HØGSKOLEN I BERGEN
Avdeling for ingeniørutdanning

EKSAMEN I TOD062 – Grunnleggende programmering

KLASSE : 1. informasjonsteknologi og 1. dataingeniør

DATO : 20. mai 2010

ANTALL OPPGAVER	: 3
ANTALL SIDER	: 8
VEDLEGG	: 2 sider
HJELPEMIDLER	: ingen
TID	: 5 timer (09.00 – 14.00)
MÅLFORM	: Bokmål
FAGLÆRER(E)	: Lars-Petter Helland Sven-Olai Høyland
MERKNADER	: ingen

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

Oppgave 1 (20%)

a)
Vis nøyaktig hva som blir skrevet ut på skjermen når metoden *einA(7)* blir kjørt (metoden blir kjørt med 7 som parameter).

```
public static void einA(int n){
    for(int i = 0; i <= n; i++){
        for(int j = 0; j < n; j = j + 2){
            if (i+j <= n){
                System.out.print('#');
            }
        }
        System.out.println();
    }
}
```

b)
Vi har laget følgende metode for å beregne summen av de n første heltallene (1 + 2 + 3 … + n):

```
public static int einB(int n){
    int sum = 0;
    for (int i = 1; i <= n; i++){
        sum = sum + i;
    }
    return sum;
}
```

Skriv om metoden slik at den i stedet for en for-løkke bruker:

i) en while-løkke
ii) en do – while løkke

c)
Bruk nøstete løkker for å lage metoden

```
public static void einC(int n)
```

som skriver følgende figur når den blir kjørt med 5 som parameter:

```
+****
*+***
**+**
***+*
****+
```

d)
Hva blir skrevet ut når metoden *einD()* blir utført? Gi en kort begrunnelse i hvert av tilfellene. Logiske verdier blir skrevet ut som true / false.

```
public static void einD(){
    int a = 11;
    String s = new String("Høgskolen i Bergen");
    String t = new String("Høgskolen i Bergen");

    System.out.println(a-7/2);
    System.out.println((a%2)==1);
    System.out.println(s.substring(2,5));
    System.out.println(s == t);
    System.out.println(s.equals(t));
}
```

Oppgave 2 (50%)

I denne oppgaven skal du lage noen klasser som skal brukes til timeregistrering av arbeid.

Vi tenker oss at vi har en **Timeliste** som representerer en timeliste for en person. Denne timelisten inneholder en rekke **Timeregistrering**-er som sier noe om når man har jobbet med hva. Hver timeregistrering inneholder blant annet start- og slutt-**Tidspunkt**.

Om du ikke klarer å implementere enkelte av metodene i klassene, kan du likevel anta at de finnes og bruke dem ved behov.

For å løse oppgaven vil det være nyttig å bruke noen metoder i Java-APIet:

- `Integer.parseInt(...) throws NumberFormatException`
- `String.charAt(...)`
- `String.format(...)`
- `String.substring(...)`

Nærmere beskrivelse av hvordan disse virker finner du i vedlegget.

a) (15%)
Lag klassen **Tidspunkt**.

Feltvariabler:

- `time (heltall)`
- `minutt (heltall)`

Private hjelpemetoder (disse er listet først fordi de kan være nyttige i de andre metodene):

- `boolean gyldigMinutt(int minutt)`

  En klassemetode som avgjør om parameteren minutt inneholder en gyldig verdi for minutt. Gyldige verdier må være i intervallet [0..59] og dessuten være hele 5-minutter, dvs. en av verdiene 0, 5, 10, 15, ..., 55.

- `boolean gyldigTime(int time)`

  En klassemetode som avgjør om parameteren time inneholder en gyldig verdi for time. Gyldige verdier må være i intervallet [0..23], dvs. en av verdiene 0, 1, 2, 3, ..., 23.

- `boolean gyldigTidspunkt(String tt_mm)`

  En klassemetode som avgjør om parameteren tt_mm inneholder en gyldig verdi for et tidspunkt. Gyldige verdier er strenger på formatet "tt:mm" der tallverdiene for tt og mm må være gyldig time og minutt. Metoden skal **ikke** kaste et unntak om tt eller mm ikke er tall.

Konstruktører:

- En konstruktør som har to heltall (time og minutt) som parametre. Hvis verdiene ikke er gyldige skal tidspunktet settes til å være *00:00*.

- En konstruktør som har en streng (tt_mm) som parameter. Hvis verdien ikke er gyldig skal tidspunktet settes til å være *00:00*.

