HØGSKOLEN I BERGEN
Avdeling for ingeniørutdanning
Institutt for data- og realfag

EKSAMEN I: **TOD062 Grunnleggende programmering**

KLASSE: 1DA og 1DB

DATO: 2. juni 2005

ANTALL OPPGAVER	: 3
ANTALL SIDER	: 5
VEDLEGG	: 1 side
HJELPEMIDDEL	: Ingen. Kalkulator ikke lovlig.
TID	: 09.00 - 14.00 (5 klokketimer)
FAGLÆRERE	: Karl Johan Holmås Svein-Ivar Lillehaug
MERKNADER	: Du kan fritt bruke de spesielle norske bokstavene (æ, ø, å) i navn og eventuelt oversette gitte metode- og variabelnavn til nynorsk/engelsk. Metoder gitt i vedlegget kan benyttes.

Postboks 7030, 5020 Bergen. Tlf. 55 58 75 00, Fax 55 58 77 90
Besøksadr.: Nygårdsgt. 112, Bergen

Oppgave 1 (Vekt 20%)

a) Hva er verdien av sum etter at følgende kode er ferdig kjørt?

```
int sum = 0;
int teller = 0;
while (teller < 5) {
   sum = sum + teller;
   teller++;
}
```

b) Skriv om følgende while-setning med initiering til en for-setning:

```
int sum = 0;
int teller = 1;
while (teller <= 30) {
   sum = sum + teller;
   teller = teller + 2;
}
```

c) Gitt metode **pokus** under:

```
public static int[] pokus (int[] tabell) {
   int[] tab = new int[tabell.length - 1];
   for (int i = tab.length - 1 ; i >= 0; i--) {
      tab[i] = tabell[i] + i;
   }
   return tab;
}
```

Vis innholdet i tabell **hokus** etter at følgende to (2) linjer med javakode har kjørt:

```
int[] tabbe = {1, 2, 3, 4};
int[] hokus = pokus(tabbe);
```

Oppgave 2 (Vekt 50 %)

Du skal i denne oppgaven skrive deler av et lønns- og timeregistreringssytem for et hotell.

a) Definer en klasse (objekttype) **Uketimer** som inneholder opplysninger om hvor mye en ansatt har jobbet i løpet av en uke

Data i klassen:
dagTimer (reell), nattTimer (reell).

Metoder i klassen:
- konstruktør uten parameter (0-/"blank-still alle data)
- konstruktør med alle data som parameter
- lesUkeTimer() som leser inn hvor mange timer av hver kategori en ansatt har jobbet i gitt uke. Timene blir gitt som desimaltall og du kan anta de blir gitt korrekt.
- Hent- og ny- metoder skal skrives for en av instansvariablene. For de øvrige to kan du anta at slike metoder eksisterer.
- finnUkeOvertid() som beregner og returnerer eventuell overtid i timer (timer utover 37,5).
- finnUkeBrutto(double timelønn) som beregner og returnerer bruttoinntekt for uken når timelønnen er som gitt i parameteren.

For alle ansatte gjelder følgende for utbetaling av de enkelte timekategoriene:

dagTimer	Timelønn
nattTimer	Timelønn + 40% tillegg

For eventuell overtid (arbeidstid utover 37,5 timer per uke) får en et tillegg på 50% av ordinær timelønn. Dette kommer i tillegg til eventuelt natt-tillegg.

b) Definer en klasse (objekttype) **Ansatt** som inneholder informasjon om en enkelt ansatt ved hotellet:

Følgende konstanter kan kanskje være til hjelp:

```
private static final String[] KATEGORI = {"", "hotelldirektør", "respesjonsjef", "resepsjonist",
"piccolo","oldfrue", "stuepike", "kokk", "kelner", "ryddehjelp"};

public static final int MAKS_KATEGORI = 9;
```

Data i klassen:
ansattNr (heltall), navn (String), stillingsKode (1. Hotelldirektør, 2. Resepsjonssjef, 3. Resepsjonist, 4. Piccolo, 5. Oldfrue, 6. Stuepike, 7. Kokk, 8. Kelner, 9. Ryddehjelp), timeGrunnlønn (reell), timeTabell (tabell med plass til timelister av klassen **Uketimer** for inntil 52 uker).

Metoder i klassen:
- konstruktør uten parameter (0-/"blank-still alle data) som oppretter en ansatt.
- lesAnsatt () som leser inn alle data for en ansatt bortsett fra timelistene. Kontroller at stillingskoden er gyldig.
- hentStillingsNavn(int kode) som returnerer med betegnelsen for en gitt stillingskode (kode).
- registrertData(int ukenr) som retunerer med verdien **true** dersom det er registrert timer for denne ansatte i den gitte uken, **false** ellers.
- registrerTimer(int ukenr) som leser inn hvor mange timer en ansatt har jobbet gitt uke. Dersom det alt er registrert timer for oppgitt ansatt (dvs. pekeren i aktuell posisjon i timetabell er ulik **null**) så skal det gis varsel "om en virkelig ønsker å endre tidligere registrerte timer" før en eventuell lagring av nye data.
- finnAarsBrutto() som beregner og returnerer bruttoinntekt for ansatt i inneværende år.

Du kan anta at nødvendige hent- og ny- metoder eksisterer for alle instansvariabler.

c) Definer en klasse (objekttype) **Hotell** som inneholder time- og lønns-informasjon for hotellet.

Data i klassen:
navn på hotellet (String), tabell med plass for inntil 100 ansatte av klassen Ansatt, antall ansatte ved hotellet (heltall).

