

Skriftlig eksamen i grundlæggende programmering

IT-C, 12 juni, 2002

Dansk udgave

Alle hjælpemidler er tilladt, undtaget computere og andet elektronisk udstyr.

Eksamensspørgsmålene skal besvares inden for fire timer. Besvarelsen vil blive bedømt efter 13 skalaen.

Der er 4 overordnede spørgsmål som hver har lige vægt.

Din besvarelse kan inddrage klasser og metoder fra lærebogen, og andet kursus materiale. I så fald behøver du ikke at kopiere dem, men kan give en præcis reference der præciserer sidetal ,sektions nummer etc.

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

Det anbefales at læse alle spørgsmål igennem ved eksamens start.

Spørgsmål 1

Betragt følgende definition af klassen Loop:

```
class Loop {
    public static void loop(int x, int y, int z) {
        for (int c = x; c > y; c = c-z)
            System.out.println("c: " + c + " x: " + x + " y: " + y + " z: " + z);
    }
    public static void nestedLoop(int x, int y, int z) {
        while (x >= y)
            loop(x--, ++y, z++);
    }
}
```

Spørgsmål 1.1

Hvad udskrives på skærmen ved metodekaldet `Loop.loop(9, 3, 2)`?

Spørgsmål 1.2

Hvad udskrives på skærmen ved metodekaldet `Loop.nestedLoop(6, 0, 2)`?

Spørgsmål 1.3

Skriv kroppen på metoden `classify` nedenfor. Metoden tager en 2-dimensional tabel (array af arrays) af heltal som parameter, og udskriver antallet af nuller (0), ettaller (1) og andre (ikke 0 eller 1) som forekommer i tabellen.

```
public class ZeroesOnes {
    public static void classify(int[][] arr) {
        // her skal din kode indsættes.
    }
    public static void test() {
        int[][] a = {{0, 1, 8, 9}, {4, 4, 1, 0}, {1, 1, 0}};
        classify(a);
    }
}
```

Som eksempel skal resultatet af at udføre metoden `ZeroesOnes.test()` være:

```
number of 0: 3
number of 1: 4
number of other: 4
```

Spørgsmål 2

Betragt følgende ukomplette definition af klassen Hat.

```
class Hat {
    protected char printSymbol;
    public Hat(char printSymbol) {
        this.printSymbol = printSymbol;
    }
    // prints a number of characters in a row
    private void draw(char symbol, int times) {
        for (int i = 0; i < times; i++)
            System.out.print(symbol);
    }

    // prints a new line
    private void draw() {
        System.out.println();
    }
    public void print() {
        ... // see Spørgsmål 2.1
    }
    public void print(int topWidth, int bottomWidth, int height) {
        ... // see Spørgsmål 2.2
    }
    public boolean valid(int topWidth, int bottomWidth, int height) {
        ... // see Spørgsmål 2.3
    }
    public void printValid(int topWidth, int bottomWidth, int height) throws Exception {
        if (valid(topWidth, bottomWidth, height))
            print(topWidth, bottomWidth, height);
        else
            throw new Exception("Invalid input arguments");
    }
}
```

Spørgsmål 2.1

Skriv definitionen af metoden print med signaturen print(). Når metoden kaldes skal den udskrive nedenstående figur på skærmen. Figuren forestiller en hat tegnet med symbolet der er værdien af variabelen printSymbol. Brug de 2 overloaded private metoder draw i klassen Hat til at for at tegne en række symboler og til at skifte linje med. For eksempel vil følgende kodefragment

```
Hat hat0 = new Hat('*');
hat0.print();
```

udskrive

```
*****
*   *
*   *
*   *
*****
```

på skærmen. Bemærk at denne "hat" er præcis 5 stjerner bred i toppen, 11 stjerner bred i bunden, og 5 stjerner høj.

Vink : brug f.eks. metode kaldet draw(' ', 3) til at tegne 3 blanke på række. Din metode behøver kun at kunne tegne hatte med ovennævnte facon, men skal bruge værdien af printSymbol.

Spørgsmål 2.2

Skriv definitionen af kroppen til metoden print med signaturen print(int, int, int), hvor parametrene angiver hvor bred hatten skal være i toppen, hvor bred den skal være i bunden, og hvor høj den skal være.

