Bokmål

UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

Eksamen i:	INF1000 — Grunnkurs i objektorientert programmering
Eksamensdag:	Fredag 4. desember 2015
Tid for eksamen:	14.30 (4 timer)
Oppgavesettet er på:	8 sider
Vedlegg:	Ingen
Tillatte hjelpemidler:	Alle trykte og skrevne

1. Kontroller at oppgavesettet er komplett, og les nøye gjennom oppgavene før du løser dem.
2. 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.
3. Poengangivelsen øverst i hver oppgave angir maksimalt antall poeng. Sammenlagt gir alle oppgavene maksimalt **100** poeng. Unngå å bruke en stor del av tiden din på oppgaver som gir deg få poeng.
4. 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.
5. Du beholder selv underste ark etter levering av de to øverste til eksamens-inspektøren.

Oppgave 1 (3 poeng)

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

```
int tall = 3+2+1;
tall = tall*2;
```

12

b) Hva skrives ut på skjermen når følgende kode utføres?

```
int a = 10;
int b = 1;
while (a>0) {
  b=b*2;
  a=a-b;
}
System.out.println("a=" + a);
System.out.println("b=" + b);
```

Oppgave 2 (4 poeng)

Vi har en metode **doble** som vist nedenfor:

```
int doble(int a){
  a = a*2;
  return a;
}
```

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

```
int a = 2;
int b = doble(a);
```

2

b) Hva er verdien til **b** etter at følgende kode er kjørt?

```
int a = 2;
int b = doble(a+1);
```

6

Oppgave 3 (4 poeng)

a) Skriv binærtallet 10001010 som et desimaltall. 138
b) Skriv desimaltallet 17 som et binærtall. 10001
c) Skriv summen av de to binærtallene 0100 og 1111 som et binærtall. 10011
d) Skriv det heksadesimale tallet A7 som et desimaltall. 167

Oppgave 4 (7 poeng)

a) Skriv en metode **double beregnAreal (double lengde, double bredde)** som regner ut arealet av et rektangel ved å multiplisere lengde og bredde, og returnerer resultatet. Se under b)

b) Endre metoden slik at verdien -1 returneres hvis oppgitt lengde eller bredde er et negativt tall.

```
static double beregnAreal (double lengde, double bredde) {
  if ((lengde<0) || (bredde < 0)) {
    return -1;
  }
  return lengde*bredde;
}
```

Oppgave 5 (5 poeng)

Hva skrives ut om du kjører dette programmet?

```
class TestMinKlasse {

  public static void main(String[] args){
    MinKlasse mittObjekt = new MinKlasse("AB");
    String tekst = mittObjekt.hentBeskjed();
    mittObjekt.inkrementerVerdi();
    mittObjekt.settTekst(tekst);
    mittObjekt.inkrementerVerdi();
    System.out.println(mittObjekt.hentBeskjed());
  }
}

class MinKlasse {
  private int minVerdi;
  private String minTekst;

  public MinKlasse(String startTekst){
    minVerdi = 2;
    minTekst = startTekst;
  }

  public void inkrementerVerdi(){
    minVerdi += 4;
  }

  public void settTekst(String tekst){
    minTekst = tekst;
  }

  public String hentBeskjed(){
    return minTekst + minVerdi;
  }
}
```

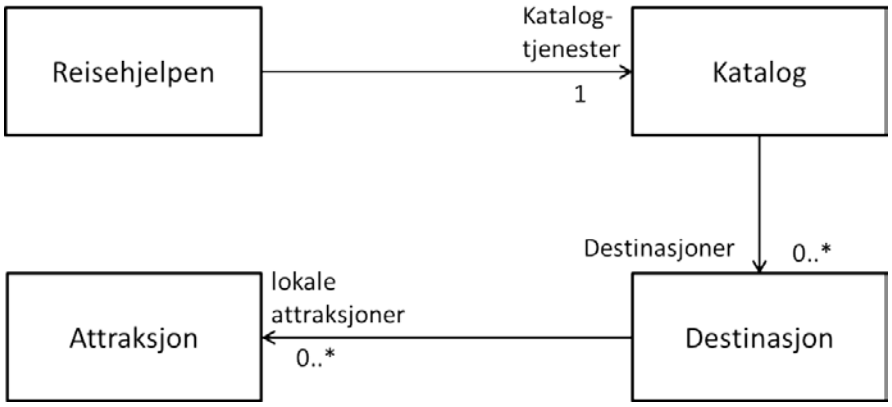
Oppgave 6 (5 poeng)

