Høgskulen på Vestlandet

EKSAMENSOPPGAVE/EKSAMENSOPPGÅVE

Emnekode: DAT100

Emnenavn/Emnenamn: Grunnleggende Programmering

Utdanning/kull/klasse:
Dataingeniør + Informasjonsteknologi / V2018 /
1Data og 1Informajonsteknologi

Dato: 6. juni 2018

Eksamensform: Skriftlig

Eksamenstid: 4 klokketimer

Antall eksamensoppgaver/ Tal på eksamensoppgåver: 5

Antall vedlegg/ Tal på vedlegg: Ingen vedlegg

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 15%)

a) Hva blir skrevet ut når metodekallet **`ma()`** blir utført?

```
public static void ma() {
	double a = 4.1;
	double b = 2.7;
	String s = "abcde";

	System.out.println(4 + 5 / 2);
	System.out.println(13 % 4);
	System.out.println(!(a > b));
	System.out.println((a < b) || (a != 0));
	System.out.println('a' == 'A');
	System.out.println("1" + "7");
	System.out.println(s.charAt(2));

}
```

b) Hva blir skrevet ut når metodekallet **`mb(3)`** blir utført?

```
public static void mb(int n) {
	for (int i = 0; i < n; i++) {
		for (int j = i; j < n; j++) {
			System.out.print("#");
		}
		System.out.println();
	}
}
```

c) Hva blir skrevet ut når metodekallet **`mc()`** blir utført?

```
public static void mc() {
	int i = 5;
	do {
		System.out.println(i);
		i = i - 2;
	} while (i > 0);

}
```

d) Hva blir skrevet ut når metodekallet **`md()`** blir utført?

```
public static int d(int a, int b) {
        a = 2 * a;
        b = b / 2;
        System.out.println(" I metode d: a = " + a + ", b = " + b);
        return a * b;
}

public static void md() {
        int a = 3;
        int b = 5;

        int c = d(a, b);
        System.out.println("a = " + a + ", b = " + b + ", c = " + c);

        c = d(b, a);
        System.out.println("a = " + a + ", b = " + b + ", c = " + c);
}
```

Oppgave 2 (vekt 20%)

Golf er en sport der spillerne prøver å slå en ball ned i et hull på så få slag som mulig. En golfklubb ønsker et system der spillere kan få en oversikt over resultatet under/etter en runde (består som oftest av 18 hull).

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

- Objektvariabler som bare skal være synlige innenfor klassen: `dato` (streng), `navn` (streng), `antallFerdige` (heltall, som er antall hull vi har resultater for), og `resTab` (tabell av heltall som er antall slag spilleren bruker på hvert hull).
- `get`- og `set`-metoder for `navn` og `dato` og `get`-metode for `antallFerdige`.
- Konstruktør med parametere for objektvariablene `navn` og `dato`. Videre skal konstruktøren ha en parameter `antall` som gir antall hull som skal spilles og som skal brukes til å opprette tabellen for resultater, `resTab`.

b) Lag en metode `void leggTilRes(int r)` som legger til et nytt resultat i tabellen `resTab`. Resultatene blir lagret sammenhengende fra starten av tabellen slik at det nye elementet skal inn på neste ledige plass.

c) Lag en metode `int sumSlag()` som returnerer summen av antall slag på de hullene som er spilt ferdig.

d) Lag en metode `boolean erFerdig()` som returnerer `true` dersom spilleren har registrert resultat på alle hullene og `false` ellers.

e) Lag en metode `String toString()` som returnerer en streng på formen:

```
"Ole, 01.06.2018, Sum slag: 8 etter 2 av 18 hull\n"
```

Eksempelet viser status etter at spilleren har fått resultat på de 2 første av 18 hull.

f) I golf vil hvert hull ha et forventet antall slag (for en god spiller) som blir kalt par. Lag en metode `int antallOverPar(int[] parTab)` som returnerer antall hull der man har brukt flere slag enn par. Tabellen `parTab` gir forventet antall slag på hullene (samme rekkefølge som i `resTab`). Eksempel:

	0	1	2	3	4	5	6	7	8
parTab	6	4	3	3	5	5	5	4	6
resTab	5	4	3	5	5	6	3	6	6

I eksempelet vil svaret være 3 (spilleren har brukt flere slag enn par på hull 3, 5 og 7).

g) I golf gir vi navn på resultatet på et hull i forhold til hva som er par på hullet. Navnene på resultater som er bedre enn par (færre slag) er som følger: en betre enn par – "birdie", to bedre enn par – "eagle", tre bedre enn par – "albatross", fire bedre enn par – "kondor" og så ett slag – "hole in one". Merk at et slag skal være "hole in one" selv om det også vil passe til et av de andre navnene.

