

Skriftlig eksamen i

Grundlæggende Programmering

IT-C, 16. januar 2002
Dansk udgave

Alle ikke-elektroniske hjælpemidler er tilladt. Datamater og elektroniske kommunikationmidler er ikke tilladt.

Eksamen er skriftlig, fire timer, og bedømmes efter 13-skalaen.

Opgavesættet har 8 sider, inklusive denne, og består af fire opgaver, der alle ønskes løst. Opgaverne vægtes som angivet med procenttallene.

Hvis besvarelsen benytter klasser og metoder fra lærebogen, forelæsningsnoterne, eller forelæsningsplancherne, så behøver du ikke at skrive dem af, men du skal give en præcis henvisning med udgave (af bogen), afsnitsnummer, side-tal, osv.

The exam questions are also available in English (separately). Eksamensspørgsmålene findes også på engelsk (på separate ark).

Et godt råd: Gennemlæs opgavesættet inden du begynder at besvare de enkelte opgaver.

Opgave 1 (25%)

1. Hvad udskrives når metode1() nedenfor kaldes? Besvarelsen skal bestå af de linjer som udskrives af kaldene til System.out.println metoden. (Husk at 5 - -4 er 5+4.)

```
public static void metode1(){
    int sum = 0;
    int j = 5;
    int k = -4;
    while (j>k){
        sum = sum + (j-k);
        j = j - k;
        k = k +2;
        System.out.println("sum: " + sum + " j: " + j + " k: " + k);
    }
}
```

2. Hvad udskrives når metode2() nedenfor kaldes? Besvarelsen skal bestå af de linjer der udskrives af kaldene til System.out.println metoden.

```
public static void metode2(){
    int sum = 10;
    for (int i = 10; i>=4; i -= 2){
        for (int j = i; j < 8; j++){
            sum = sum + (j-i);
            System.out.println("sum: " + sum + " i: " + i + " j: " + j);
        }
    }
    System.out.println("endelig sum: " + sum);
}
```

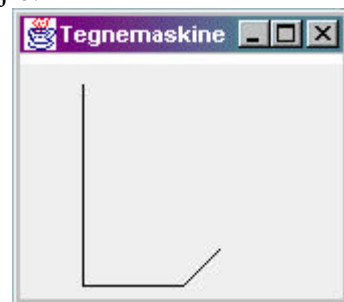
3. Tegnemaskine er en klasse med følgende metoder:

- ?? void move(int afstand) tegner en streg der er afstand pixels (punkter på skærmen) lang, fra det nuværende tegnepunkt i den retning som er gældende.
- ?? void turn(int grader) ændrer tegneretningen et antal grader. Positive grader ændrer retning til venstre (mod uret), negative ændrer retningen mod højre (med uret).
- ?? void jump(int afstand) flytter tegnepunktet afstand pixels – uden at der tegnes en streg.

Et Tegnemaskine-objekt har en intern tilstand, som består af det nuværende tegnepunkt (hvorfra der tegnes) og den nuværende tegneretning (i hvilken retning der tegnes). Læg mærke til, at tegnepunktet flyttes af både move og jump. Forskellen mellem de to metoder er, at den første tegner en streg mens den flytter tegnepunktet, hvorimod den anden gør det ikke. En instans af Tegnemaskine skabes med new Tegnemaskine(), som åbner et vindue hvori der tegnes og sætter det nuværende tegnepunkt i nærheden af det øverste venstre hjørne, med tegneretning nedad.

F.eks. vil invokering af eksempel() nedenfor give tegningen til højre.

```
public static void eksempel(){
    Tegnemaskine tm = new Tegnemaskine();
    tm.move(100);
    tm.turn(90);
    tm.move(50);
    tm.turn(45);
    tm.move(25);
}
```



Skriv en metode `void spiral(int n)` som tegner en spiral som vist nedenfor. Parameteren `n` angiver hvor lang den første linje skal være. Hver linjestykke der tegnes skal være 2 pixels kortere end det foregående, og det sidste linje stykke skal være mindst 2 pixels langt. Husk at tegnemaskinen starter øverst til venstre med tegningsretningen nedad, som i tegningen her.



