Kandidatnummer:_____

Bokmål

UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

Prøveeksamen 2014

Prøveeksamen i :	INF1000 — Grunnkurs i objektorientert programmering
Prøveeksamensdag :	Onsdag 12. november 2014
Tid for eksamen :	10:15 til 14:15 – 4 timer
Oppgavesettet er på :	5 sider
Vedlegg :	Ingen
Tillatte hjelpemidler :	Alle trykte og skrevne

- Kontroller at oppgavesettet er komplett, og les nøye gjennom oppgavene før du løser dem.
- Du kan legge dine egne forutsetninger til grunn og gjøre rimelige antagelser, så lenge de ikke bryter med oppgavens "ånd". Gjør i så fall rede for disse forutsetningene og antagelsene.
- Poengangivelsen øverst i hver oppgave angir maksimalt antall poeng.Sammenlagt gir alle oppgavene maksimalt 100 poeng. Delsummene i parentes antyder poeng per deloppgave. Unngå å bruke en stor del av tiden din på oppgaver som gir deg få poeng.
- Svarene skal skrives på gjennomslagspapir. Skriv hardt nok til at besvarelsen blir mulig å lese på alle gjennomslagsarkene, og ikke legg andre deler av eksamensoppgaven under når du skriver.
- Du beholder selv underste ark etter levering av de to øverste til eksamensinspektøren. Nummerer sidene, og husk å skrive kandidatnummeret ditt på besvarelsen.

Oppgave 1 (1+2+2 poeng)

a) Hva er verdien til **tall** etter at følgende kode er utført?

```
int tall = 5;
tall = (tall*3) + 2;
```

b) Anta at følgende programsetninger utføres. Hva skrives ut på skjermen?

```
int i = 11;
int j = i;
int k = 32;
if (k > j * i || k < i) {
   System.out.println("A");
} else {
   if (k < j * i && k > i) {
      System.out.println("B");
   } else {
      System.out.println("C");
   }
}
```

c) Hva skrives ut her?

```
String sekv = "0";
for (int i=0; i<6; i++) {
   sekv = sekv + i;
}
System.out.println ("sekv = " + sekv);
```

Oppgave 2 (2+4 poeng)

a) Gitt følgende kode. Hva returneres fra metodekallet **rest (45,7)**?

```
int rest (int n, int m) {
   while (n >= m) {
      n-=m;
   }
return n;
}
```

b) Anta at følgende program utføres:

```
class Studentregister {
   public static void main (String[] args) {
      Student s = new Student ("Ole", "Karl Johans gt 1");
      Student p = new Student ("Marit", "Karl Johans gt 2");
      System.out.println (s.faaNavn() + " og " + p.faaNavn());
   }
}

class Student {
   private String navn = ”Grete”;
   private String adresse = "Blindernveien 3";
   public Student (String navn, String adresse) {
      this.navn = navn;
      this.adresse = adresse;
   }

// Programmet fortsetter på neste side
```

```
   String faaNavn() {
      return navn;
   }
}
```

Hva blir utskriften på skjermen? Svar med nummer for riktig alternativ.

1. Grete og Grete
2. Ole og Johan
3. Marit og Marit
4. navn og navn
5. Ole og Marit
6. s.fåNavn() og p.fåNavn()
7. Marit og Ole
8. Ingen av alternativene over

Oppgave 3 (4 poeng)

a) Skriv binærtallet 1001 som et desimaltall.
b) Skriv desimaltallet 24 som et binærtall.
c) Skriv summen av de to binærtallene 101 og 110 som et binærtall.
d) Skriv det heksadesimale tallet 3F som et desimaltall.

Oppgave 4 (5 poeng)

Skriv en metode som tar inn tre parametre av typen double. Metoden skal returnere den største av de tre parameterverdiene. Kall metoden for finnStorst.

Eksempel:

```
double v = finnStorst(0, 10.2, 4.7);
```

Dette vil føre til at v får tilordnet verdien 10.2.

Oppgave 5 (8+7 poeng)

a) Du skal skrive en metode med en **int**-array som parameter og som returnerer en **int**-array. Metoden skal opprette en ny **int-**array som er dobbelt så lang som den i parameteren, kopiere over verdiene i parameter-arrayen i første halvdel av den nye arrayen (og la de resterende verdiene i den nye arrayen være 0), og til slutt returnere den nye arrayen.

b) I programmer kan det være behov for å konkatenere (slå sammen) to tekststrenger, slik som vi gjør når vi skriver **String t = s1 + s2**; der **s1** og **s2** er to String-variable. I denne oppgaven skal du lage en metode som gjør det samme, men med flere strenger. Mer presist skal du lage en metode som har to parametre: en String-array og en String. Metoden skal konkatenere hvert enkelt element i String-arrayen med strengen i den andre parameteren, og returnere resultatet i en ny String-array som er like lang som den opprinnelige. Hvis metoden heter **konkatener**, så skal f.eks. setningene

```
String[] ordliste = {"Karoline", "Anders", "Camilla"};
String[] resultat = konkatener(ordliste, " er flink");
```

resultere i at **resultat** blir en String-array av lengde 3 med verdiene "Karoline er flink", "Anders er flink" og "Camilla er flink", henholdsvis.

