UNIVERSITETET I OSLO

Kandidatnr

Det matematisk-naturvitenskapelige fakultet

Eksamen i	INF1000 — Grunnkurs i objektorientert programmering
Eksamensdag:	Mandag 30. november 2009
Tid for eksamen:	14.30 – 17.30
Oppgavesettet er på 20 sider.	
Vedlegg:	Ingen
Tillatte hjelpemidler:	Alle trykte og skrevne

Kontroller at oppgavesettet er komplett før
du begynner å besvare spørsmålene.

LØSNINGSFORSLAG

- Les nøye gjennom hver oppgave før du løser den. For hver oppgave er det angitt det maksimale antall poeng du kan få hvis du svarer helt riktig. Summen av poengene er 100, slik at f.eks 5 poeng tilsvarer omlag 9 minutter, 10 poeng omlag 18 minutter, osv. (hvis du regner med å komme igjennom alt). Pass på at du bruker tiden din riktig!
- Kontroller også 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 så fall 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. Det er satt av plass i oppgavesettet slik at du kan skrive inn svarene dine.
- 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.
- Noen av spørsmålene er flervalgsoppgaver. På disse oppgavene får du poeng etter hvor mange korrekte svar du gir. Du får *ikke* poeng hvis du lar være å besvare et spørsmål, eller dersom du krysser av begge svaralternativer.
- Hvis du har satt et kryss i en avkrysningsboks og etterpå finner ut at du ikke ønsket å krysse av der, kan du skrive "FEIL" like til venstre for den aktuelle 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.*

(Fortsettes på side 2.)

Innhold

Lykke til!

(Fortsettes på side 3.)

Oppgave 1 Løkker (5 poeng)

Hvilken verdi har **int**-variabelen `tall` etter hver av løkkene under?

1a

```
int tall = 0;
for (int i=0; i<10; i++) {
    tall++;
}
```

Svar: 10

1b

```
int tall = 0;
while (tall < 10) {
    for (int i=10; i>0; i--) {
        tall = tall + 2;
    }
}
```

Svar: 20

1c

```
int tall = 0;
do {
    for (int i=0; i<20; i++) {
        tall++;
    }
} while (tall < 10)
```

Svar: 20

1d

```
int tall = 0;
for (int i=5; i>0; i--) {
    for (int j=i; j>0; j--) {
        tall--;
    }
}
```

Svar: -15

(Fortsettes på side 4.)

Oppgave 2 Array og løkker (5 poeng)

Anta at vi har deklarert en **boolean**-array `janei`, og at alle plassene i arrayen har fått tildelt **boolean**-verdier. Anta videre at vi har deklarert en **int**-variabel `antallTrue`. Løkkene nedenfor er forsøk på å telle opp antall plasser i `janei` som har verdien **true**, ved hjelp av variabelen `antallTrue`. Er løkkene korrekte?

Ja	Nei	

☒	☐	`antallTrue = 0;`
		`for (int i=0; i<janei.length; i++) {`
		`  if (janei[i]) {`
		`  antallTrue++;`
		`  }`
		`}`

☒	☐	`antallTrue = janei.length;`
		`for (int i=janei.length; i>0; i--) {`
		`  if (!janei[i-1]) {`
		`  antallTrue--;`
		`  }`
		`}`

☐	☒	`antallTrue = 1;`
		`for (int i=1; i<janei.length; i++) {`
		`  if (janei[i-1]) {`
		`  antallTrue = antallTrue+i;`
		`  }`
		`}`

☐	☒	`antallTrue = 0;`
		`for (int i=0; i<janei.length; i=i+2) {`
		`  if (janei[i]) {`
		`  antallTrue++;`
		`  }`
		`}`

☒	☐	`antallTrue = -1;`
		`for (int i=-1; i+1<janei.length; i++) {`
		`  if (janei[i+1]) {`
		`  antallTrue++;`
		`  }`
		`}`
		`antallTrue++;`

(Fortsettes på side 5.)

Oppgave 3 Java-syntaks (5 poeng)

Er disse uttrykkene lovlige programsetninger i Java?