Skriv en metode **String kampResultat(int scoringerLagA, int scoringerLagB).** Metoden skal returnere "hjemme" dersom **scoringerLagA** er høyere enn **scoringerLagB**, "borte" dersom **scoringerLagB** er høyere enn **scoringerLagA**, og "uavgjort" dersom **scoringerLagA** er lik **scoringerLagB**.

```
static String kampResultat(int scoringerLagA, int scoringerLagB){
  if (scoringerLagA > scoringerLagB) {
    return "hjemme";
  }
  if (scoringerLagA < scoringerLagB) {
    return "borte";
  }
  return "uavgjort";
}
```

Oppgave 7 (50 poeng)

Du er blitt bedt om å programmere deler av nettjenesten «Reisehjelpen» for planlegging av opplevelsesreiser for turister. Tjenesten bruker en klasse Katalog til å holde orden på destinasjoner og finne frem informasjon om de attraksjonene hver destinasjon kan by på. UML-diagrammet viser klassene som brukes av Reisehjelpen og deres relasjoner.



Om du hopper over en deloppgave er det likevel viktig at du leser hele teksten. Du kan fritt bruke klasser og metoder som er oppgitt tidligere i oppgaven selv om du ikke har skrevet dem.

Alle datoer representeres i programmet som heltall på formen aammdd der aa, mm og dd angir henholdvis årstall, måned og dag (4. desember 2015 lagres dermed som heltallet 151204).

Hver attraksjon har et navn, en sesongstart og en sesongslutt, og kan være egnet eller ikke for barn:

```
class Attraksjon {
  // Datarepresentasjon ikke oppgitt

  // Konstruktør:
  public Attraksjon(String navn,boolean barn, int fra, int til) {.. }

  // Skriv ut alle objektvariables innhold på en linje (det er ikke
  // vesentlig hvordan du velger å formattere dette):
  public void skrivAttr () {.. }

  // Returnerer om en attraksjon er egnet for barn:
  public boolean forBarn () {.. }

  // Returnerer om en attraksjon er åpen minst en av dagene i en gitt
  // periode (inkludert første og siste dag i perioden):
  public boolean aapenIPeriode (int fra, int til) {.. }
}
```

a) 5 poeng
Skriv klassen **Attraksjon** med datarepresentasjon, konstruktør og metodene **skrivAttr** og **forBarn**.

```
class Attraksjon {
  String aNavn;
  int apnes, stenges;
  boolean bVennlig;

  public Attraksjon (String navn,boolean barn, int fra, int til) {
    aNavn = navn;
    apnes = fra;
    bVennlig = barn;
    stenges = til;
  }

  public void skrivAttr () {
    String utskrift = (aNavn + ". " + apnes + "-" + stenges);
    if (bVennlig) {
      utskrift = "BARN: " + utskrift;
    } else {
      utskrift = "VOKSNE: " + utskrift;
    }
    System.out.println (utskrift);
  }

  public boolean forBarn () {
    return bVennlig;
  }

}
```

b) 5 poeng
Skriv metoden **aapenIPeriode** i klassen **Attraksjon.**

```
  public boolean aapenIPeriode (int fra, int til) {
    if ((fra > stenges) || (til < apnes)) {
      return false;
    } else {
      return true;
    }
  }
```

Oppgave 7 (forts)