Oppgave 6 (2+3+15+10+15 poeng)

Nedenfor er skissen til et program som først skal lese en fil med informasjon om endel kontakter (en kontakt består av et navn, en adresse og et telefonnummer) og deretter gå i løkke og be brukeren om et navn og skrive ut all kontaktinformasjon knyttet til dette navnet.

```
import java.util.HashMap;

class TelefonregisterMain {
   public static void main (String[] args) {
      Telefonregister tr = new Telefonregister ();
       tr.lesFraFil(”kontakter.txt”);
       tr.snakkMedBruker ();
   }
}

class Telefonregister {
   private HashMap<String,Kontakt> h = new HashMap<String,Kontakt>();
      // Her skal det ligge to objektmetoder:
      // lesFraFil og snakkMedBruker
}

class Kontakt {
     private String navn, adresse, tlf;
     // Her skal det ligge en konstruktør og en utskriftsmetode
}
```

a) Lag først konstruktøren i klassen **Kontakt**. Konstruktøren skal ha tre parametere (en for hver av objektvariablene i klassen) og skal benytte parameterverdiene til å initiere objektvariablene.

b) Lag utskriftsmetoden i klassen **Kontakt**. Metoden skal hete **skrivUt** og skal ikke ha noen parametere eller returverdi. Den skal skrive ut på skjermen verdiene til de tre objektvariablene i klassen.

c) Lag metoden **lesFraFil** som skal ligge i klassen **Telefonregister** og som blir kalt på fra main-metoden i klassen **TelefonregisterMain**. Metoden **lesFraFil** skal ha et filnavn som parameter og skal lese fra denne filen som antas å inneholde informasjon om et antall kontakter. Filen er en tekstfil som inneholder først tre linjer for den første kontakten, deretter tre linjer for den neste kontakten, osv. De tre linjene for en kontakt inneholder henholdsvis navn, adresse og telefonnummer (i den rekkefølgen). Hver kontakt skal representeres med et objekt av klassen **Kontakt**, og du skal benytte konstruktøren fra punkt a) når du oppretter nye **Kontakt**-objekter. Programmet skal legge kontaktene inn i HashMapen **h** (med navn som søkeord).

d) Lag metoden **snakkMedBruker** som skal ligge i klassen **Telefonregister** og som blir kalt på fra **main**- metoden i klassen **TelefonregisterMain**. Metoden **snakkMedBruker** skal inneholde en evig løkke hvor hvert gjennomløp kun består i at programmet ber om og leser inn et navn fra terminal og deretter skriver ut på skjerm (ved hjelp av metoden **skrivUt**) all informasjon knyttet til dette navnet (eller en feilmelding hvis navnet er ukjent).

Programmet ditt skal nå utvides med en metode for å skrive ut adresselapper for utsending av julekort til alle kontakter – dog med kun ett kort per husstand. Vi antar at alle som har identiske tekststrenger som sin adresse, er i samme husstand. Vi antar også at metoden **snakkMedBruker** vil bli utvidet til å

håndtere en kommando for å skrive ut slike adresselapper (du skal ikke gjøre endringer i **snakkMedBruker**, bare anta at det er på plass).

e) Deklarer en klasse **Husstand** med et passende grensesnitt og innhold for å kunne holde rede på kontakter som bor på samme adresse.

f) Skriv en ny objektmetode **skrivAdresser** i klassen **Telefonregister** som oppretter og tar vare på objekter av klassen **Husstand** (ett for hver adresse i telefonregisteret), og deretter skriver ut på skjerm (i en senere versjon som ikke inngår i eksamen vil det skrives ut på fil) en tenkt adresselapp for hver husstand etter følgende mønster:

   - En linje med hvert av navnene på den aktuelle adressen
   - En avsluttende linje med husstandens adresse
   - En blank linje

   Programmet vil ikke ha behov for Husstand-objektene etter at alle adresselappene er skrevet ut.

Oppgave 7 (10 poeng)

Skriv en metode med en tekststreng som parameter. Metoden skal skrive en rekke linjer ut på skjermen, etter følgende mønster: Første linje skal være den samme som teksten i parameteren. Linje to inneholder samme tekst som første linje, men det første tegnet skal stå sist, og de resterende tegnene skal flyttes en posisjon frem. Dette gjentas inntil første tegn på den siste linjen er det som var siste tegnet på den første linjen. Du stopper altså utskriften rett før du ville skrevet ut en linje som er maken til den første linjen.

Oppgave 8 (10 poeng)

Restaurant "Den Sultne Akademiker" ønsker å øke omsetningen. De tenker da å sette sammen en del informasjon de allerede har fra kundene; navn, kredittkortnummer, hvilke menyer de har bestilt og hvor mye maten kostet. Denne informasjonen supplerer restauranten så med informasjon de finner på internett, som adresse, yrke og skatteinformasjon fra skattelistene.

Restauranten har så tenkt å lage månedens "to-på-topp" ved å gi et gratis måltid til både den gjesten som forrige måned har spist mest i forhold til sin skattbare inntekt og den som har brukt absolutt mest penger sist måned på "Den Sultne Akademiker". Navnet på de to vinnerne vil bli lagt ut på hjemmesida til restauranten og tilskrevet personlig.

Kan restauranten fritt lagre, bruke og offentliggjøre slike opplysninger? Redegjør kort for hvordan Personopplysningsloven regulerer denne typen bruk av persondata. Nevn de relevante paragrafene som bør tas med i vurderingen med begrunnelse for hvorfor de er relevante i denne sammenheng.