4. Skriv en metode `void kasser(int n, int k, int s)` som tegner `n` kvadrater inden i hinanden med afstand `k` pixels mellem hver enkelt kvadrat. Sidelængden på det yderste kvadrat skal være `s` pixels. Brug to løkker inden i hinanden, den inderste tegner en kasse, den ydere sørger for at der bliver lavet `n` kasser i alt. Med `n = 7, k = 5, s = 100` bliver tegningen som nedenfor. Husk at tegnemaskinen starter øverst til venstre med tegningsretningen nedad.



(Opgave 1 slut.)

Opgave 2 (30%)

Denne opgave drejer sig om en klasse der indeholder information om *vandløb*. Et vandløb har et navn, en bredde ved udmundingen, et udspring, en længde, og en række sidegrene (bifloder, eller tilløb). Et udspring har et navn, og der strømmer en vis mængde vand ud af det pr. time.

Eksempelvis kan man betragte Suså, der har sit udspring i Gøgsmose; Suså har sidegrenene Kongskilde Bæk, Sorø Å, og Ringsted Å. Kongskilde Bæk har sit udspring i Kongskilden. Sorø Å udspringer i Sorø Sø. Ringsted Å har sit udspring i Gyrstinge Sø. Ringsted Å har en sidegren, Vigerdalså, som udspringer i Valsøllille Sø.

Vi vil repræsentere vandløb og udspring med følgende to klasser, hvoraf den ene, Vandløb, er ufærdig. Opgaven går ud på at gøre klassen Vandløb færdig. Klassen Udspring skal ikke ændres.

```
public class Udspring {
    private String navn;
    private double literPrTime;

    public Udspring(String n, double lpt){
        navn = n;
        literPrTime = lpt;
    }
    public String getNavn(){ return navn; }
    public double getLiterPrTime(){ return literPrTime; }
}

public class Vandløb
{
    private String navn;
    private double bredde; // målt i meter
    private double længde; // målt i kilometer
    private Udspring udspring;

    private Vandløb[] sideGrene = new Vandløb[20]; // maksimalt 20 sidegrene
    private int antalSideGrene = 0;

    // Her kommer konstruktøren og tilgangsmetoderne fra delopgave 1

    public void tilføjSidegren(Vandløb v){
        // Her kommer den kode du skriver i delopgave 2
    }

    public double systemLængde(){
        // Her skal koden fra delopgave 5 indsættes
    }

    public double vandMængde(){
        double sum = udspring.getLiterPrTime();
        for (int sideGren = 0; sideGren < antalSideGrene; sideGren++){
            sum += sideGrene[sideGren].vandMængde();
        }
        return sum;
    }
}
```

1. Skriv en konstruktør for klassen Vandløb der tager parametre så felterne navn, bredde, længde og udspring kan gives en værdi, f.eks skaber.

```
new Vandløb("Ringsted å", 7, 13, new Udspring("Gyrstinge sø", 250) )
```

 et objekt der repræsenterer Ringsted å.
 Lav også 2 tilgangsmetoder så man kan aflæse navn og antal sidegrene.
2. Udfyld metoden void tilføjSidegren(Vandløb v) til klassen Vandløb, der kan bruges til at tilføje en sidegren til et vandløb. Du må gerne antage at der er maksimum 20 sidegrene til et vandløb. Du behøver ikke i metoden at checke om dette maksimumstal på 20 overskrides.
3. Man kan lave Ringsted å fra eksemplet i indledningen af denne delopgave ved følgende erklæringer:


```
Vandløb ringstedÅ = new Vandløb("Ringsted å", 7, 13, new Udspring("Gyrstinge sø", 250) );
Vandløb vigerdalsÅ = new Vandløb("VigerdalsÅ", 3, 6, new Udspring("Valsøllille sø", 130) );
ringstedÅ.tilføjSidegren( vigerdalsÅ);
```

