Høgskulen på Vestlandet

EKSAMENSOPPGAVE/EKSAMENSOPPGÅVE

Emnekode: DAT100

Emnenavn/Emnenamn: Grunnleggende Programmering

Utdanning/kull/klasse:
Dataingeniør + Informasjonsteknologi / H2018 /
1Data og 1Informajonsteknologi

Dato: 5. juni 2019

Eksamensform: Skriftlig

Eksamenstid: 4 klokketimer

Antall eksamensoppgaver/ Tal på eksamensoppgåver: 5

Antall vedlegg/ Tal på vedlegg: 1 side

Tillatte hjelpemidler/ Tekniske hjelpemiddel: Ingen

Fagansvarlig/ Fagansvarleg:
Sven-Olai Høyland (472 59 543), Lars Michael Kristensen (938 66 491)

Språk: Bokmål (nynorsk finnes i andre del av oppgavesettet)

Merknader/ Merknad: Ingen

Oppgave 1 (Vekt 10%)

I hver deloppgave har du gitt et Java-program som kan kompileres og kjøres. Du skal svare hva som blir skrevet ut på skjermen når metoden main blir kjørt. Merk at det kan være skrivesetninger i både main og metodene som blir kalt.

a)

```
public static void main(String[] args) {
   System.out.println(5 + 3 * 2);
   System.out.println(9 % 5);
   System.out.println(9 / 5);

   int i = 3;
   int j = 7;
   System.out.println( (i < 3) && (j >= 5) );
   System.out.println( (j != 3) || (i < j) );
}
```

b)

```
public static int b(int x, int y) {

   while (x != y) {
      System.out.println("x = " + x + ", y = " + y);
      if (x > y) {
         x = x - y;
      } else {
         y = y - x;
      }
   }

   return x;
}

public static void main(String[] args) {
   System.out.println(b(28, 12));
}
```

c)

```
public static int c(int x) {
   int i = 0;

   do {
      x = x / 10;
      i++;
      System.out.println("x = " + x);
   } while (x != 0);

   return i;
}

public static void main(String[] args) {
   System.out.println(c(32812));
}
```

d) Merk at det er to metoder med navnet **f**.

```
public static double f(int a, double b){
   return a - b;
}

public static double f(double a, int b){
   return a + b;
}

public static void main(String[] args) {
   System.out.println(f(4.0, 7));
}
```

e)

```
public static void e() {
   String[] sTab = new String[4];
   sTab[0] = "a";
   sTab[1] = "ab";
   sTab[2] = "abc";

   int totalLengde = 0;
   try {
      for (int i = 0; i < sTab.length; i++) {
         totalLengde += sTab[i].length();
         System.out.println("Lengde så langt: " + totalLengde);
      }
      System.out.println("Total lengde av strengene: " + totalLengde);
   } catch (ArithmeticException e) {
      System.out.println("Unntak nr 1 kasta.");
   } catch (NullPointerException e) {
      System.out.println("Unntak nr 2 kasta.");
   } catch (Exception e) {
      System.out.println("Unntak nr 3 kasta.");
   }
}

public static void main(String[] args) {
   e();
}
```

Oppgave 2 (Vekt 25%)

I denne oppgaven skal vi skrive JAVA kode for klasser til et blogg-system på nettet. Dersom parametertypen eller returtypen ikke er gitt, må du bestemme typene selv.

a) Definer en klasse `Innlegg` med objektvariabler `bruker` (streng), `dato` (streng), og antall `likes` (heltall). Dei tre objektvariablene skal ikke være synlige utenfor klassen og klassen skal ha en konstruktør `Innlegg(String bruker, String dato)` som setter bruker og dato for de to objektvariablene og setter antall `likes` til 0.

b) Definer en metode `skrivUt()` som skriver ut verdier for de tre objektvariablene `bruker`, `dato` og antall `likes` på skjermen.

c) Definer to subklasser for klassen `Innlegg` med navn `Bilde` og `Tekst`. `Bilde` skal ha en objektvariabel `url` (String) som gir en URL til der vi finner bildet. `Tekst` skal ha en objektvariabel `tekst` som er teksten i et blogg-innlegg.