Lag en metode `void visScore(int[] parTab)` som skriver ut resultatet på de hullene som er spilt. For hvert hull skal metoden skrive ut: nummer på hullet, resultat, par på hullet og så navn på resultatet om det er par eller bedre. Om spilleren har brukt mer enn par, skal ingenting skrives ut for navn på resultat.

Ei utskrift kan sjå ut som følger:

```
Ole, 01.06.2018, Sum slag: 28 etter 9 av 18 hull.
Hull 1: 5 (4) -
Hull 2: 4 (4) - Par
Hull 3: 3 (4) - Birdie
Hull 4: 6 (4) -
Hull 5: 2 (4) - Eagle
Hull 6: 2 (5) - Albatross
Hull 7: 2 (6) - Kondor
Hull 8: 1 (3) - Hole in one
Hull 9: 3 (3) - Par
```

h) Lag en main-metode, der du oppretter to spillere og registrer et resultat for hver av spillerne. Vi tenker oss at resten av resultatene blir registrert. Skriv ut resultatene for de to spillerne og skriv ut hvem som er best (brukt færrest slag), eventuelt at det er uavgjort (de to spillerne har brukt like mange slag).

Oppgave 3 (vekt 20%)

I denne oppgaven skal du implementere noen klasser for et system som skal brukes til å administrere reservasjoner av rom.

a) Lag klassen `Rom` med objektvariabler `bygg` (tegn) og `nummer` (heltall) som bare skal være synlige innenfor klassen. Klassen skal videre ha en konstruktør som kan gi verdi til begge objektvariablene samt `get-` og `set`-metoder for de to objektvariablene.

b) Lag klassen `Reservasjon` som har følgende objektvariabler: `rom` (av typen `Rom`) og `person` (av typen `Person`) som bare skal være synlige innenfor klassen. Vi antar at det bare er mulig å reservere rom for en hel dag.

   Du kan anta at klassen `Person` (som skal brukes til å angi navnet på den personen som eier reservasjonen) finnes fra før.

c) Implementer en konstruktør `public Reservasjon(Rom rom, Person person)` i klassen `Reservasjon` som kan gi verdi til alle objektvariablene i klassen. Du skal ikke implementere `set`/`get` metoder, men kan bruke de der det trengs i oppgavene nedenfor.

d) Implementer en metode `String toString()` i klassen `Reservasjon` som returnerer en streng med informasjon om reservasjonen. Eksempelvis skal en reservasjon av rom 514 i bygg E gjort av personen Vidar Hove gi følgende streng:

   `"[ E514 ] Vidar Hove"`

   Du kan anta at der finnes en `toString()` metode i klassen `Person` som returnerer navn på personen og at der finnes en metode `toString()` i klassen `Rom` som returnerer en streng med informasjon om rommet. Eksempelvis vil romnummer 514 i E bygget gi følgende streng: `"[ E 514 ]"`

e) Implementer en subklasse `UkesReservasjon` av klassen `Reservasjon` som i tillegg til de objektvariablene som blir arvet, har en objektvariabel `dager` som er en tabell av sannhetsverdier. Tabellen skal brukes til å angi hvilke dager i uken reservasjonen gjelder for. Posisjon 0 i tabellen skal svare til mandag, 1 til tirsdag osv. fram til søndag som har posisjon 6.

f) Implementer en konstruktør i klassen `UkesReservasjon` som kan gi verdi til alle objektvariablene arvet fra superklassen og som videre oppretter en tabell og setter objektvariabelen `dager` til en tabell av lengde 7 (et element for hver dag i uken) der alle elementene har verdien `false`.