Ja	Nei	
☒	☐	`int i = (int) 1000;`
☒	☐	`int j = 'c';`
☐	☒	`boolean verdi = "INF1000";`
☒	☐	`String kurs = "INF1000";`
☐	☒	`int k = -1.0;`
☒	☐	`float f = -1.0F;`
☐	☒	`String[] tekst = { 1, 2, 3 };`
☐	☒	`String bokstav = 'a';`
☒	☐	`return "abc".equals("bcd");`
☒	☐	`boolean resultat = (resultat = true) == false;`

Oppgave 4 String (5 poeng)

Anta at følgende variable er deklarert:

```
String a = "kurator";
String b = "kur";
String c = "ator";
```

Hva er verdien til følgende **boolean**-uttrykk?

true	false	
☐	☒	`a == (b+c)`
☒	☐	`a.equals(b+c)`
☒	☐	`a.startsWith(b)`
☐	☒	`a.compareTo(b) < 0`
☐	☒	`a.indexOf(c) == 4`

(Fortsettes på side 6.)

Oppgave 5 Metoder (10 poeng)

5a

Nedenfor er det deklarert en metode `sum(int[] array)`, som skal beregne summen av tallene i en **int**-array. Vil metoden kompilere? Hvis nei, forklar hva som må endres for at metoden skal kompilere.

```
public void sum(int[] array) {
    int sum = 0;
    for (int i=0; i<array.length: i++) {
        sum += array{i};
    }
    return sum;
}
```

Skriv svaret ditt her:

Tre syntaksfeil:

I linje 3 brukes ':' i stedet for ';' etter betingelsen i for-løkka.

I linje 4 brukes '{'/'}' i stedet for '['/']' som paranteser for å angi plass i arrayen.

I linje 6 returneres en int-verdi, selv om metoden er deklarert void. Endre metodens returtype til int.

5b

Deklarér en metode `snitt(int[] array)`, som skal beregne gjennomsnittet av tallene i parameteren `array` av type **int**-array. Snittet skal beregnes med flyttallsdivisjon og returverdien skal være av type **double**. Du skal skrive *hele* metoden, inkludert signaturen.

```
double snitt(int[] array) {
    double snitt = 0.0;
    for (int i=0; i<array.length; i++) {
        snitt += array[i];
    }
    return snitt / array.length;
}
```

(Fortsettes på side 7.)

5c

Vi kan beregne en gjenstands snitthastighet ved å dividere tilbakelangt distanse på tid. Deklarér en metode som tar to **double**-parametre `distanse` og `tid`, og som returnerer hastigheten som en **double**-verdi. Hvis verdien til `tid` er `0`, skal metoden returnere `-1`. Du skal skrive *hele* metoden, inkludert signaturen. Finn på et passende metodenavn.

```
double snittHastighet(double distanse, double tid) {
    if (tid == 0) {
        return -1;
    } else {
        return distanse / tid;
    }
}
```

5d

I sjø- og luftfart oppgis ofte hastigheter i *knop*. Én knop er definert som én nautisk mil pr. time, som tilsvarer 1852 meter pr. time. Deklarér en metode som tar en **double**-parameter `knop` og returnerer den tilsvarende hastigheten i *km/t* som en **double**-verdi. Du skal skrive *hele* metoden, inkludert signaturen. Finn på et passende metodenavn.

```
double beregnKmT(double knop) {
    return knop * 1.852;
}
```

(Fortsettes på side 8.)