Klassen **Destinasjon** representerer destinasjoner denne tjenesten kjenner til. Hvert objekt av klassen **Destinasjon** har et navn og en samling attraksjoner som lagres i en **ArrayList**. Navnet og attraksjonene oppgis som parametere til konstruktøren når det opprettes en ny **Destinasjon**.

```
class Destinasjon {
  // Datasrepresentasjon og konstruktør skrives av deg

  // Returnerer destinasjonens navn:
  public String hentDestNavn () {.. }

  // Skriver ut sitt navn og alle sine attraksjoner på terminalen:
  public void skrivDest () {.. }

  // Legger til en ny lokal attraksjon:
  public void leggTilAttr (Attraksjon nyAttr) {.. }

  // Returnerer et tall som angir hvor mange av de lokale
  // attraksjonene som oppfyller et sett med krav (se deloppgave e):
  public int antallAktuelleAttr (boolean barn, int fraDato,
                                 int tilDato){.. }
}
```

c) 3 poeng
Skriv datarepresentasjon og konstruktør for klassen **Destinasjon.**

```
class Destinasjon {
  private String dNavn;
  private ArrayList<Attraksjon> lokaleAttr;

  public Destinasjon (String navn, ArrayList<Attraksjon> attrListe) {
    dNavn = navn;
    lokaleAttr = attrListe;
  }
}
```

d) (7 poeng)
Skriv metodene **hentDestNavn**, **skrivDest** og **leggTilAttr** i klassen **Destinasjon.**

```
  public String hentDestNavn () {
    return dNavn;
  }

  public void skrivDest () {
    System.out.println ("Destinasjon " + dNavn);
    for (Attraksjon a : lokaleAttr) {
      a.skrivAttr();
    }
  }

  public void leggTilAttr (Attraksjon nyAttr) {
    lokaleAttr.add(nyAttr);
  }
```

e) (5 poeng) (vanskelig)
Skriv metoden **antallAktuelleAttr (boolean barn, int fraDato, int tilDato)** i klassen Destinasjon. Metoden skal returnere et tall som angir hvor mange av de lokale attraksjonene som oppfyller et sett med krav etter følgende regler:

- Hvis parameteren **barn** er **true** skal kun attraksjoner som egner seg for barn regnes med (alle egner seg for voksne)
- Kun attraksjoner som er åpne minst en dag i perioden fra og med **fraDato** til og med **tilDato** skal regnes med

```
public int antallAktuelleAttr (boolean barn, int fra, int til){
  int antall = 0;
  for (Attraksjon a : lokaleAttr) {
    if ((barn && a.forBarn()) || (!barn)) {
      if (a.aapenIPeriode(fra, til)) {
        antall++;
      }
    }
  }
  return antall;
}
```

Oppgave 7 (forts)

Klassen **Katalog** holder orden på alle destinasjoner og tilbyr ulike operasjoner på disse:

```
class Katalog {
  private HashMap<String, Destinasjon> destKatalog = new HashMap<>();

  // Konstruktør. Oppretter alle destinasjonene med navn og
  // de tilhørende attraksjonene ved å lese data fra fil:
  public Katalog (String katalogfil) throws Exception {
    lesDestinasjonsfil(katalogfil);
  }

  // Metode som kalles fra konstruktøren for å lese inn katalogdata.
  // Denne skal du ikke skrive:
  private void lesDestinasjonsfil (String filnavn) throws Exception{}

  // Metode som skriver ut navnet på alle destinasjoner i katalogen:
  public void skrivDestListe () {.. }

  // Metode som skriver ut navn og informasjon om alle attraksjoner
  // på den oppgitte destinasjonen:
  public void skrivEnDest (String destNavn) {.. }

  // Metode som legger til en ny attraksjon for en destinasjon.
  // Om destinasjonen ikke finnes skal metoden returnere uten å
  // gjøre noe:
  public void nyAttr (String destNavn, String attrNavn,
                      boolean bVennlig, int apenFra,int apenTil){.. }

  // Metode som leser nye attraksjoner for en destinasjon fra fil.
  // Om destinasjon med angitt navn ikke eksisterer fra før opprettes
  // en ny destinasjon, ellers legges attraksjonene til den
  // eksisterende destinasjonen. Filformat beskrives i deloppgave h):
  public void nyDestFraFil (String destNavn, String filnavn)
                            throws Exception {.. }

  // Metode som går gjennom alle destinasjoner, og returnerer navnet
  // på den destinasjonen som har flest attraksjoner som
  // tilfredsstiller krav som beskrevet i oppgave e):
  public String finnBesteDest (boolean barn, int fra, int til) {.. }
}
```