g) Implementer en metode `String toString()` i klassen `UkesReservasjon` som skriver ut informasjon om reservasjonen. For eksempel, så skal reservasjon av rom 514 i bygg E på mandag, onsdag og torsdag gi følgende streng:

`"[ E514 ] Vidar Hove\nmandag onsdag torsdag\n"`

Du kan anta at det finnes en metode `public String tilDag(int d)` i klassen `UkesReservasjon` som gitt indeks (posisjon) for en dag returnerer en streng som svarer til navnet på dagen. Eksempelvis vil `tilDag(0)` returnere "`mandag`" og `tilDag(6)` returnere "`søndag`".

h) Implementer en metode:

```
public static UkesReservasjon nyReservasjon(String navn,
                                            char bygg, int nummer,
                                            boolean[] dager)
```

som oppretter et nytt ukesreservasjons-objekt fra parameterne til metoden. Metoden skal opprette de nødvendige `Person` og `Rom` objekter som del av dette.

Du kan anta at det finnes en konstruktør for Person-klassen som tar et navn (streng) som parameter og at klasssen `UkesReservasjon` har en set-metode for objektvariabelen `dager`.

Oppgave 4 (vekt 25%)

Spillet «Fire på rad» blir spilt av to spillere på et stående rektangulært brett med K kolonner og R rekker inndelt i felt. De to spillerne som har hver sin farge på brikkene (eksempelvis gul og blå), veksler på å legge inn brikker i toppen av brettet. Når en brikke blir lagt inn, vil den falle ned så langt den kan. Spilleren som først får fire brikker på rad vannrett, loddrett eller diagonalt vinner spillet.

To eksempel på tilstander (A og B) i spillet er gitt nedenfor for et brett med 6 kolonner og 4 rekker der tegnet ´B´ i et felt angir en blå brikke og ´G´ en gul brikke. Tallene i toppen og til venstre angir posisjoner (kolonne og rekke indekser) på brettet.

Tilstand A

	0	1	2	3	4	5
0						
1						
2	B		G	G	G	
3	B	B	G	G	B	B

Tilstand B

	0	1	2	3	4	5
0						
1						
2	B		G	G	G	G
3	B	B	G	G	B	B

Om det er spilleren med gule brikker sin tur i tilstand A og han/hun velger å plassere en brikke i kolonnen lengst til høyre, vil vi komme til tilstand B der spiller G har vunnet spillet siden denne spilleren har fire brikker på rad i rekke 2.

I Java kan vi representere en tilstand i spillet som en to-dimensjonal tabell av tegn, der ´G´ representerer en gul brikke, ´B´ en blå brikke og ´I´ at der ikke er noen brikke i feltet. Tilstand A ovenfor vil da kunne representeres som:

```
public char[][] brett =
            { { 'I', 'I', 'I', 'I', 'I', 'I'},
              { 'I', 'I', 'I', 'I', 'I', 'I'},
              { 'B', 'I', 'G', 'G', 'G', 'I'},
              { 'B', 'B', 'G', 'G', 'B', 'B'} };
```

Du skal lage noen metoder som bruker den to-dimensjonale tabellen `brett`. For å få full score, må metodene fungere generelt, det vil si for brett med annen størrelse enn 6x4 felt.

a) Skriv en metode `public boolean harTegn(int r, int k, char c)` som returnerer `true` dersom brettet har tegnet `c` i rekke `r`, kolonne `k`, og `false` ellers.

b) Skriv en metode `public boolean ingenBrikke(int r, int k)` som returnerer `true` dersom feltet i rekke `r`, kolonne `k` ikke har noen brikke (dvs. inneholder tegnet 'I').

c) Skriv metoden `public void visTilstand()` som skriver ut tilstanden av brettet. Metoden skal skrive ut 'B' om der er en blå brikke i et gitt felt og 'G' om der er en gul brikke. Om der ikke er noen brikke, skal metoden skrive ut en blank (" ").