d) Implementer konstruktørar i de to klassene `Bilde` og `Tekst` som initialiserer alle objektvariablene (også de som er arvet fra superklassen) ut fra verdier som er gitt som argumenter til konstruktøren.

e) Implementer en metode `skrivUt()` i hver av subklassene `Bilde` og `Tekst` som skriver verdien av alle objektvariablene ut på skjermen (inkludert objekvariablene som er arvet fra superklassen).

f) Skriv en main-metode som tillater brukeren å opprette et blogg-innlegg (bilde eller tekst) dvs. oppretter et `Bilde` eller et `Tekst`-objekt ut fra input fra brukeren og som bruker `skrivUt`-metoden til slutt for å skrive informasjonen ut på skjermen.

Oppgave 3 (Vekt 35%)

I denne oppgaven skal du først lage noen klasser som inngår i et system for å administrere en samling instruksjonsvideoer. Klassene vi skal se på er: `Video` og `Videoarkiv`. Klassene er nærmere beskrevet i delspørsmålene nedenfor.

a) Lag klassen `Video`. Klassen skal inneholde:

- Objektvariabler som skal være private.
  - `tittel` – Tittel på videoen (tekststreng).
  - `produsent` – Navn på den som har produsert videoen.
  - `sekund` – Lengden på videoen i hele sekunder.
- Konstruktør med parametere der vi kan gi verdi til alle objektvariablene.
- `get`- og `set`-metoder for alle objektvariablene.
- `toString()` metode som returnerer en streng på formen

  "Tittel: **Løkker i Java**, Produsent: **Ole Olsen**, Tid: **5.02**".

  Legg merke til at tiden skal være i minutter og sekunder og at sekundene skal vises med to siffer.

b) Klassen `Videoarkiv` inneholder en samling videoer. Klassen skal inneholde:

- Objektvariabler som skal være private.
  - `vTab` – tabell som inneholder videoer som ligger lagret fortløpende i starten av tabellen.
  - `antal` – antall videoer som faktisk er i tabellen.

- Konstruktør
  `public Videoarkiv(int maksAntal)`
  Konstruktøren skal opprette en tabell med plass til `maksAntal` videoer.

Lag følgende public-metoder:

c) `void leggTil(Video v)`
Objektmetode som legger til en video i neste ledige posisjon i tabellen dersom det er plass. Dersom tabellen er full, skal det skrives en feilmelding.

d) `int finnTotalTid()`
Objektmetode som returnerer samlet tid (i sekunder) for alle videoene i arkivet.

e) `void listAlle(String produsent)`
Objektmetode som lister alle videoer av en produsent (gitt som parameter). Først skal metoden skrive en passende overskrift, deretter kommer en linje med bare tittel for hver video av produsenten. Til slutt skal antall videoer knyttet til produsenten skrives ut.

f) `void slett(String tittel)`
Objektmetode som sletter en video med gitt tittel (kan anta det finnes høyst en). Dersom tittelen ikke finnes skal det skrives en passende feilmelding. Dersom tittelen finnes skal tilhørende video slettes og etter sletting skal fremdeles alle videoene ligge etter hverandre i starten av tabellen (skal ikke være "hull").

g) Lag en main-metode du bruker klassene `Video` og `Videoarkiv`. Du skal opprette et arkiv med plass til 100 videoer. Legg til 3 videoer. List alle videoer produsert av "Ole Olsen". Skriv ut beste par som passer innenfor en tidsgrense på 10 minutter (fra punkt h) nedenfor) . Slett videoen med tittel "Løkker i Java".

h) Lag metoden

`void bestePar(int tidsgrense)`

Objektmetode som finner og skriver ut de **to** videoene som passer best innenfor en tidsgrense. Det betyr at summen av lengdene på de to videoene skal være så stor som mulig, men ikke over tidsgrensen. Metoden skal også gi en fornuftig utskrift om det ikke finnes et par med samlet tid innenfor grensen.

Eksempel (har brukt små tall så det skal være lettere å følge med)
Tider: 14, 5, 13, 9, 8
Tidsgrense: 16

Nå er det videoene med tid 5 og 9 (totaltid 14) som passer best selv om de med 9 og 8 (samlet tid 17) er nærmere tidsgrensen.