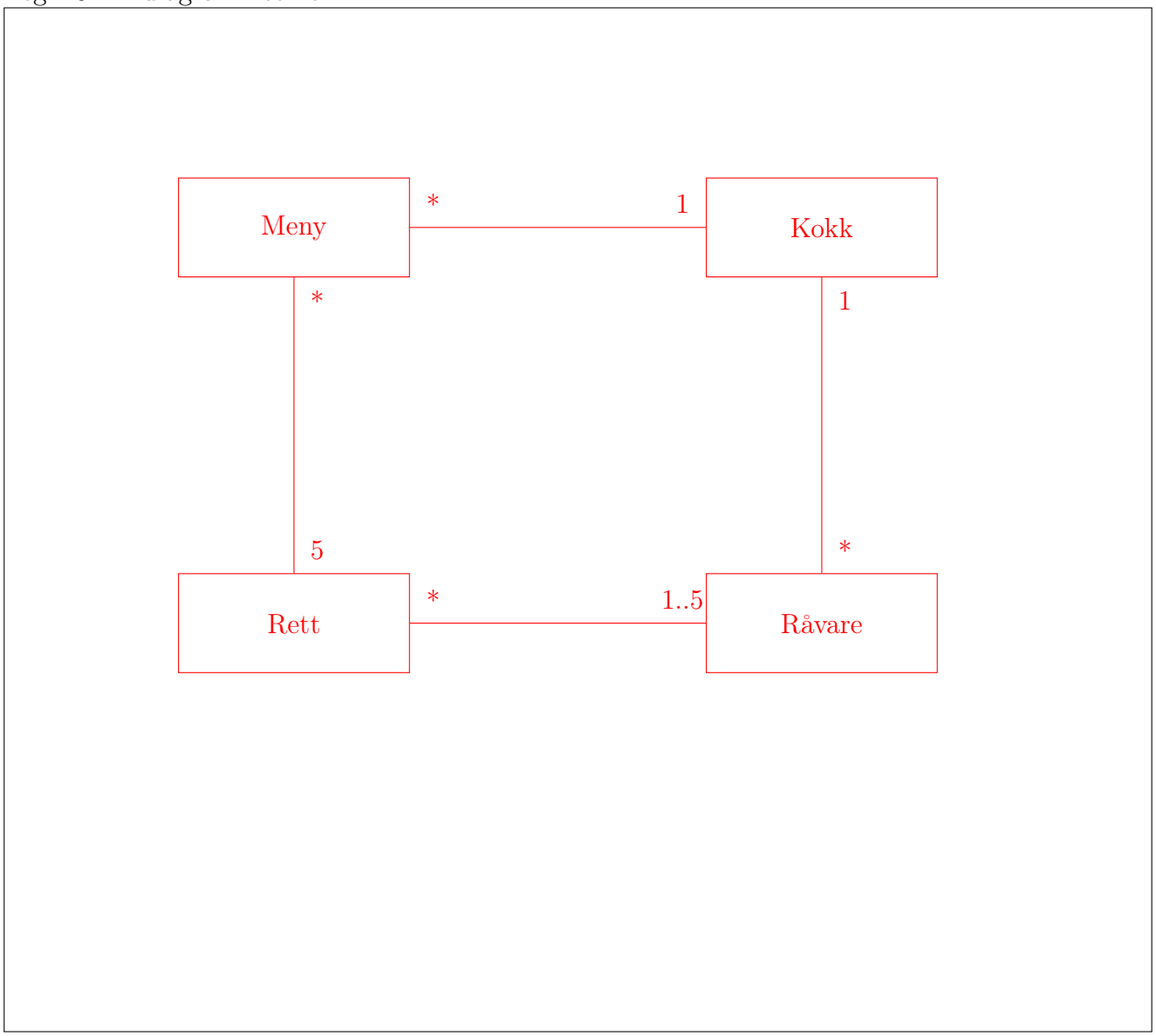
Oppgave 6 UML (10 poeng)

Professor Flink er stamkunde på restauranten "Den Sultne Akademiker". En dag blir Prof. Flink spurt av hovmesteren om han ikke kan hjelpe restauranten med å lage et Java-program for å holde styr på *menyer*, *retter*, *råvarer* og *kokker*. Siden Prof. Flink er en nybegynner i Java, ber han deg om hjelp.

En meny har nøyaktig fem retter, og en rett kan forekomme i flere menyer. Hver rett kan inneholde maksimalt fem råvarer. Hver meny har én ansvarlig kokk, og hver råvare har én kokk som er ansvarlig for innkjøp av råvaren. En Kokk kan være ansvarlig for flere menyer og råvarer.

Tegn et UML klassediagram for Java-programmet. Gi navn til klassene, tegn forhold mellom dem og sett på riktig antall på hver side av forholdene. Du behøver *ikke* skrive inn variable og metoder i klassene i diagrammet.

Tegn UML-diagrammet her:



(Fortsettes på side 9.)