Eksempel: i tilstand A skal utskriften bli (de to første linjene består av blanke):

```
B GGG
BBGGBB
```

d) Skriv metoden `public int hvilkenRekke(int k)` som gitt en kolonneindeks k, finner indeks (posisjon) på den rekken som en brikke vil plassere seg på om den blir lagt i kolonne k. Dersom kolonnen er fylt (i eksempelet at der er 4 brikker i kolonnen), skal metoden returnere -1.

Eksempel: i tilstand A skal `hvilkenRekke(0)` returnere 1 og `hvilkenRekke(5)` skal returnere 2.

e) Skriv metoden `public void plasserBrikke(int k, char c)` som oppdaterer brettet slik at brikken gitt ved tegnet `c` blir plassert i kolonne `k`.

Eksempel: `plasserBrikke(5,'G')` med et brett i tilstand A skal resultere i et brett i tilstand B.

f) Skriv metoden `public boolean sjekkVannrett(int r, int k, char c)` som returnerer `true` om spilleren gitt ved tegnet `c` har fire på rad vannrett på rekke `r` fra kolonne `k` og til høyre (og `false` ellers). Du kan anta at verdien til `k` er slik at der er minst tre kolonner etter kolonne `k`.

g) Skriv metoden `public boolean fireVannrett(int r, char c)` som returnerer `true` om spilleren gitt ved tegnet `c` har fire på rad i rekke `r`.

Oppgave 5 (vekt 20%)

En aktuator er et samlebegrep for ting som vi kan aktivere og deaktivere, som for eksempel en lampe, et varmepanel, en motor eller en alarm. Når eksempelvis et varmepanel blir aktivert vil det varme og når det blir deaktivert vil det slutte å varme.

I Java kan vi representere operasjoner (metoder) på en aktuator som et interface (kontrakt):

```
public interface IAktuator {

      public void aktiver();

      public void deaktiver();

}
```

Nedenfor skal du implementere noen interfaces, klasser og metoder som kan brukes til å kontrollere en aktuator.

a) Skriv et interface `IKontrol` med metoder `on` og `off`. Metodene skal ikke ha parametere og ikke returnere noen verdi. Videre skal interfacet ha en metode `getStatus` som senere skal brukes til å bestemme om enheten er aktivert eller deaktivert. Du skal selv bestemme type for `getStatus` metoden.

b) Lag en klasse `VarmePanel` som implementerer `IKontrol` interfacet og som kan brukes til styring og kontroll av et varmepanel. Klassen skal i tillegg til metodene fra interfacet implementere følgende:

   i. en objektvariabel `status` (sannhetsverdi) som skal brukes til å angi om varmepanelet er på eller ikke.
   ii. en objektvariabel `aktuator` av typen `IAktuator` som representerer den enheten /aktuatoren (her et varmepanel) som blir styrt.
   iii. En konstruktør `public VarmePanel(IAktuator aktuator)` som setter objektvariabelen `aktuator` lik parameteren som er gitt til konstruktøren og som setter objektvariabelen `status` lik `false`;

   Implementasjonen av metoden `on` i klassen skal bare slå på (aktivere) panelet (aktuatoren) om det ikke allerede er aktivert og tilsvarende skal metoden `off` bare slå av (deaktivere) panelet om det ikke allerede er slått av.

c) En sensor er en enhet som kan måle/observere en verdi, eksempelvis temperatur, vibrasjon og luftfuktighet.

En sensor kan måle verdier som er flyttall og kan representeres som følgende interface:

```
public interface ISensor {

      public double leseav();

}
```

Implementer metoden:

```
public static void smartPanel(ISensor sensor,
                              VarmePanel panel, double grense)
```

som med 60 sekunders intervall leser av (temperatur) sensoren og viss den aktuelle temperaturen er mindre en ønsket (temperatur) `grense`, så skal varmepanelet slå seg på (ellers skal panelet slå seg av).

Du kan anta at der finnes en metode `void static vent(int secs)` som kan få Java-programmet til å vente (sove) i et gitt antall sekunder.

Lykke til!