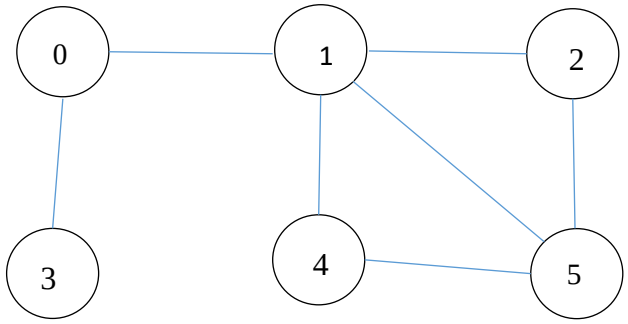
Utskrift: <tittelNr1> og <tittelNr2> passer best.

Oppgave 4 (Vekt 20%)

En graf er en mengde noder og en mengde kanter. To noder er naboer om det er en kant mellom dem.

Vi kan representere grafer på ulike måter. En måte er å bruke en nabomatrise av sannhetsverdier. Da bruker vi en n x n matrise der n er tallet på noder i grafen. Dersom det er en kant mellom to noder u og w, så vil vi ha true på rekke u og kolonne w og true på rekke w og kolonne u i matrisen.

I figuren nedenfor under ser vi en graf og tilhørende nabomatrise (T = true, blank = false). Første rekke og første kolonne er «overskrifter» slik at tabellen skal bli lettere å lese.



	0	1	2	3	4	5
0		T		T		
1	T		T		T	T
2		T				T
3	T					
4		T				T
5		T	T		T	

I oppgaven vil vi kan anta at nodene er nummerert fra 0 til n – 1 slik at vi enkelt kan representere en nabomatrise som en to-dimensjonal tabell.

Nedenfor er det gitt deklarasjon av en to-dimensjonal tabell og den første linjen av metodene som skal lages. Vi har også vist hvordan matrisen i eksemplet blir representert i Java.

```
public class Graf {
      // matrisen i eksemplet over
      private boolean[][] nabomatrise = {
                  { false, true, false, true, false, false },
                  { true, false, true, false, true, true },
                  { false, true, false, false, false, true },
                  { true, false, false, false, false, false },
                  { false, true, false, false, false, true },
                  { false, true, true, false, true, false } };
      …
      public boolean erNaboer(int u, int w) {…}

      public int grad(int v) {…}

      public int antallLokker() {…}

      public boolean uavhengigMengde (int[] s) {…}

}
```

a) Lag metoden `erNaboer(int u, int w)` som returnerer true om node u og node w er naboer.

b) Lag metoden `grad(int v)` for å finne gradtallet for en node gitt som parameter. Gradtallet er antall naboer til noden.
Eksempel: Node 0 har 2 naboer (nodene 1 og 3).

c) En løkke i en graf er en kant fra en node til seg selv. Om node nr v har en løkke, så vil element (v, v) i matrisen være `true` . Lag metoden `antallLokker()` som returnerer antall løkker i grafen.

d) En uavhengig mengde i en graf er en mengde noder der ingen av nodene i mengden er nabo med en annen node i mengden.
Eksempel: Mengden {0, 2, 4} er en uavhengig mengde i grafen ovenfor.

Lag metoden `uavhengigMengde(int[] s)` som avgjør om mengden av noder gitt som parameter er en uavhengig mengde i grafen.

Oppgave 5 (Vekt 10%)

Vi har en liste av ord der hvert ord kan forekomme flere ganger. Vi ønsker å lage en frekvensoversikt (antall ganger hvert ord forekommer).
Eksempel: Dersom listen av ord er:

```
{"er", "det", "alle", "er", "det", "det"}
```

så blir frekvensoversikten:

```
er: 2
det: 3
alle: 1
```

Vi skal bruke de forhåndsdefinerte klassene `ArrayList` og `HashMap` i Java for å løse oppgaven. Du har gitt starten av programmet (du trenger ikke gjenta dette i svaret). Du finner deler av API-dokumentasjonen for `HashMap` som vedlegg.