Oppgave 7 HashMap, objekter og metoder (25 poeng)

Professor Flink har begynt å skrive Java-programmet for restaurant "Den Sultne Akademiker". Han har deklarert en klasse `Kokk`, gjengitt nedenfor.

```
public class Kokk {

    private String navn;         // kokkens navn
    private int tlfnr;           // kokkens tlfnr
    private int ansettelsesAar; // det aaret kokken ble ansatt

    public Kokk(String navn, int tlfnr, int aar) {
        this.navn = navn;
        this.tlfnr = tlfnr;
        this.ansettelsesAar = aar;
    }

    public String toString() {
        return "Kokk: " + navn + ", tlf " + tlfnr +
            ", ansatt " + ansettelsesAar;
    }

    public String getNavn() {
        return navn;
    }

    public int getTlfnr() {
        return tlfnr;
    }

    public int getAnsattAar() {
        return ansettelsesAar;
    }

    // Returnerer antall aar kokken har vaert ansatt
    public int ansienitet() {
        Calendar cal = Calendar.getInstance();
        return cal.get(Calendar.YEAR) - ansettelsesAar;
    }

}
```

Du behøver ikke forstå hvordan metoden `ansienitet()` fungerer. Det er tilstrekkelig at du vet at den returnerer det antall år kokken har vært ansatt. Du kan fritt bruke metoden i svarene dine.

(Fortsettes på side 10.)

7a HashMap og løkker (5 av 25 poeng)

Anta at Prof. Flink har deklarert

```
HashMap<String, Kokk> kokker = new HashMap<String, Kokk>();
```

og lagt inn noen `Kokk`-objekter i den. For hver av løkkene under, kryss av for om løkka enten

(a) ikke kompilerer, eller

(b) kompilerer og skriver ut *alle* `Kokk`-objektene i `kokker`, eller

(c) kompilerer, men skriver *ikke* ut alle objektene i `kokker`.

(a)	(b)	(c)	
☒	☐	☐	`for (Kokk kokk : kokker) { System.out.println(kokk.toString()); }`
☒	☐	☐	`for (Kokk kokk : kokker.keySet()) { String txt = kokk.toString(); System.out.print(txt + "\n"); }`
☐	☒	☐	`for (Kokk kokk : kokker.values()) { System.out.println(kokk.toString()); }`
☐	☐	☒	`Iterator<Kokk> iterator = kokker.values().iterator(); String tmp; while (iterator.hasNext()) { tmp = iterator.next().toString(); System.out.println(tmp); if (iterator.hasNext()) { iterator.next(); } }`
☐	☒	☐	`Iterator<String> keyIterator = kokker.keySet().iterator(); Kokk kokk; while (keyIterator.hasNext()) { kokk = kokker.get(keyIterator.next()); System.out.println(kokk.toString()); }`

(Fortsettes på side 11.)

7b Objektvariable (2 av 25 poeng)

Følgende kode gir kompileringsfeil. Hvilke endringer kan gjøres for å få koden til å kompilere?

```
Kokk kokk = new Kokk("Eivind", 12345678, 1982);
kokk.ansettelsesAar = 1983;
```

Skriv svaret ditt her:

Variabelen ansettelsesAar er deklarert som private i klassen Kokk, så linje 2 gir kompileringsfeil. Man kan enten endre variabelen til public, eller lage et set-metode for den.

7c Objektmetoder og HashMap (18 av 25 poeng)

Professor Flink deklarerer klassen `Restaurant` som vist nedenfor. Klassen inneholder foreløpig bare deklarasjon av `HashMap`-variabelen `kokker`, så Prof. Flink ønsker din hjelp til å deklarere følgende metoder i klassen. Du kan anta at kokkenes navn skal brukes som nøkkel i `HashMap`'en `kokker`.

- **public boolean** `leggTilKokk(Kokk kokk)`
  Metoden tar et `Kokk`-objekt som parameter. Hvis det finnes et `Kokk`-objekt i `kokker` med samme navn som `kokk`, skal metoden returnere **false** *uten* å legge `kokk` inn i `kokker`. Hvis det derimot *ikke* finnes noe `Kokk`-objekt i `kokker` med samme navn som `kokk`, skal `kokk` legges inn i `kokker` og **true** returneres.

- **public boolean** `fjernKokk(String navn)`
  Metoden skal fjerne kokken med navn `navn` fra `kokker`. Metoden skal returnere **true** dersom et `Kokk`-objekt ble fjernet, og **false** dersom ingen kokk med navnet `navn` ble funnet.

- **public** `Kokk finnKokk(String navn)`
  Metoden skal sjekke om det finnes et `Kokk`-objekt med navn `navn` i `kokker`. Hvis et slikt objekt finnes, skal objektet returneres. Hvis det *ikke* finnes noe slikt objekt i `kokker`, skal **null** returneres.

(Fortsettes på side 12.)