 Skriv de erklæringer der skal til for Kongskilde bæk, Sorø å, og Suså. Tilføj Kongskilde bæk, Sorø å, og Ringsted å til Suså som sidegrene. (Du må bruge vilkårlige værdier for manglende oplysninger om længde, bredde og liter-per-time i eksemplet.)
4. For at beregne hvor meget vand der løber ud af udmundingen på et vandløb skal vi se hvor meget vand vandløbets egen kilde bidrager med, og så hvor meget der kommer fra hver biflod. Det kan gøres ved metoden double vandMængde() som vist ovenfor.
 - a. Metoden vandMængde kalder igen vandMængde inde i for-løkken. Hvordan kan man være sikker på at metoden ikke bliver ved med at blive kaldt?
 - b. Hvor mange kald vil der ske af metoden vandMængde i eksemplet med Suså, altså ved kaldet Suså.vandMængde()?
5. Vi ønsker at beregne den samlede længde af hele vandløbssystemet, altså af længden af vandløbet plus længden af alle vandløbets sidegrene, plus længden af sidegrenenes sidegrene o.s.v. Skriv en metode double systemLængde() der returnerer den samlede længde af vandløbssystemet.

(Opgave 2 slut.)

Opgave 3 (30%)

Denne opgave omhandler primært brug af tabeller af objekter. Vi betragter en boligblok med lejligheder. Boligblokken er et antal etager høj, og er delt ind i opgange, hvor der i hver opgang kun er en lejlighed per etage. I hver lejlighed bor nul (ubeboet) eller flere personer, og hver person har en række egenskaber – navn, alder og profession. Hver lejlighed kan have en telefon.

Billedet vil højre kunne være en sådan typisk boligblok.



```
public class Person {
    public String navn;
    public int alder;
    public String profession;
    Person (String n, int a, String p){
        navn = n; alder = a; profession = p;
    }
}
public class Lejlighed {
    public String telefonNr;
    public Person[] beboere;
    public Lejlighed(){
        beboere = new Person[0]; telefonNr = null;
    }
    public void flytInd(Person[] b, String tlf){
        beboere = b; telefonNr = tlf;
    }
    public int antalBørn(){
        // her skal din besvarelse af delspørgsmål 5 sættes ind
    }
}
public class BoligBlok
{
    Lejlighed[][] lejligheder;
    BoligBlok(int opgange, int etager){
        // Her skal din besvarelse af delopgave 1 sættes ind
    }
    public Lejlighed getLejlighed(int opgang, int etage){
        // her skal din besvarelse af delopgave 2 sættes ind
    }
    public void flytInd(Person[] personer, String tlf, int opgang, int etage){
        Lejlighed lejlighed = getLejlighed(opgang, etage);
        lejlighed.flytInd(personer, tlf);
    }

    public int antalPersoner(){
        // Her skal din besvarelse af delopgave 3 sættes ind
    }
}

// programkoden fortsætter på næste side...
```

```
// fortsat fra forrige side...

public Lejlighed flest(){
    // Her skal din besvarelse af delopgave 4 sættes ind
}

public int antalBørn(){
    // Her skal din besvarelse af delopgave 5 sættes ind
}
}
```

1. Skriv konstruktøren til klassen Boligblok. Det er vigtigt at tabellen lejligheder bliver initialiseret korrekt med *tomme* lejligheder, og ikke med null værdier. Efter konstruktion af en boligblok, bliver den befolket ved invokeringer af metoden flytInd. Det er altså *ikke* del af konstruktøren at sørge for, at lejligheder er beboet når de initialiseres. F.eks. vil følgende kode skabe en boligblok i 7 etager med 20 opgange, hvorefter familien Hansen flytter ind i opgang 3, på 4 etage.


```
BoligBlok betonBo = new BoligBlok(20, 7);
Person[] familienHansen = { new Person("Anna Hansen", 32, "Smed"),
                             new Person("Jonas Hansen", 35, "Portør"),
                             new Person("Amalie Hansen", 4, "Forældre plager") },
betonBo.flytInd(familienHansen, "12 13 14 15", 3, 4);
```
2. Udfyld kroppen på metoden getLejlighed. Det skal være sådan at opgange starter fra nummer 1, og etager starter fra 1.
3. Udfyld kroppen på metoden antalPersoner. Den skal returnere antallet af personer i hele boligblokken. Antal personer i en lejlighed kan findes ved beboere.length, idet beboere er en tabel som indeholder netop det antal personer der er i lejligheden.
4. Udfyld kroppen på metoden flest. Metoden skal returnere en reference til den lejlighed i boligblokken hvori der bor flest. Hvis der er flere lejligheder med samme maksimale antal beboere er det ligegyldigt hvilken af disse lejligheder den returnerer.
5. Et barn er en person under 12 år. Skriv metoden antalBørn i klassen BoligBlok. Opgaven løses ved først at udfylde kroppen i antalBørn i klassen Lejlighed, sådan at den dernæst kan bruges i metoden antalBørn i klassen BoligBlok, som du også skal udfylde kroppen af.