Starten av programmet:

```
public static void main(String[] args) {
      ArrayList<String> listeAvOrd = new ArrayList<String>();
      HashMap<String, Integer> fo = new HashMap<String, Integer>();

      //b Lag frevensoversikten

      //a Leser inn et ord fra bruker og skriver ut tilhørende
      // frekvens
}
```

a) Anta at du har en HashMap som inneholder ord med tilhørende frekvens (skal gjøres i neste delspørsmål). Les inn et ord fra brukeren og skriv ut hvor mange ganger det finnes (svaret kan være 0).

b) Gå gjennom array-listen `listeAvOrd` ved bruk av en utvidet for-løkke og lag frekvensoversikten. Første gang du finner ordet, setter du ordet inn i HashMap'en med frekvens 1. Om ordet finnes fra før, øker du frekvensen med en.

Lykke til!

Vedlegg: Deler av API-dokumentasjon for Class HashMap<K,V>

void	**clear**() Removes all of the mappings from this map.
boolean	**containsKey**(**Object** key) Returns true if this map contains a mapping for the specified key.
boolean	**containsValue**(**Object** value) Returns true if this map maps one or more keys to the specified value.
V	**get**(**Object** key) Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.
V	**getOrDefault**(**Object** key, **V** defaultValue) Returns the value to which the specified key is mapped, or defaultValue if this map contains no mapping for the key.
boolean	**isEmpty**() Returns true if this map contains no key-value mappings.
Set<**K**>	**keySet**() Returns a **Set** view of the keys contained in this map.
V	**put**(**K** key, **V** value) Associates the specified value with the specified key in this map.
void	**putAll**(**Map**<? extends **K**,? extends **V**> m) Copies all of the mappings from the specified map to this map.
V	**remove**(**Object** key) Removes the mapping for the specified key from this map if present.
boolean	**remove**(**Object** key, **Object** value) Removes the entry for the specified key only if it is currently mapped to the specified value.
V	**replace**(**K** key, **V** value) Replaces the entry for the specified key only if it is currently mapped to some value.
boolean	**replace**(**K** key, **V** oldValue, **V** newValue) Replaces the entry for the specified key only if currently mapped to the specified value.
int	**size**() Returns the number of key-value mappings in this map.

Høgskulen på Vestlandet

EKSAMENSOPPGAVE/EKSAMENSOPPGÅVE

Emnekode: DAT100

Emnenavn/Emnenamn: Grunnleggende Programmering

Utdanning/kull/klasse:
Dataingeniør + Informasjonsteknologi / H2018 /
1Data og 1Informajonsteknologi

Dato: 5. juni 2019

Eksamensform: Skriftlig

Eksamenstid: 4 klokketimer

Antall eksamensoppgaver/ Tal på eksamensoppgåver: 5

Antall vedlegg/ Tal på vedlegg: 1 side

Tillatte hjelpemidler/ Tekniske hjelpemiddel: Ingen

Fagansvarlig/ Fagansvarleg:
Sven-Olai Høyland (472 59 543), Lars Michael Kristensen (938 66 491)

Språk: Nynorsk (bokmål finn du i første del av oppgåvesettet)

Merknader/ Merknad: Ingen

Oppgåve 1 (Vekt 10%)

I kvar deloppgåve har du gitt eit Java-program som kan kompilerast og køyrast. Du skal svare kva som blir skrive ut på skjermen når metoden main blir køyrt. Merk at det kan vere skrivesetningar i både main og metodene som blir kalla.

a)

```
public static void main(String[] args) {
   System.out.println(5 + 3 * 2);
   System.out.println(9 % 5);
   System.out.println(9 / 5);

   int i = 3;
   int j = 7;
   System.out.println( (i < 3) && (j >= 5) );
   System.out.println( (j != 3) || (i < j) );
}
```

b)

```
public static int b(int x, int y) {

   while (x != y) {
      System.out.println("x = " + x + ", y = " + y);
      if (x > y) {
         x = x - y;
      } else {
         y = y - x;
      }
   }

   return x;
}

public static void main(String[] args) {
   System.out.println(b(28, 12));
}
```

c)

```
public static int c(int x) {
   int i = 0;

   do {
      x = x / 10;
      i++;
      System.out.println("x = " + x);
   } while (x != 0);

   return i;
}

public static void main(String[] args) {
   System.out.println(c(32812));
}
```