- **public int** `antallKokker()`
  Metoden skal returnere antall `Kokk`-objekter i `kokker`.

- **public** `Kokk[] lengstAnsienitet()`
  Metoden skal finne den eller de kokken(e) som har lengst ansienitet (dvs. har vært ansatt lengst) regnet i antall år. De aktuelle `Kokk`-objektene skal returneres som en `Kokk`-array. Dersom `kokker` er tom, skal metoden returnere en tom `Kokk`-array. Dersom én kokk er alene om å ha lengst ansienitet, skal den returnerte `Kokk`-arrayen ha lengde 1 og inneholde det respektive `Kokk`-objektet på plass 0. Hvis det derimot er $n > 1$ kokker som deler på å ha lengst ansienitet, skal den returnerte `Kokk`-arrayen ha lengde n og inneholde *alle* de respektive `Kokk`-objektene på hver sin plass. Du kan benytte metoden `ansienitet()` i klassen `Kokk` til å finne hvor mange år en kokk har vært ansatt.

Metodene skal deklareres inne i klassen `Restaurant` under.

```
public class Restaurant {

    HashMap<String,Kokk> kokker = new HashMap<String,Kokk>();

    // Deklarer metodene dine her:

    public boolean leggTilKokk(Kokk kokk) {
        Kokk funnetKokk = kokker.get(kokk.getNavn());
        if (funnetKokk != null) {
            return false;
        } else {
            kokker.put(kokk.getNavn(), kokk);
            return true;
        }
    }

    public boolean fjernKokk(String navn) {
        Kokk fjernetKokk = kokker.remove(navn);
        if (fjernetKokk != null) {
            return true;
        } else {
            return false;
        }
    }

    public Kokk finnKokk(String navn) {
        return kokker.get(navn);
    }
```

(Fortsettes på side 13.)

```
public int antallKokker() {
    return kokker.size();
}

public Kokk[] lengstAnsienitet() {
    HashMap<String,Kokk> lengst = new HashMap<String, Kokk>();
    int lengstAnsienitet = -1;
    for (Kokk kokk : kokker.values()) {
        if (kokk.ansienitet() > lengstAnsienitet) {
            lengst.clear();
            lengst.put(kokk.getNavn(), kokk);
            lengstAnsienitet = kokk.ansienitet();
        } else if (kokk.ansienitet() == lengstAnsienitet) {
            lengst.put(kokk.getNavn(), kokk);
        }
    }
    Kokk[] resultat = new Kokk[lengst.size()];
    int index = 0;
    for (Kokk kokk : lengst.values()) {
        resultat[index++] = kokk;
    }
    return resultat;
}
```

(Fortsettes på side 14.)

```
}
```

(Fortsettes på side 15.)

Oppgave 8 Klasser og objekter (25 poeng)

Professor Flink skal ferdigstille programmet for restaurant "Den Sultne Akademiker". Han skriver raskt klassen `Raavare` gjengitt nedenfor.

```
public class Raavare {

    private String navn;
    private Kokk ansvarligKokk;

    public Raavare(String navn, Kokk kokk) {
        this.navn = navn;
        this.ansvarligKokk = kokk;
    }

    public String getNavn() {
        return navn;
    }

    public Kokk getAnsvarligKokk() {
        return ansvarligKokk;
    }
}
```

8a Datastruktur for råvarer i klassen `Rett` (10 av 25 poeng)

Det framgår av programspesifikasjonen i oppgave 6 at en rett kan ha maksimalt fem råvarer. Dette skal gjenspeiles i programmet ved at klassen `Rett` har en `Raavare`-array med pekere til de `Raavare`-objektene retten inneholder. Merk at en rett godt kan ha *mindre enn* fem råvarer. Dette må vi ta høyde for. Skriv ferdig klassen `Rett` nedenfor ved å legge til følgende. Det er satt av tilstrekkelig plass i klassen til å skrive svarene dine.

1. Deklarér `Raavare`-array med passende antall plasser og evt. andre variable du trenger for å besvare oppgaven.

2. Skriv ferdig metoden **public boolean** `leggTilRaavare(Raavare raavare)`. Hvis en rett allerede inneholder fem råvarer, skal metoden returnere **false**. Dersom retten inneholder færre enn fem råvarer, skal metoden legge til `raavare` i neste ledige plass i arrayen du deklarerte i punkt 1 og returnere **true**.