Oppgave 7 (forts)

f) 5 poeng
Skriv metodene **skrivDestListe** og **skrivEnDest** i klassen **Katalog**.

```
public void skrivDestListe () {
  System.out.println ("Destinasjoner i katalogen:");
  for (Destinasjon d : destKatalog.values()) {
    System.out.println (d.hentDestNavn());
  }
}

public void skrivEnDest (String dNavn) {
  Destinasjon dest = destKatalog.get(dNavn);
  if (dest != null) {
    dest.skrivDest();
  } else {
    System.out.println ("Ingen destinasjoner med navnet " + dNavn);
  }
}
```

g) 5 poeng
Skriv metoden **nyAttr** i klassen **Katalog**.

```
public void nyAttr (String destNavn, String attrNavn,
                    boolean bVennlig, int apenFra, int apenTil) {
  Destinasjon dest = destKatalog.get (destNavn);
  if (dest != null) {
    Attraksjon nyAttr = new Attraksjon (attrNavn, bVennlig,
                                        apenFra, apenTil);
    dest.leggTilAttr(nyAttr);
  }
}
```

h) 7 poeng
Skriv metoden **nyDestFraFil** i klassen **Katalog.** Filen som skal leses inneholder en eller flere attraksjoner som tilhører samme destinasjon. For hver attraksjon ligger det *alltid* 4 linjer på følgende format, en linje for hver attributt. Eksempelet viser en fil for destinasjon Oslo:

```
Slottet
VOKSNE
150101
161231
Frognerbadet
BARN
150517
150820
```

```
public void nyDestFraFil (String destNavn, String filnavn)
                          throws Exception {
  if (destKatalog.get(destNavn) == null) {
    Destinasjon d = new Destinasjon (destNavn,
                                     new ArrayList<Attraksjon>());
    destKatalog.put (destNavn, d);
  }

  File fil = new File(filnavn);
  Scanner attrFil = new Scanner(fil);

  while (attrFil.hasNextLine()) {
    String nvn = attrFil.nextLine();
    String voksnebarn = attrFil.nextLine();
    boolean barn = (voksnebarn.equals("BARN"));
    int fra = Integer.parseInt (attrFil.nextLine());
    int til = Integer.parseInt (attrFil.nextLine());
    nyAttr (destNavn, nvn, barn, fra, til);
  }
}
```

i) 8 poeng (vanskelig)
Skriv metoden **finnBesteDest** i klassen **Katalog**.

```
public String finnBesteDest (boolean barn, int fra, int til) {
  int maxTreff = 0;
  String besteHittil = "";

  for (Destinasjon d : destKatalog.values()) {
    int antTreff = d.antallAktuelleAttr(barn, fra, til);
    if (antTreff > maxTreff) {
      maxTreff = antTreff;
      besteHittil = d.hentDestNavn();
    }
  }

  return besteHittil;
}
```

Oppgave 8 (7 poeng)

På et sosialt nettsted er det oppstått en åpen gruppe med tittelen «Vi med nakkesleng-skader», der det i løpet av kort tid publiseres en rekke bidrag fra deltakere som deler sine helseproblemer og erfaringer fra helsevesenet under fullt navn.

a) Diskuter om informasjonen på nettstedet kommer inn under Personvernloven (POPPL) og om det i så fall foreligger hjemmel for offentliggjøring. Vis til relevante paragrafer.