d) Merk at det er to metodar med namnet **f**.

```
public static double f(int a, double b){
   return a - b;
}

public static double f(double a, int b){
   return a + b;
}

public static void main(String[] args) {
   System.out.println(f(4.0, 7));
}
```

e)

```
public static void e() {
   String[] sTab = new String[4];
   sTab[0] = "a";
   sTab[1] = "ab";
   sTab[2] = "abc";

   int totalLengde = 0;
   try {
      for (int i = 0; i < sTab.length; i++) {
         totalLengde += sTab[i].length();
         System.out.println("Lengde så langt: " + totalLengde);
      }
      System.out.println("Total lengde av strengene: " + totalLengde);
   } catch (ArithmeticException e) {
      System.out.println("Unntak nr 1 kasta.");
   } catch (NullPointerException e) {
      System.out.println("Unntak nr 2 kasta.");
   } catch (Exception e) {
      System.out.println("Unntak nr 3 kasta.");
   }
}

public static void main(String[] args) {
   e();
}
```

Oppgåve 2 (Vekt 25%)

I denne oppgåva skal vi skrive JAVA kode for klassar til eit blogg-system på nettet. Dersom parametertypen eller returtypen ikkje er gitt, må du bestemme typane sjølv.

a) Definer ein klasse `Innlegg` med objektvariablar `bruker` (streng), `dato` (streng), og antall `likes` (heiltal). Dei tre objektvariablane skal ikkje vere synlege utanfor klassen og klassen skal ha ein konstruktør `Innlegg(String bruker, String dato)` som set bruker og dato for dei to objektvariablane og set antal `likes` til 0.

b) Definer ein metode `skrivUt()` som skriv ut verdiar for dei tre objektvariablane `bruker`, `dato` og antal `likes` på skjermen.

c) Definer to subklassar for klassen `Innlegg` med namna `Bilde` og `Tekst`. `Bilde` skal ha ein objektvariabel `url` (String) som gir ein URL til der vi finn bildet. `Tekst` skal ha ein objektvariabel `tekst` som er teksten i eit blogg-innlegg.

d) Implementer konstruktørar i dei to klassane `Bilde` og `Tekst` som initialiserer alle objektvariablane (også dei som er arva frå superklassen) ut frå verdiar som er gitt som argument til konstruktøren.

e) Implementer ein metode `skrivUt()` i kvar av subklassane `Bilde` og `Tekst` som skriv verdien av alle objektvariablane ut på skjermen (inkludert objekvariablane som er arva frå superklassen).

f) Skriv ein main-metode som tillet brukaren å opprette eit blogg-innlegg (bilde eller tekst) dvs. oppretter eit `Bilde` eller eit `Tekst`-objekt ut frå input frå brukeren og som bruker `skrivUt`-metoden til slutt for å skrive informasjonen ut på skjermen.

Oppgåve 3 (Vekt 35%)

I denne oppgåva skal du først lage nokre klassar som inngår i eit system for å administrere ei samling instruksjonsvideoar. Klassane vi skal sjå på er: `Video` og `Videoarkiv`. Klassane er nærare beskrive i delspørsmåla nedanfor.

a) Lag klassen `Video`. Klassen skal innehalde:

- Objektvariablar som skal være private.
  - `tittel` – Tittel på videoen (tekststreng).
  - `produsent` – Namn på den som har produsert videoen.
  - `sekund` – Lengda på videoen i heile sekund.
- Konstruktør med parametrar der vi kan gi verdi til alle objektvariablane.
- `get`- og `set`-metodar for alle objektvariablane.
- `toString()` metode som returnerer ein streng på forma

  "Tittel: **Løkker i Java**, Produsent: **Ole Olsen**, Tid: **5.02**".

  Legg merke til at tida skal vere i minutt og sekund og at sekunda skal visast med to siffer.

b) Klassen `Videoarkiv` inneheld ei samling videoar. Klassen skal innehalde:

- Objektvariablar som skal vere private.
  - `vTab` – tabell som inneheld videoar.
  - `antal` – talet på videoar som faktisk er i tabellen som ligg lagra fortløpande i starten av tabellen.

- Konstruktør
  `public Videoarkiv(int maksAntal)`
  Konstruktøren skal opprette ein tabell med plass til `maksAntal` videoar.