3. Skriv ferdig metoden **public boolean** `inneholderRaavare(String navn)`. Metoden skal lete gjennom alle råvarene retten inneholder. Dersom den finner en råvare med et navn som er *tekstlig lik* `navn`, skal metoden returnere **true**. Dersom en slik råvare ikke finnes, skal metoden returnere **false**.

(Fortsettes på side 16.)

```
public class Rett {

    private String navn;

    public Rett(String navn) {
        this.navn = navn;
    }

    public String getNavn() {
        return navn;
    }

    // Deklarer Raavare-arrayen og evt. andre variable her:


    Raavare[] raavarer = new Raavare[5];
    int antallRaavarer = 0;




    public boolean leggTilRaavare(Raavare raavare) {




        if (antallRaavarer < raavarer.length) {
            raavarer[antallRaavarer] = raavare;
            antallRaavarer++;
            return true;
        } else {
            return false;
        }









    }
```

(Fortsettes på side 17.)

```
    public boolean inneholderRaavare(String navn) {




        for (int i=0; i<antallRaavarer; i++) {
            if (raavarer[i].getNavn().equals(navn)) {
                return true;
            }
        }
        return false;









    }
}
```

8b Hvor hyppig brukes råvarene? (15 av 25 poeng)

Professor Flink har ved din hjelp nesten skrevet ferdig Java-programmet. Han har lagt til følgende deklarasjoner i klassen `Restaurant`:

```
    HashMap<String, Raavare> raavarer =
        new HashMap<String, Raavare>();
    HashMap<String, Rett> retter = new HashMap<String, Rett>();
```

Professoren har lagt inn testdata i systemet. Som en ekstra bonus til restauranten, ønsker han å legge til en metode i `Restaurent`-klassen som finner ut hvor ofte hver enkelt råvare blir brukt i rettene. Skriv en metode som for hver råvare teller opp i hvor mange retter råvaren forekommer. Metoden skal skrive ut en liste på skjermen der hver linje inneholder navnet på en råvare og i hvor mange retter råvaren blir brukt. Du kan anta at hver råvare har et unikt navn.

Skriv metoden på neste side.

(Fortsettes på side 18.)

```
public void raavareHyppighet() {
    int antallRetter;
    for (Raavare raavare : raavarer.values()) {
        antallRetter = 0;
        for (Rett rett : retter.values()) {
            if (rett.inneholderRaavare(raavare.getNavn())) {
                antallRetter++;
            }
        }
        System.out.print(raavare.getNavn());
        System.out.print(" forekommer i ");
        System.out.println(antallRetter + " retter");
    }
}
```

(Fortsettes på side 19.)

Oppgave 9 Personvernloven (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 Personvernloven 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.

Bryter §8 om samtykke og saklig begrunnet innsamling av data - ingen avtale med gjestene, heller ikke nødvendig for restaurantens drift. Ivaretar neppe den registrertes interesse.

Bryter klart med §11 pkt. a (pga. brudd §8) og pkt. c - data brukes til et klart annet formål (reklame) enn det var samlet inn for (betalig av måltider) uten samtykke.

Det står heller ikke at restaurantgjestene har gitt sitt samtykke til denne type offentliggjøring.

Den bryter også med §11 pkt b - dette er skattedata som samles inn som ikke er saklig at en restaurant samler inn (selv om skattedata er offentlig).

§ 8 Behandling av personopplysninger skal bare gjøres etter samtykke og være saklig begrunnet, må være nødvendig for:
a) Oppfylle avtale med den registrerte
c) Ivareta den registrertes interesser
e) å utøve offentlig myndighet
f) å ivareta en berettiget interesse som overstiger den registrertes interesse

§ 11. Grunnkrav til behandling av personopplysninger
Den behandlingsansvarlige skal sørge for at personopplysningene som behandles
a) bare behandles når dette er tillatt etter § 8 og § 9,
b) bare nyttes til uttrykkelig angitte formål som er saklig begrunnet i den behandlingsansvarliges virksomhet,
c) ikke brukes senere til formål som er uforenlig med det opprinnelige formålet med innsamlingen, uten at den registrerte samtykker,
d) er tilstrekkelige og relevante for formålet med behandlingen, og
e) er korrekte og oppdatert, og ikke lagres lenger enn det som nødvendig ut fra formålet med behandlingen, jf. § 27 og § 28.

(Fortsettes på side 20.)