(Opgave 3 slut.)

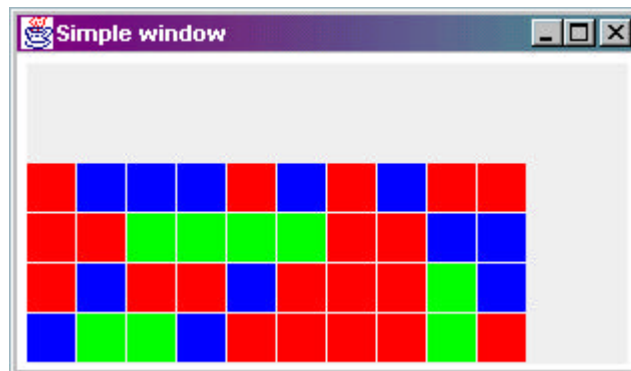
Opgave 4 (15%)

Denne opgave omhandler design af funktionel afprøvning (delopgave 1) samt grafik og grafiske brugergrænseflader. Der tages udgangspunkt i boligblok-eksemplet fra opgave 3, men opgaven kan løses uafhængigt af løsningen til opgave 3.

1. Design en funktionel (ekstern) afprøvning af metoden `flest` i klassen `Boligblok`. Bemærk at `flest` ikke har nogen parametre. Du kan derfor kun variere i testen de boligblokke, `flest` bliver invokeret på, specifikt fordelingen af beboere i lejlighederne, samt boligblokkenes størrelse. En test kunne f.eks. være om `flest` håndterer en 3 etagers 2 opgange boligblok med hhv. 0,1,2,3,4,5 beboere i lejlighederne korrekt.
2. Denne opgave går ud på at lave en del af en brugergrænseflade som viser lejlighederne i en boligblok grafisk. Konkret skal du skrive metoden `tegnBoligblok` (se metodehovedet nedenunder) som vil blive kaldt som en del af denne brugergrænseflade. Boligblokken skal tegnes op i et grafikobjekt (*graphics objekt*) gr der gives som parameter til `tegnBoligblok`. Der skal tegnes en firkant på 25*25 pixels for hver lejlighed. Lejligheder med 0 beboere tegnes grønne, lejligheder med 1 og 2 beboere tegnes blå, mens lejligheder med flere end 2 beboere tegnes røde. Du må antage at grafikobjektet gr er 300 pixels bredt og 150 pixels højt, og at du må tegne helt ud til kanten. Opgaven går derfor ud på at udfylde kroppen af følgende metode:

```
public void tegnBoligblok(Boligblok blok, Graphics gr){
    // her skal din besvarelse sættes ind
}
```

F.eks. vil en boligblok med 10 opgange i 4 etager blive tegnet som vist nedenfor, idet vi forestiller os boligblokken set fra siden.



3. Brugergrænsefladen er programmeret således – koden er ikke vist her – at hver gang man klikker på tegningen med boligblokken omregnes musekoordinaterne til opgang og etage og metoden `lejlighed_clicked` bliver invokeret en reference til den givne boligblok, samt opgangsnummer og etagenummer som argumenter. Du skal i denne opgave erklære en label der indeholder teksten ”Telefon nummer :” og et tekstfelt med plads til 8 tegn, og skrive metodekroppen til `lejlighed_clicked` så telefonnummeret på en lejlighed skrives i tekstfeltet når der klikkes i den. Opgaven går derfor ud på at udfylde kroppen af følgende metode:

```
public void lejlighed_clicked(BoligBlok blok, int opgang, int etage){
    // her skal din besvarelse sættes ind
}
```

Du behøver *ikke* at overveje layout for de grafiske komponenter (boligblokken, label og tekstfeltet).

(Opgave 4 slut.)