Metoder i klassen:
- konstruktør med firmanavn som parameter som oppretter et hotell uten ansatte.
- nyAnsatt() som registrerer en ny ansatt ved hotellet.
- registerSjekk(int ukeNr) som lister ut ansatte (ansattnummer og navn) som det ikke er registrert timer på i oppgitt uke.
- skrivLønnsoversikt() som skriver ut navn og bruttolønn så langt dette året for samtlige ansatte. I tillegg skal total brutto avlønning (for alle ansatte) skrives ut til slutt.
- visMaksLønn(int kode) som finner og skriver ut den ansatte (navn og bruttolønn) innen en gitt stillingskode som har høyest registrert årsbruttolønn. Dersom det finnes flere ansatte innen en stillingskategori med "lik høyeste" lønn, så holder det at en skriver ut data for den ene av disse.
- stillingsOversikt() som finner og skriver ut hvor mange ansatte hotellet har innen hver av stillingskodene. Eks:

Hotelldirektør:	1
Resepsjonssjef:	1
Resespsjonist:	5
...	

d) Lag til slutt en klasse med en enkel main-metode som:

- oppretter et objekt av klassen Hotell med navn "Hotel Bizzar"
- leser inn data for en del ansatte fra tastatur. Brukeren skal først få spørsmål om antall ansatte som ønskes registrert. Deretter skal data for det ønskede antall ansatte leses inn.

(Vi antar så at systemet har blitt brukt en stund, slik at følgende punkter også har mening)

- skriver ut alle ansatte som det ikke er registrert timer for i uke 32.
- skriver ut total lønnsoversikt.
- skriver ut stillingsoversikt for hotellet.

Oppgave 3 (Vekt 30%)

a) Skriv en metode

```
public static boolean finnesITabell(int[][] tab, int n, int x)
```

som går gjennom den kvadratiske nxn-tabellen referert til med **tab** og tester om verdien x er inneholdt her. Metoden skal returnere med verdien **true** dersom verdien finnes, og **false** dersom den ikke finnes.

b) Skriv en metode

```
public static boolan alleUlike(int[] tab)
```

som går gjennom den fylte, usorterte heltallstabellen referert til med **tab** og tester om det i denne finnes like verdiar eller ikke. Metoden skal returnere med verdien **true** dersom alle verdiene er ulike, og **false** dersom minst to av verdiene er like.

c) Skriv en metode

```
public static int[] finnStørste(int[] tab1,int[] tab2)
```

som skal lage en ny tabell. Du har gitt to heltallstabeller, referert til med **tab1** og **tab2,** som er like lange. Lag en ny tabell med samme lengde der hver posisjon i den nye tabellen inneholder den største verdien av de to verdiene med tilsvarende posisjon i de opprinnelige tabellene (**tab1** og **tab2**). Til slutt skal metoden returnere med referanse til den nye tabellen.

d) Skriv en metode

```
public static void sumRekkerOgKolonner(int[][] tab, int m,
                                                    int n)
```

som tar referanse (**tab**) til en mxn tabell som parameter. Metoden skal først summere de (n-1) første kolonnene og plassera summene i den siste rekka. Deretter summerer metoden hver rekke og plasserer svaret i siste kolonne.

Eksempel med m=3 og n=4:

Før

10	20	30	
1	2	3	

Etter

10	20	30	60
1	2	3	6
11	22	33	66

Lykke til!

Vedlegg til eksamen i TOD062 desember 2004 / juni 2005

Metodar i Terminal-klassen (frå Mughals bok)

Returverdi	Metode
int	lesInt ( )
double	lesDouble ( )
String	lesString ( )
char	lesChar ()

Filbehandling

Viss det blir behov for metodar frå desse klassane vil dei bli gitt som del av oppgåve-teksten.

Klassen Utskrift

Du får kanskje bruk for følgjande metodar frå Utskrift-klassen:

repeterTeikn(char teikn, int antPos)

skrivHjHeiltal(int tal, int antPos)

skrivHjEinDesimal(double tal, int antPos)

skrivHj2Desimal(double tal, int antPos)

skrivHj3Desimal(double tal, int antPos)

skrivJustert(String tekst, int antPos, char justering)

Klassen Random

Konstruktørar
Random ( ) - gir Random-objekt der ein brukar klokka for å gi startverdi
Random (int startverdi) - gir Random-objekt med gitt startverdi

Returverdi	Metode
double	nextDouble( ) - gir eit tilfeldig reelt tal mellom 0 og 1
int	nextInt( ) - gir eit tilfeldig heiltal, også negative

Eit utvalg av metodar frå klassen String

Returverdi	Metode
int	**length ( )** - gir antal teikn i ein streng.
char	**charAt (int pos)** - gir teikn i posisjon **pos** (start på 0).
boolean	**equals (String b)** - a.equals(b) sann om strengane a og b er like, usann elles.
boolean	**equalsIgnoreCase (String b)** - a.equalsIgnoreCase(b) sann om strengane a og b er like uavhengig av små og store bokstavar, usann elles.
int	**compareTo (String s)** - a.compareTo(b) gir eit negativ tal om a kjem først, 0 om strengane er like, og eit positiv tal om b kjem først (alfabetisk / ordboksordning).
int	**indexOf(String s)** - a.indexOf(b) gir -1 om s ikkje finst som delstreng i a, elles gir den første posisjon der b finst i a.
String	**substring (int start, int slutt)** - gir delstrengen frå og med posisjon start til (ikkje med) posisjon slutt.
String	**toLowerCase( ) -** gir same streng, men alle bokstavane er små. (Viss ikkje anna er sagt kan de anta at det også gjeld for dei spesielle norske bokstavane).
String	**toUpperCase( ) -** gir same streng, men alle bokstavane er store. (Viss ikkje anna er sagt kan de anta at det også gjeld for dei spesielle norske bokstavane).