F.eks. skal følgende kode fragment

```
Hat hat1 = new Hat('+');
hat1.print(5, 11, 5);
System.out.println();
hat1.print(4, 6, 6);
```

udskrive

```
+++++
+  +
+  +
+  +
+++++

++++
+  +
+  +
+  +
+  +
+++++
```

på skærmen. I dette spørgsmål behøver du ikke at teste på parametrene der angiver øverste og nederste bredde samt højde, men kan antage at de er meningsfulde.

Spørgsmål 2.3

Skriv kroppen af metoden valid, som checker om argumenterne der skal gives til print metoden er lovlige. For at være lovlige skal argumenterne opfylde følgende krav.

1. Hatten skal have mindst et mellemrum i midten, det vil sige at den skal mindst 3 symboler bred i toppen, og mindst 3 symboler høj.
2. Bredden i bunden skal være mindst 2 symboler bredere end bredden i toppen.
3. Hatten skal være symmetrisk, kanten for neden skal altså stikke lige meget ud i højre og i venstre side.

Det betyder f.eks. at de følgende kald af metoden valid skal returnere falsk:

1. valid(4, 5, 4), fordi bunden ikke er tilpas bredere end toppen.
2. valid(2, 8, 7), fordi toppen ikke er bred nok.
3. valid(4, 6, 2), fordi hatten ikke er bred nok.
4. valid(5, 10, 7), fordi hatten ikke kan tegnes symmetrisk.

På den anden side, så skal følgende kald af valid returnere sand:

1. valid(4, 6, 4),
2. valid(3, 9, 7),
3. valid(4, 6, 3),
4. valid(5, 11, 7).

Spørgsmål 2.4

Betragt følgende definition af klassen SubHat:

```
class SubHat extends Hat {
    public SubHat(char printSymbol) {
        super(printSymbol);
    }
    public void draw(char symbol, int times) {
        System.out.println('o');
    }
    public boolean valid(int t, int b, int h) {
        return true;
    }
}
```

Hvad udskriver følgende kode fragment på skærmen:

```
Hat hat2 = new SubHat('x');
try {
    hat2.printValid(2, 2, 2);
} catch (Exception e) {
    System.out.println("An exception has been thrown.");
}
```

Giv en kort forklaring på den udførsel der leder til dette resultat.

Spørgsmål 3

Dette spørgsmål drejer sig om at forstå en klasse (spørgsmål 3.1 og 3.2) og siden udvide den (spørgsmål 3.3 og 3.4).

Betragt følgende definition af klassen SimpleVector

```
class SimpleVector {
    protected Object[] arr;
    protected int length;
    public SimpleVector(int size) {
        arr = new Object[size];
        length = 0;
    }

    public Object get(int index) throws Exception {
        if (0 <= index && index < length)
            return arr[index];
        else
            raise Exception("Illegal index argument to get method");
    }

    public void set(int index, Object obj) {
        arr[index] = obj;
        if (index >= length) length = index + 1;
    }

    public int length() {
        return length;
    }

    public void print() {
        for (int i=0; i < length; i++)
            System.out.println(arr[i]);
    }
}
```

Bemærk at længden af `new SimpleVector(17)` (eller et vilkårligt andet argument til konstruktøren) initialt er 0 (nul).

Spørgsmål 3.1

Hvad udskriver udførelsen af følgende kode fragment på skærmen?

```
SimpleVector a0 = new SimpleVector(4);
a0.set(0, "Freddy");
a0.set(2, "George");
a0.set(1, "Madonna");
try {
    a0.print();
} catch (Exception e) {
    System.out.println(e.getMessage());
}
```

Spørgsmål 3.2

Nedenstående kode fragment udskriver “r”, det fjerde bogstav i tekst strengen “George”. Hvad vil ske hvis vi udelader downcast operationen (String) i tredje linje?

```
SimpleVector a1 = new SimpleVector(5);
a1.set(2, "George");
String name = (String) a1.get(2); // contains downcast operation
System.out.println(name.charAt(3));
```

Spørgsmål 3.3

Skriv klassen FlexibleVector som en subklasse af SimpleVector, så FlexibleVector har en metode add som skal indsætte et element i det næste ledige felt i et vector objekt. For eksempel, udførelsen af

```
FlexibleVector a2 = new FlexibleVector(10);
a2.add("Freddy");
a2.add("Madonna");
a2.add("George");
System.out.println("a2 has " + a2.length() + " elements. They are:");
a2.print();
```

skal udskrive:

```
a2 has 3 elements. They are:
Freddy
Madonna
George
```

Klassen FlexibleVector skal, ligesom sin superklasse, have en konstruktør der som argument tager det maximale antal elementer der kan lagres .

