Roskilde Universitetscenter
Datalogi, eksamen modul 1, sommer 1999

Skriftlig eksamen i Datalogi

Modul 1

Sommer 1999

Opgavesættet består af 5 opgaver, der ved bedømmelsen tillægges følgende vægte:

Opgave 1	15%
Opgave 2	15%
Opgave 3	8%
Opgave 4	37%
Opgave 5	25%

Alle sædvanlige hjælpemidler er tilladt. I tilfælde af unøjagtigheder i opgaveteksterne forventes det, at deltagerne selv præciserer besvarelsernes forudsætninger.

Opgavesættet består af en forside og 7 paginerede sider. Kontroller at din kopi er fuldstændig.

Opgave 1: Træer (15%)

Nedenfor er en binær hob repræsenteret ved hjælp af et array.

T	S	R	M	O	G	I	A	H	L

Spørgsmål 1.1 Tegn hoben som et binært træ.

Lad der være givet følgende klasseerklæring

```
class Node {
    char key;
    Node left, right;

    Node(char k, Node l, Node r) {
        key = k; left = l; right = r;
    }
}
```

Klassen kan benyttes til at repræsentere en binær hob ved hjælp af et binært træ.

Spørgsmål 1.2 Programmér en kodestump, der opretter et binært træ svarende til hoben ovenfor. Lad `root` betegne træets rod.

Spørgsmål 1.3 Antag at en binær hob med N nøgler er repræsenteret i array-elementerne `a[1]`, `a[2]`, …, `a[N]`. Programmér en generel metode

```
Node makeTree(char a[], int N)
```

som opretter det binære træ, der svarer til hoben. Metoden skal returnere roden i træet.

Vink: Benyt en rekursiv hjælpemetode

```
Node makeSubtree(char a[], int i, int N)
```

der returnerer roden i det binære træ, der har `a[i]` som nøgle i roden.

Opgave 2: Stakke og køer (15%)

Nedenfor ses et udkast til en Java-klasse, `Stack`, der implementerer en stak af objekter.

```
class Stack {
    public void push(Object item) { ... }
    public Object pop() { ... }
    public boolean isEmpty() { ... }
}
```

Klassen har tre offentlige metoder.

`push(item)` lægger objektet `item` øverst på stakken
`pop()` fjerner og returnerer stakkens øverste element
`isEmpty()` returner `true`, hvis stakken er tom; ellers `false`

Lad der være givet en stak `s`.

Spørgsmål 2.1 Skriv en kodestump, der ved hjælp af klassen `Stack` vender objekternes rækkefølge i `s`. Vink: Benyt en eller flere hjælpestakke.

Spørgsmål 2.2 Skriv en kodestump, der ved hjælp af klassen `Stack` danner en kopi `t` af `s`. Efter udførelse af kodestumpen skal indholdet af `s` være som før udførelsen.

Nedenfor ses et udkast til en Java-klasse, `Queue`, der implementerer en kø af objekter.

```
class Queue {
    public void insert(Object item) { ... }
    public Object remove() { ... }
    public boolean isEmpty() { ... }
}
```

Klassen har tre offentlige metoder.

`insert(item)` sætter objektet `item` bagerst i køen
`remove()` fjerner og returnerer køens forreste element
`isEmpty()` returner `true`, hvis køen er tom; ellers `false`

Lad der være givet en kø `q`.

Spørgsmål 2.3 Skriv en kodestump, der ved hjælp af klassen `Queue` danner en kopi `r` af `q`. Efter udførelse af kodestumpen skal indholdet af `q` være som før udførelsen.

Opgave 3: Rekursion (8%)

Lader der være givet følgende metode

```
int f(int a, int b) {
    return a >= b ? a+b : f(f(a+2,b-1),f(a+1,b-2));
}
```

Spørgsmål 3.1: Hvad bliver værdien af `f(1,7)`? Begrund svaret.

Opgave 4: Objektorienteret programmering (37%)

I Studienævnets database over studieaktiviteter registreres kurser og projekter.
Et Kursus eller Projekt skal som minimum være knyttet til et modul.
Vi bruger derfor følgende udgangspunkt for et klassehierarki.

```
abstract class Studieaktivitet{}

class Kursus extends Studieaktivitet{
  String modul;
}

class Projekt extends Studieaktivitet{
  String modul;
}
```

Spørgsmål 4.1: Skriv konstruktorer for klasserne Kursus og Projekt således at feltet "modul" initialiseres.

Et objekt kan så f.eks. oprettes med linien:
Studieaktivitet s = new Kursus("Modul 1");

Spørgsmål 4.2: Tilføj til klasserne Kursus og Projekt ekstra konstruktorer som kaldes med modulnummer som heltal.