Iht til POPPL §2.1) og 2.2) er dette behandling av personopplysninger. Iht POPPL §2.8) er det også snakk om sensitive personopplysninger. Hjemmel: Iht §8 kan personopplysninger bare behandles dersom den registrerte har samtykket – dette ansees dekket i og med at offentliggjøring er gjort på eget initiativ. §9 omhandler tilleggskrav for sensitive personopplysninger, her tilfredsstilles punkt d) av samme grunn som over. Konklusjon: JA det kommer inn under POPPL, JA det finnes hjemmel.

b) Forsikringsselskapet Sikkert tilbyr forsikringer mot blant annet uførhet til privatpersoner. For å holde risiko og dermed premiene på forsikringen lave samler selskapet inn opplysninger fra nettsteder som det beskrevet ovenfor i et eget pasientregister med opplysninger om kjente tilstander hos potensielle fremtidige forsikringstakere. Nye kunder blir sjekket mot dette for å avdekke eventuelle skader og sykdommer før de mottar tilbud om forsikring fra selskapet Sikkert. Har forsikringsselskapet anledning til å etablere et slikt register basert på offentlig tilgjengelig informasjon? Begrunn svaret med henvisning til relevante paragrafer i POPPL.

Dette har Sikkert ikke adgang til. De registrerte har ikke samtykket til bruk av informasjonen til andre formål og det foreligger heller ikke hjemmel etter andre deler av §8. Sikkert kan dermed ikke behandle verken sensitive eller andre typer personopplysninger hentet fra dette eller tilsvarende nettsteder (med mindre de går inn i en dialog med de aktuelle personene og innhenter samtykke).

Oppgave 9 (15 poeng)

a) (7 poeng)
Skriv en metode **trimZeros(int[] a)** som tar inn en array av **int**-verdier og returnerer en **int**-array hvor alle (eventuelle) nuller i starten og slutten av arrayen er fjernet. Dersom det er nuller inne i arrayen (med andre tall foran og bak seg) skal disse ikke fjernes. Gitt en array {0,0,1,2,0,3,0,0,4,0} som parameterverdi for a, skal den altså returnere en array {1,2,0,3,0,0,4}. Effektiviteten av løsningen blir ikke tillagt vekt, formålet er kun at koden skal gi ønsket resultat.

```
static int[] trimZeros(int[] a) {
  if ((a == null) || (a.length == 0)) return new int[0];

  int i = 0;
  while ((i<a.length) && a[i] == 0) {
    i++;
  }

  int fjernet = i;
  i = a.length-1;
  while ((i > fjernet) && (a[i] == 0)) {
    i--;
  }

  int[] tmp = new int[i-fjernet+1];
  for (i=0; i<tmp.length; i++) {
    tmp[i] = a[i+fjernet];
  }
  return tmp;
}
```

b) (8 poeng)
Skriv en metode som tar inn en verdi av type **String** som parameter og som returnerer en verdi av type **int**. Metoden skal telle antall ulike bokstaver i String-verdien den fikk inn som parameter og returnere dette antallet. Hvis metoden heter **telling**, så skal f.eks. setningen.

```
int v = telling (“accag”);
```

føre til at variabelen **v** blir tilordnet verdien 3.

Det er i denne oppgaven lov å bruke klassene String, ArrayList og HashMap om man ønsker, men ikke andre klasser fra Java APIet.

```
static int telling (String str) {
  if (str.length() < 2) {
    return str.length();
  }

  HashMap<String, String> ant = new HashMap<String, String>();
  for (int i = 0; i<(str.length()); i++) {
    String tmp = str.substring (i, i+1);
    if (!ant.containsKey(tmp)) {
      ant.put (tmp,tmp);
    }
  }
  return ant.size();
}
```