Høgskulen på Vestlandet

EKSAMENSOPPGAVE/EKSAMENSOPPGÅVE

Emnekode: DAT100

Emnenavn/Emnenamn: Grunnleggende Programmering

Utdanning/kull/klasse:
Dataingeniør + Informasjonsteknologi / V2018 /
1Data og 1Informajonsteknologi

Dato: 6. juni 2018

Eksamensform: Skriftlig

Eksamenstid: 4 klokketimer

Antall eksamensoppgaver/ Tal på eksamensoppgåver: 5

Antall vedlegg/ Tal på vedlegg: Ingen vedlegg

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 15%)

a) Kva blir skrive ut når metodekallet **`ma()`** blir utført?

```
public static void ma() {
	double a = 4.1;
	double b = 2.7;
	String s = "abcde";

	System.out.println(4 + 5 / 2);
	System.out.println(13 % 4);
	System.out.println(!(a > b));
	System.out.println((a < b) || (a != 0));
	System.out.println('a' == 'A');
	System.out.println("1" + "7");
	System.out.println(s.charAt(2));

}
```

b) Kva blir skrive ut når metodekallet **`mb(3)`** blir utført?

```
public static void mb(int n) {
	for (int i = 0; i < n; i++) {
		for (int j = i; j < n; j++) {
			System.out.print("#");
		}
		System.out.println();
	}
}
```

c) Kva blir skrive ut når metodekallet **`mc()`** blir utført?

```
public static void mc() {
	int i = 5;
	do {
		System.out.println(i);
		i = i - 2;
	} while (i > 0);

}
```

d) Kva blir skrive ut når metodekallet **`md()`** blir utført?

```
public static int d(int a, int b) {
	a = 2 * a;
	b = b / 2;
	System.out.println(" I metode d: a = " + a + ", b = " + b);
	return a * b;
}

public static void md() {
	int a = 3;
	int b = 5;

	int c = d(a, b);
	System.out.println("a = " + a + ", b = " + b + ", c = " + c);

	c = d(b, a);
	System.out.println("a = " + a + ", b = " + b + ", c = " + c);
}
```

Oppgåve 2 (vekt 20%)

Golf er ein sport der spelarane prøver å slå ein ball ned i eit hol på så få slag som mulig. Ein golfklubb ønskjer eit system der spelarar kan få ei oversikt over resultatet under/etter ein runde (består som oftast av 18 hol).

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

- Objektvariablar som berre skal vere synlege innanfor klassen: `dato` (streng), `navn` (streng), `antallFerdige` (heiltal, som er antal hol vi har resultat for), og `resTab` (tabell av heiltal som er antal slag spelaren brukar på kvart hol).
- `get`- og `set`-metodar for `navn` og `dato` og `get` metode for `antallFerdige`.
- Konstruktør med parametrar for objektvariablane `navn` og `dato`. Vidare skal konstruktøren ha ein parameter `antall` som gir antal hol som skal spelast og som skal brukast til å opprette tabellen for resultat, `resTab`.

b) Lag ein metode `void leggTilRes(int r)` som legg til eit nytt resultat i tabellen `resTab`. Resultata blir lagra samanhengande frå starten av tabellen slik at det nye elementet skal inn på neste ledige plass.

c) Lag ein metode `int sumSlag()` som returnerer summen av antal slag på dei hola som er spelt ferdig.

d) Lag ein metode `boolean erFerdig()` som returnerer `true` dersom spelaren har registrert resultat på alle hola og `false` elles.

e) Lag ein metode `String toString()` som returnerer ein streng på forma:

```
"Ole, 01.06.2018, Sum slag: 8 etter 2 av 18 hull\n"
```

Eksempelet viser status etter at spelaren har fått resultat på dei 2 første av 18 hol.

f) I golf vil kvart hol ha eit forventa antal slag (for ein god spelar) som blir kalla par. Lag ein metode `int antallOverPar(int[] parTab)` som returnerer antal hol der ein har brukt fleire slag enn par. Tabellen `parTab` gir forventa antal slag på hola (same rekkefølge som i `resTab`). Eksempel:

	0	1	2	3	4	5	6	7	8
parTab	6	4	3	3	5	5	5	4	6
resTab	5	4	3	5	5	6	3	6	6

I eksempelet vil svaret vere 3 (spelaren har brukt fleire slag enn par på hol 3, 5 og 7).

g) I golf gir vi namn på resultatet på eit hol i forhold til kva som er par på holet. Namna på resultat som er betre enn par (færre slag) er som følgjer: ein betre enn par – "birdie", to betre enn par – "eagle", tre betre enn par – "albatross", fire betre enn par – "kondor" og så eit slag – "hole in one". Merk at eit slag skal vere "hole in one" sjølv om det også vil passe til eit av dei andre namna.