Lag følgjande public-metodar:

c) `void leggTil(Video v)`
Objektmetode som legg til ein video i neste ledige posisjon i tabellen dersom det er plass. Dersom tabellen er full, skal det skrivast ei feilmelding.

d) `int finnTotalTid()`
Objektmetode som returnerer samla tid (i sekund) for alle videoane i arkivet.

e) `void listAlle(String produsent)`
Objektmetode som listar alle videoar av ein produsent (gitt som parameter). Først skal metoden skrive ei passande overskrift, deretter ei linje med berre tittel for kvar video av produsenten. Til slutt skal talet på videoar knytt til produsenten skrivast ut.

f) `void slett(String tittel)`
Objektmetode som slettar ein video med gitt tittel (kan anta det finst høgst ein). Dersom tittelen ikkje finst skal det skrivast ei passande feilmelding. Dersom tittelen finst skal tilhøyrande video slettast og etter sletting skal framleis alle videoane ligge etter kvarandre i starten av tabellen (skal ikkje vere "hol").

g) Lag ein main-metode der du brukar klassane `Video` og `Videoarkiv`. Du skal opprette eit arkiv med plass til 100 videoar. Legg til 3 videoar. List alle videoar produsert av "Ole Olsen". Skriv ut beste par som passar innanfor ei tidsgrense på 10 minutt (frå punkt h) nedanfor). Slett videoen med tittel "Løkker i Java".

h) Lag metoden

`void bestePar(int tidsgrense)`

Objektmetode som finn og skriv ut dei **to** videoane som passer best innanfor ei tidsgrense. Det betyr at summen av lengdene på dei to videoane skal vere så stor som mulig, men ikkje over tidsgrensa. Metoden skal også gi ei fornuftig utskrift om det ikkje finst eit par med samla tid innanfor grensa.

Eksempel (har brukt små tall så det skal være lettare å følge med)
Tider: 14, 5, 13, 9, 8
Tidsgrense: 16

No er det videoane med tid 5 og 9 (totaltid 14) som passar best sjølv om dei med 9 og 8 (samla tid 17) er nærare tidsgrensa.

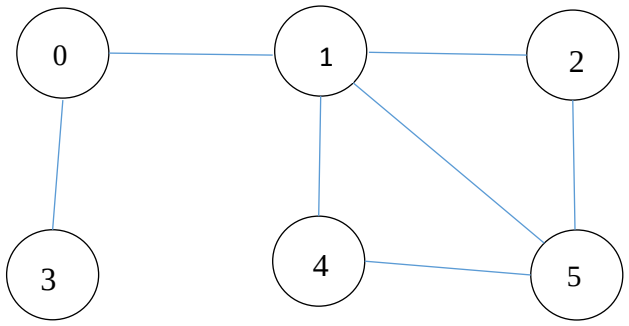
Utskrift: <tittelNr1> og <tittelNr2> passer best.

Oppgåve 4 (Vekt 20%)

Ein graf er ei mengde nodar og ei mengde kantar. To nodar er naboar om det er ein kant mellom dei.

Vi kan representere grafar på ulike måtar. Ein måte er å bruke ei nabomatrise av sannheitsverdiar. Då brukar vi ei n x n matrise der n er talet på nodar i grafen. Dersom det er ein kant mellom to nodar u og w, så vil vi ha true på rekke u og kolonne w og true på rekke w og kolonne u i matrisa.

I figuren nedanfor under ser vi ein graf og tilhøyrande nabomatrise (T = true, blank = false). Første rekke og første kolonne er «overskrifter» slik at tabellen skal bli lettere å lese.



	0	1	2	3	4	5
0		T		T		
1	T		T		T	T
2		T				T
3	T					
4		T				T
5		T	T		T	

I oppgåva vil vi kan anta at nodane er nummerert frå 0 til n – 1 slik at vi enkelt kan representere ei nabomatrise som ein to-dimensjonal tabell.