Public metoder:

- `double somTimer()`

  En instansmetode som returnerer tidspunktet som et antall timer siden midnatt. F.eks. skal tidspunktet *14:30* gi verdien 14,5.

- `int somMinutter()`

  En instansmetode som returnerer tidspunktet som et antall minutter siden midnatt. F.eks. skal tidspunktet *14:30* gi verdien 870.

- `String toString()`

  En instansmetode som returnerer tidspunktet som en streng på formatet "hh:mm".

b) (15%)
Lag klassen **Timeregistrering**.

Feltvariabler:

- `dato (streng)`
- `start (Tidspunkt)`
- `slutt (Tidspunkt)`
- `beskrivelse (streng)`

Konstruktør:

- En konstruktør som har fire strenger (dato, starttidspunkt, sluttidspunkt og beskrivelse) som parametre. Det er ikke nødvendig å sjekke om parametrene inneholder gyldige verdier. Hvis start er senere enn slutt, skal disse byttes om slik at start ikke blir senere enn slutt.

Public metoder:

- `double beregnAntallTimer()`

  En instansmetode som beregner og returnerer antall timer fra starttidspunkt til sluttidspunkt. F.eks. blir "14:30" – "16:45" beregnet til 2,25 timer.

- `boolean overlapper(Timeregistrering enAnnen)`

  En instansmetode som avgjør om denne timeregistreringen overlapper en annen timeregistrering. F.eks. vil "14:30" – "15:30" og "15:00" – "16:00" på samme dag ha overlapp. (Dette kan være greit å kunne sjekke på for å unngå dobbeltføring av arbeidstid.)

- `String overskrift()`

  En klassemetode som returnerer en overskrift som kan brukes sammen med resultatet av *toString()* i en utskrift, se eksemplet under.

- `String toString()`

  En instansmetode som returnerer timeregistreringen som en streng på formatet vist i eksemplet under.

  Eksemplet viser overskriften (ett metodekall til *overskrift()*) pluss to timeregistreringer (to metodekall til *toString()*).

```
Dato        Start  Slutt  Timer Beskrivelse
27.04.2010  12:30  16:45  4,25  Teste ut litt greier
27.04.2010  16:00  16:30  0,50  Begynne på eksamen i TOD062
```

c) (15%)
Lag klassen **Timeliste**.

Konstanter:

- `TABELLSTORRELSE (heltall)`

Feltvariabler:

- `ansattnavn (streng)`
- `timelonn (heltall)`
- `timeliste (tabell av Timeregistreringer)`
- `antall (heltall)  //antall registreringer`

Konstruktør:

- En konstruktør som har en streng og et heltall (ansattnavn og timelønn) som parametre.

Public metoder:

- ```
  void leggInnTimeregistrering(
        String dato, String start, String slutt, String beskrivelse)
  ```

  En instansmetode som legger inn en timeregistrering i timelisten. Pass på at det er plass i tabellen før innlegging. Hvis det ikke er plass legges den ikke inn.

- `double beregnTotaltAntallTimer()`

  En instansmetode som beregner og returnerer totalt antall registrerte timer.

- `boolean inneholderOverlappendeRegistreringer()`

  En instansmetode som avgjør om timelisten inneholder overlappende registreringer.

- `void skrivUt()`

  En instansmetode som skriver ut timelisten på skjermen som vist i eksemplet under.

```
Timeliste for Lars-Petter
Timelønn: kr 250

Dato        Start  Slutt  Timer Beskrivelse
27.04.2010  16:00  16:30  0,50  Begynne på eksamen
27.04.2010  12:30  16:45  4,25  Teste ut litt greier
27.04.2010  16:45  17:30  0,75  Litt finpuss pluss pluss

Totalt antall timer = 5,50
Total beregnet lønn = kr 1375,00
```

d) (5%)
Lag en **main-metode** som oppretter en timeliste, legger inn de tre registreringene i eksemplet over og skriver ut timelisten på skjermen.
```

Oppgave 3 (30%)

a)
Skriv en metode

```
public static boolean erSortertStigande(int[] tab)
```

som returnerer **true** dersom tabellen er sortert og **false** ellers. NB! Du skal ikke sortere tabellen.

Eksempel:

{1, 7, 15, 21} er sortert stigende
{3, 1, 8, 12} er **ikke** sortert stigende

b)
Skriv en metode som gitt en todimensjonal tabell av heltall som parameter, teller opp hvor mange av elementene som er mindre enn 0, lik 0 og større enn 0. Første linjen av metoden ser slik ut:

```
public static void statistikk(int[][] tab)
```

Eksempel:

-1	0	0
5	-2	7
-7	0	-4

Dersom metoden blir utført med tabellen ovenfor som parameter, skal utskriften være som følger:

```
Antall < 0: 4
Antall = 0: 3
Antall > 0: 2
```

c)
Du skal lage en metode

```
public static int maksAvstand(int[] a)
```

som finner og returnerer den største avstanden mellom to naboelementer i en heltallstabell. Du kan anta at tabellen har minst to elementer. Eksempel:

13	18	12	5	20	22	18	28	30	35
	5	6	7	15	2	4	10	2	5

Øverst ser vi tabellen. Under denne har vi en ny tabell som viser avstanden til elementet foran. I vårt eksempel skal metoden returnere 15. For å få full score på oppgaven skal det ikke lages en ny (hjelpe)tabell.

d)
Gitt klassen *Test* og metoden *main* nedenfor som begge er uten kompileringsfeil. Hva blir skrevet ut når *main* blir utført? Gi en kort begrunnelse.

```
public class Test {
    private int a;
    private int b;
    private int c;

    public Test(int x, int y, int z){
        a = x;
        b = y;
        c = z;
    }

    public void vis(){
        System.out.println("a = " + a);
        System.out.println("b = " + b);
        System.out.println("c = " + c);
    }

    public void f(int a, int x) {
        int b;

        a = 1;
        b = 2;
        c = x;
    }
}

public static void main(String[] args) {
    Test t = new Test(10, 20,30);
    t.vis();
    t.f(7,8);
    System.out.println();
    t.vis();
}
```

Lykke til!

Vedlegg til eksamen i TOD062 våren 2010

Utdrag fra Java API-dokumentasjon for noen nyttige metoder:

Integer:
`public static int parseInt(String s)throws NumberFormatException`

Parses the string argument as a signed decimal integer. The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign `'-'` (`'\u002D'`) to indicate a negative value. The resulting integer value is returned, exactly as if the argument and the radix 10 were given as arguments to the `parseInt(java.lang.String, int)` method.

Parameters:
`s` - a `String` containing the `int` representation to be parsed
Returns:
the integer value represented by the argument in decimal.
Throws:
`NumberFormatException` - if the string does not contain a parsable integer.

String:
`public char charAt(int index)`

Returns the `char` value at the specified index. An index ranges from `0` to `length() - 1`. The first `char` value of the sequence is at index `0`, the next at index `1`, and so on, as for array indexing. If the `char` value specified by the index is a surrogate, the surrogate value is returned.

Parameters:
`index` - the index of the `char` value.
Returns:
the `char` value at the specified index of this string. The first `char` value is at index `0`.
Throws:
`IndexOutOfBoundsException` - if the `index` argument is negative or not less than the length of this string.

```
public static String format(String format, Object... args)
```

Returns a formatted string using the specified format string and arguments. The locale always used is the one returned by `Locale.getDefault()`.

Parameters:
`format` - A format string
`args` - Arguments referenced by the format specifiers in the format string. If there are more arguments than format specifiers, the extra arguments are ignored. The number of arguments is variable and may be zero. The maximum number of arguments is limited by the maximum dimension of a Java array as defined by the Java Virtual Machine Specification. The behaviour on a `null` argument depends on the conversion.
Returns:
A formatted string
Throws:
`IllegalFormatException` - If a format string contains an illegal syntax, a format specifier that is incompatible with the given arguments, insufficient arguments given the format string, or other illegal conditions. For specification of all possible formatting errors, see the Details section of the formatter class specification.
`NullPointerException` - If the `format` is `null`

```
public String substring(int beginIndex,
                        int endIndex)
```

Returns a new string that is a substring of this string. The substring begins at the specified `beginIndex` and extends to the character at index `endIndex - 1`. Thus the length of the substring is `endIndex-beginIndex`.

Examples:

```
"hamburger".substring(4, 8) returns "urge"
"smiles".substring(1, 5) returns "mile"
```

Parameters:
`beginIndex` - the beginning index, inclusive.
`endIndex` - the ending index, exclusive.
Returns:
the specified substring.
Throws:
`IndexOutOfBoundsException` - if the `beginIndex` is negative, or `endIndex` is larger than the length of this `String` object, or `beginIndex` is larger than `endIndex`.