Lag ein metode `void visScore(int[] parTab)` som skriv ut resultatet på dei hola som er spelt. For kvart hol skal metoden skrive ut: nummer på holet, resultat, par på holet og så namn på resultatet om det er par eller betre. Om spelaren har brukt meir enn par, skal ingenting skrivast ut for namn på resultat.

Ei utskrift kan sjå ut som følgjer:

```
Ole, 01.06.2018, Sum slag: 28 etter 9 av 18 hull.
Hull 1: 5 (4) -
Hull 2: 4 (4) - Par
Hull 3: 3 (4) - Birdie
Hull 4: 6 (4) -
Hull 5: 2 (4) - Eagle
Hull 6: 2 (5) - Albatross
Hull 7: 2 (6) - Kondor
Hull 8: 1 (3) - Hole in one
Hull 9: 3 (3) - Par
```

h) Lag ein main-metode, der du opprettar to spelarar og registrer eit resultat for kvar av spelarane. Vi tenker oss at resten av resultata blir registrert. Skriv ut resultata for dei to spelarane og skriv ut kven som er best (brukt færrast slag), eventuelt at det er uavgjort (dei to spelarane har brukt like mange slag).

Oppgåve 3 (vekt 20%)

I denne oppgåva skal du implementere nokre klassar for eit system som skal brukast til å administrere reservasjonar av rom.

a) Lag klassen `Rom` med objektvariablar `bygg` (teikn) og `nummer` (heiltal) som berre skal vere synlege innanfor klassen. Klassen skal vidare ha ein konstruktør som kan gi verdi til begge objektvariablane og `get`- og `set`-metodar for dei to objektvariablane.

b) Lag klassen `Reservasjon` som har følgande objektvariablar: `rom` (av typen `Rom`) og `person` (av typen `Person`) som berre skal vere synlege innanfor klassen. Vi antar at det berre er mulig å reservere rom for ein heil dag.

   Du kan anta at klassen `Person` (som skal brukast til å angi namnet på den personen som eig reservasjonen) finst frå før.

c) Implementer ein konstruktør `public Reservasjon(Rom rom, Person person)` i klassen `Reservasjon` som kan gi verdi til alle objektvariablane i klassen. Du skal ikkje implementere `set`/`get` metodar, men kan bruke dei der det trengs i oppgåvene nedanfor.

d) Implementer ein metode `String toString()` i klassen `Reservasjon` som returnerer ein streng med informasjon om reservasjonen. Eksempelvis skal ein reservasjon av rom 514 i bygg E gjort av personen Vidar Hove gi følgande streng:

   `"[ E514 ] Vidar Hove"`

   Du kan anta at der finst ein `toString()` metode i klassen `Person` som returnerer namn på personen og at der finst ein metode `toString()` i klassen `Rom` som returnerer ein streng med informasjon om rommet. Eksempelvis vil romnummer 514 i E bygget gi følgande streng: `"[ E 514 ]"`

e) Implementer ein subklasse `UkesReservasjon` av klassen `Reservasjon` som i tillegg til dei objektvariablane som blir arva, har ein objektvariabel `dager` som er ein tabell av sannheitsverdiar. Tabellen skal brukast til å angi kva dagar i veka reservasjonen gjeld for. Posisjon 0 i tabellen skal svare til mandag, 1 til tirsdag osv. fram til søndag som har posisjon 6.

f) Implementer ein konstruktør i klassen `UkesReservasjon` som kan gi verdi til alle objektvariablane arva frå superklassen og som vidare oppretter ein tabell og set objektvariablen `dager` til ein tabell av lengde 7 (eit element for kvar dag i veka) der alle element har verdien `false`.