Nedanfor er det gitt deklarasjon av ein to-dimensjonal tabell og den første linja av metodane som skal lagast. Vi har også vist korleis matrisa i eksemplet blir representert i Java.

```
public class Graf {
      // matrisa i eksemplet over
      private boolean[][] nabomatrise = {
                  { false, true, false, true, false, false },
                  { true, false, true, false, true, true },
                  { false, true, false, false, false, true },
                  { true, false, false, false, false, false },
                  { false, true, false, false, false, true },
                  { false, true, true, false, true, false } };
      …
      public boolean erNaboer(int u, int w) {…}

      public int grad(int v) {…}

      public boolean public int antallLokker() {…}

      public boolean uavhengigMengde (int[] s) {…}

}
```

a) Lag metoden `erNaboer(int u, int w)` som returnerer `true` om node u og node w er naboar.

b) Lag metoden `grad(int v)` for å finne gradtalet for ein node gitt som parameter. Gradtalet er antal naboar til noden.
Eksempel: Node 0 har 2 naboar (nodane 1 og 3).

c) Ei løkke i ein graf er ein kant frå ein node til seg sjølv. Om node nr v har ei løkke, så vil element (v, v) i matrisa vere `true` . Lag metoden `antallLokker()` som returnerer antal løkker i grafen.

d) Ei uavhengig mengde i ein graf er ei mengde nodar der ingen av nodane i mengda er nabo med ein annan node i mengda.
Eksempel: Mengda {0, 2, 4} er ei uavhengig mengde i grafen ovanfor.

Lag metoden `uavhengigMengde(int[] s)` som avgjer om mengda av nodar gitt som parameter er ei uavhengig mengde i grafen.

Oppgåve 5 (Vekt 10%)

Vi har ei liste av ord der kvart ord kan førekomme fleire gonger. Vi ønskjer å lage ei frekvensoversikt (antal gonger kvart ord førekjem).
Eksempel: Dersom lista av ord er:

```
{"er", "det", "alle", "er", "det", "det"}
```

så blir frekvensoversikta:

```
er: 2
det: 3
alle: 1
```

Vi skal bruke dei forhåndsdefinerte klassane `ArrayList` og `HashMap` i Java for å løyse oppgåva. Du har gitt starten av programmet (du treng ikkje gjenta dette i svaret). Du finn deler av API-dokumentasjonen for `HashMap` som vedlegg.

Starten av programmet:

```
public static void main(String[] args) {
      ArrayList<String> listeAvOrd = new ArrayList<String>();
      HashMap<String, Integer> fo = new HashMap<String, Integer>();

      //b Lag frevensoversikten

      //a Les inn eit ord frå brukar og skriv ut tilhøyrande frekvens
}
```

a) Anta at du har ein HashMap som inneheld ord med tilhøyrande frekvens (skal gjerast i neste delspørsmål). Les inn eit ord frå brukaren og skriv ut kor mange gonger det finst (svaret kan vere 0).

b) Gå gjennom array-lista `listeAvOrd` ved bruk av ei utvida for-løkke og lag frekvensoversikta. Første gong du finn ordet, set du ordet inn i HashMap'en med frekvens 1. Om ordet finst frå før, aukar du frekvensen med ein.

Lukke til!

Vedlegg: Deler av API-dokumentasjon for Class HashMap<K,V>

void	**clear**() Removes all of the mappings from this map.
boolean	**containsKey**(**Object** key) Returns true if this map contains a mapping for the specified key.
boolean	**containsValue**(**Object** value) Returns true if this map maps one or more keys to the specified value.
V	**get**(**Object** key) Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.
V	**getOrDefault**(**Object** key, **V** defaultValue) Returns the value to which the specified key is mapped, or defaultValue if this map contains no mapping for the key.
boolean	**isEmpty**() Returns true if this map contains no key-value mappings.
Set<**K**>	**keySet**() Returns a **Set** view of the keys contained in this map.
V	**put**(**K** key, **V** value) Associates the specified value with the specified key in this map.
void	**putAll**(**Map**<? extends **K**,? extends **V**> m) Copies all of the mappings from the specified map to this map.
V	**remove**(**Object** key) Removes the mapping for the specified key from this map if present.
boolean	**remove**(**Object** key, **Object** value) Removes the entry for the specified key only if it is currently mapped to the specified value.
V	**replace**(**K** key, **V** value) Replaces the entry for the specified key only if it is currently mapped to some value.
boolean	**replace**(**K** key, **V** oldValue, **V** newValue) Replaces the entry for the specified key only if currently mapped to the specified value.
int	**size**() Returns the number of key-value mappings in this map.