personer;
   private int antall;


   PersonTabell(int maxAntall) {
      personer = new Person[maxAntall];
       nøkler = new String[maxAntall];
      antall = 0;
   }

   void put(String nøkkel, Person p) {
      // Legg inn (nøkkel, p) i tabellen
   }

   Person get(String nøkkel) {
      // Returner med peker til Person-objektet som ble lagt inn
      // med den gitte nøkkelen
   }

   Person [] getAllValues() {
      // Returner med arrayen av Person-pekere, men bare
      // den delen av arrayen som er fylt opp med Person-pekere
   }

   boolean containsKey(String nøkkel) {
      // Returner med true hvis nøkkelen ligger i arrayen 'nøkler'
      // og med false ellers.
   }
}
```

Skriv ferdig metodene put, get, getAllValues og containsKey i programskissen ovenfor.