Et objekt kan så f.eks. oprettes med linien:
Studieaktivitet s = new Kursus(1);

Spørgsmål 4.3: Hvorledes kan feltet "modul" beskyttes så det ikke kan ændres uden for den klasse hvori den er erklæret?

Spørgsmål 4.4: Tilføj metoder "hentmodul" til klasserne Kursus og Projekt. Metoderne skal returnere værdien af feltet "modul".

Spørgsmål 4.5:
Alle studieaktiviteter skal være knyttet til et modul. Gør det derfor muligt at kalde en metode "hentmodul" i vilkårlige objekter nedarvet fra den abstrakte klasse "Studieaktivitet".

F.eks. med erklæringen Studieaktivitet s = new Kursus(1);
skal kaldet s.hentmodul() være tilladt.

Spørgsmål 4.6: Projekter skal have en vejleder. Der skal derfor være et felt "vejleder" i klassen "Projekt". Skriv erklæring af feltet samt metoder der kan sætte feltet og returnere dets værdi.

Spørgsmål 4.7: Studienævnet ser ikke gerne at man skifter vejleder. Forslå derfor en måde at sikre at feltet ikke kan ændres når det først er sat. Yderligere kald til metoden, der skal sætte dets værdi, skal derfor ikke have nogen effekt.

Opgave 5: OOA (25%)

Forestil dig at du skal lave et system til et autoværksted, der reparerer personbiler, lastvogne og motorcykler. Autoværkstedet er organiseret med et kontor, hvor der arbejder tre kontormedarbejdere, og et værksted, hvor ti mekanikere arbejder.

Autoværkstedet tilbyder den service, at indkalde køretøjerne til eftersyn. For hver af de tre køretøjstyper er der forskellige intervaller for indkaldelse og forskellige opgaver, der skal udføres.

Hver mekaniker har som speciale at reparere en eller flere af de tre køretøjstyper.

Når en kunde bestiller tid til en reparation, skal en kontormedarbejder kunne gøre følgende: Hvis det er en ny kunde, skal firmaet have kundens stamdata (navn, adresse, telefonnummer) samt vide hvilken slags køretøj der er tale om og dets registreringsnummer. Hvis det er en gammel kunde, skal det undersøges, om de oplysninger, værkstedet allerede har på kunden, stadig er gældende.

Når køretøjet er afleveret, skal kunden kunne få oplysninger om, hvor langt det er i processen og om det forventede afleveringstidspunkt. Erfaringsmæssigt har det vist sig at være en god ide, at kunne spørge kunden, hvis det under reparationen viser sig, at der er tale om en større skade end først antaget, eller hvis det viser sig, at være en god ide at reparere noget, der ikke var aftalt på forhånd. Kunden har så mulighed for at sige fra eller til i forhold til de yderligere reparationer. Kunden skal desuden på et hvilket som helst tidspunkt i processen kunne forlange reparationen stoppet.

Systemet skal fra starten endvidere bruges til at holde styr på hvilken mekaniker, der har repareret hvilke køretøjer. Og mekanikeren skal kunne rapportere hvilke reparationer han har foretaget.

Din opgave er nu at gennemføre en analyse ved hjælp af OOA, og vi forestiller os, at du sammen med ejeren er kommet frem til følgende systemdefinition udtrykt i BATOFF:

Betingelser: Edb-systemet skal benyttes både på kontoret og på værkstedet, og det skal udvikles i samarbejde med ejeren, en kontormedarbejder og en mekaniker. Skulle der opstå modstridende krav til systemet, er det ejeren der bestemmer.

Anvendelsesområde: Administration af tilmelding og reparation af køretøjer.

Teknologi: Det skal være tre PC'er på kontoret, en til ejeren og en på værkstedet. De skal forbindes i et lokalnetværk.

Objektsystem: Kunde, køretøj, mekaniker.

Funktionalitet: Støtte til kontorarbejdet og til rapportering af foretagne reparationer. På sigt skal det også bruges til at fordele køretøjerne på mekanikerne, i forhold til deres speciale og i forhold til en effektiv udnyttelse af den samlede gruppe af mekanikere.

Filosofi: Et værktøj til administration samt til støtte for kommunikation med kunderne.

Spørgsmål 5.1: Lav og begrund et klassediagram for objektsystemet.

Spørgsmål 5.2: Lav og begrund et tilstandsdiagram for køretøj.

NB! Hvis du foretager afgrænsninger eller gør dig yderligere forudsætninger end de beskrevne, skal du gøre rede for disse i besvarelsen.