g) Implementer ein metode `String toString()` i klassen `UkesReservasjon` som skriv ut informasjon om reservasjonen. For eksempel, så skal reservasjon av rom 514 i bygg E på mandag, onsdag og torsdag gi følgande streng:

`"[ E514 ] Vidar Hove\nmandag onsdag torsdag\n"`

Du kan anta at der finst ein metode `public String tilDag(int d)` i klassen `UkesReservasjon` som gitt indeks (posisjon) for ein dag returnerer ein streng som svarar til namnet på dagen. Eksempelvis vil `tilDag(0)` returnere `"mandag"` og `tilDag(6)` returnere `"søndag"`.

h) Implementer ein metode:

```
public static UkesReservasjon nyReservasjon(String navn,
                                            char bygg, int nummer,
                                            boolean[] dager)
```

som opprettar eit nytt ukesreservasjons-objekt frå parametrane til metoden. Metoden skal opprette dei nødvendige `Person` og `Rom` objekt som del av dette.

Du kan anta at det finst ein konstruktør for Person-klassen som tek eit namn (streng) som parameter og at klasssen `UkesReservasjon` har ein set-metode for objektvariabelen `dager`.

Oppgåve 4 (vekt 25%)

Spelet «Fire på rad» blir spelt av to spelarar på eit ståande rektangulært brett med K kolonnar og R rekker inndelt i felt. Dei to spelarane som har kvar sin farge på brikkene (eksempelvis gul og blå), vekslar på å legge inn brikker i toppen av brettet. Når ei brikke blir lagt inn, vil den falle ned så langt den kan. Spelaren som først får fire brikker på rad vassrett, loddrett eller diagonalt vinn spelet.

To eksempel på tilstandar (A og B) i spelet er gitt nedanfor for eit brett med 6 kolonnar og 4 rekker der teiknet ´B´ i eit felt angir ei blå brikke og ´G´ ei gul brikke. Tala i toppen og til venstre angir posisjoner (kolonne og rekke indeksar) på brettet.

Tilstand A

	0	1	2	3	4	5
0						
1						
2	B		G	G	G	
3	B	B	G	G	B	B

Tilstand B

	0	1	2	3	4	5
0						
1						
2	B		G	G	G	G
3	B	B	G	G	B	B

Om det er spelaren med gule brikker sin tur i tilstand A og han/ho vel å plassere ei brikke i kolonnen lengst til høgre, vil vi kome til tilstand B der spelar G har vunne spelet sidan denne spelaren har fire brikker på rad i rekke 2.

I Java kan vi representere ein tilstand i spelet som ein to-dimensjonal tabell av teikn, der ´G´ representerer ei gul brikke, ´B´ ei blå brikke og ´I´ at der ikkje er noko brikke i feltet. Tilstand A ovanfor vil då kunne representerast som:

```
public char[][] brett =
            { { 'I', 'I', 'I', 'I', 'I', 'I'},
              { 'I', 'I', 'I', 'I', 'I', 'I'},
              { 'B', 'I', 'G', 'G', 'G', 'I'},
              { 'B', 'B', 'G', 'G', 'B', 'B'} };
```

Du skal lage nokre metodar som brukar den to-dimensjonale tabellen `brett`. For å få full score, må metodane fungere generelt, det vil seie for brett med anna storleik enn 6x4 felt.

a) Skriv ein metode `public boolean harTegn(int r, int k, char c)` som returnerer `true` dersom brettet har teiknet `c` i rekke `r`, kolonne `k`, og `false` elles.

b) Skriv ein metode `public boolean ingenBrikke(int r, int k)` som returnerer `true` dersom feltet i rekke `r`, kolonne `k` ikkje har noko brikke (dvs. inneheld teiknet 'I').

c) Skriv metoden public `void visTilstand()` som skriv ut tilstanden av brettet. Metoden skal skrive ut 'B' om der er ei blå brikke i eit gitt felt og 'G' om der er ei gul brikke. Om der ikkje er noko brikke, skal metoden skrive ut ein blank (" ").

Eksempel: i tilstand A skal utskrifta bli (dei to første linjene består av blanke):

```
B GGG
BBGGBB
```

d) Skriv metoden `public int hvilkenRekke(int k)` som gitt ein kolonneindeks k, finn indeks (posisjon) på den rekka som ei brikke vil plassere seg på om den blir lagt i kolonne k. Dersom kolonnen er fylt (i eksempelet at der er 4 brikker i kolonnen), skal metoden returnere -1.

Eksempel: i tilstand A skal `hvilkenRekke(0)` returnere 1 og `hvilkenRekke(5)` skal returnere 2.

e) Skriv metoden `public void plasserBrikke(int k, char c)` som oppdaterer brettet slik at brikka gitt ved teiknet `c` blir plassert i kolonne `k`.

Eksempel: `plasserBrikke(5,'G')` med eit brett i tilstand A skal resultere i eit brett i tilstand B.

f) Skriv metoden `public boolean sjekkVannrett(int r, int k, char c)` som returnerer true om spelaren gitt ved teiknet `c` har fire på rad vassrett på rekke `r` frå kolonne `k` og til høgre (og `false` elles). Du kan anta at verdien til `k` er slik at der er minst tre kolonner etter kolonne `k`.

g) Skriv metoden `public boolean fireVannrett(int r, char c)` som returnerer `true` om spelaren gitt ved teiknet `c` har fire på rad i rekke `r`.

Oppgåve 5 (vekt 20%)

Ein aktuator er eit samleomgrep for ting som vi kan aktivere og deaktivere, som for eksempel ei lampe, eit varmepanel, ein motor eller ein alarm. Når eksempelvis eit varmepanel blir aktivert vil det varme og når det blir deaktivert vil det slutte å varme.

I Java kan vi representere operasjonar (metodar) på ein aktuator som eit interface (kontrakt):

```
public interface IAktuator {

      public void aktiver();

      public void deaktiver();

}
```

Nedanfor skal du implementere nokre interfaces, klassar og metodar som kan brukast til å kontrollere ein aktuator.

a) Skriv eit interface `IKontrol` med metodar `on` og `off`. Metodane skal ikkje ha parametrar og ikkje returnere nokon verdi. Vidare skal interfacet ha ein metode `getStatus` som seinare skal brukast til å bestemme om einheita er aktivert eller deaktivert. Du skal sjølv bestemme type for `getStatus` metoden.

b) Lag ein klasse `VarmePanel` som implementerer `IKontrol` interfacet og som kan brukast til styring og kontroll av eit varmepanel. Klassen skal i tillegg til metodane frå interfacet implementere følgande:

   i. Ein objektvariabel `status` (sannheitsverdi) som skal brukast til å angi om varmepanelet er på eller ikkje.
   ii. Ein objektvariabel `aktuator` av typen `IAktuator` som representerer den einheita /aktuatoren (her eit varmepanel) som blir styrt.
   iii. Ein konstruktør `public VarmePanel(IAktuator aktuator)` som set objektvariabelen `aktuator` lik parameteren som er gitt med konstruktøren og som set objektvariabelen `status` lik `false`;

   Implementasjonen av metoden `on` i klassen skal berre slå på (aktivere) panelet (aktuatoren) om det ikkje allerede er aktivert og tilsvarande skal metoden `off` berre slå av (deaktivere) panelet om det ikkje allerede er slått av.

c) Ein sensor er ei einheit som kan måle/observere ein verdi, eksempelvis temperatur, vibrasjon og luftfuktigheit.

Ein sensor kan måle verdiar som er flyttal og kan representerast som følgande interface:

```
public interface ISensor {

      public double leseav();

}
```

Implementer metoden:

```
public static void smartPanel(ISensor sensor,
                              VarmePanel panel, double grense)
```

som med 60 sekunders intervall les av (temperatur) sensoren og viss den aktuelle temperaturen er mindre en ønska (temperatur) `grense`, så skal varmepanelet slå seg på (elles skal panelet slå seg av).

Du kan anta at der finst ein metode `void static vent(int secs)` som kan få Java-programmet til å vente (sove) i eit gitt antal sekund.

Lukke til!