Spørgsmål 3.4

(Dette er en fortsættelse af spørgsmål 3.3. Det anbefales *ikke* at starte på denne opgave før *alle* andre opgaver er besvaret.)

Skriv en klasse UnboundedFlexibleVector som subklasse af SimpleVector. UnboundedFlexibleVector skal opføre sig som FlexibleVector, men skal ikke have nogen restriktion på hvor mange elementer der kan indsættes i et vektor objekt.

Spørgsmål 4

Dette spørgsmål drejer sig om ansatte i et firma. En ansat har et navn og en unikt identifikationsnummer. En ansat er repræsenteret ved klassen Employee nedenfor.

```
class Employee {
    private String name;
    private static int nextId = 1;
    private int id;
    public Employee(String name) {
        this.name = name;
        this.id = nextId++;
    }
    public String getName() {
        return name;
    }
    public int getId() {
        return id;
    }
}
```

Spørgsmål 4.1

Klassen Employee kunne alternativt være skrevet som den meget kortere klasse Employee2 nedenfor

```
class Employee2 {
    public String name;
    public static int nextId = 1;
    public int id;
}
```

Spørgsmål 4.1.1

Employee2 har ikke tilgangsmetoderne getName og getId, og heller ingen konstruktør. Vis hvordan det følgende Java fragment skal skrives hvis man skal bruge Employee2 i stedet for Employee. Altså, skriv Java kode der først skaber en instans af klassen Employee2, og dernæst sætter den ansattes navn og identifikationsnummer.

```
Employee henrik = new Employee("Henrik");
System.out.println("Employee: " + henrik.getName() + ", " + henrik.getId());
```

Spørgsmål 4.1.2

Giv en begrundelse/motiv for at vælge Employee fremfor Employee2 til at repræsentere ansatte med, på trods af at Employee involverer mere kode.

Spørgsmål 4.2

En "ledet ansat" (managed employee) er en ansat der har en leder. Lederen er også en ledet ansat.

Spørgsmål 4.2.1

Skriv ManagedEmployee som en subklasse af Employee sådan at

- ?? subklassen har en ny privat instansvariabel manager af typen ManagedEmployee
- ?? subklassen har tilgangs metoder setManager og getManager til at tilgå værdien af manager. Metoden getManager skal returnere null for nye instanser af ManagedEmployee (altså før der er lavet et kald af setManager).

Spørgsmål 4.2.2

Skriv en række Java sætninger der skaber ansatte "Fritz", "Henrik" and "Mads" (i den rækkefølge), og så gør Mads til leder for Henrik, og Henrik til leder for Fritz.

Spørgsmål 4.3

Lad ManagedEmployee være den klasse der blev defineret i spørgsmål 4.2. (du behøver ikke at have løst 4.2 for at løse denne opgave). Givet en ledet ansat employee, direktøren for employee er defineret som:

- ?? hvis employee.getManager() returnerer null (null referencen), så er employee direktør for sig selv.
- ?? ellers er employee's direktør den der er direktør for employee's leder.

Spørgsmål 4.3.1

Tilføj en metode getDirector() til klassen ManagedEmployee. Metoden skal returnere hvem der er direktør for en ledet ansat.

Spørgsmål 4.3.2

Forklar hvad der sker når den følgende kode bliver udført (med din definition af metoden getDirector).

```
ManagedEmployee hans = new Employee("Hans");
ManagedEmployee per = new Employee("Per");
hans.setManager(per);
per.setManager(hans);
System.out.println("Hans's director: " + hans.getDirector());
```

Spørgsmål 4.4

Spørgsmål 4.4.1

Vi siger at en ansat emp (kort for employee) er en underordnet (subordinate) af emp2, hvis emp2 er emp's leder.

Definer kroppen på klassemetoden `printNumSubordinates` (udskriv antal underordnede) nedenfor:

```
public class ManagedEmployee {
    public static void printNumSubordinates(ManagedEmployee[] emps) {
        ...
    }
    ... andre metoder og felter i klassen fra opgave 4.2 og 4.3
}
```

Metoden skal udskrive hvor mange underordnede hver ledet ansat i tabellen emps har, med en linje per ledet ansat. Du kan antage at alle ansatte har et unikt id mellem 1 og 10000, og du kan bruge dette id (som stammer fra opgave 4.1) til at identificere en ansat i output. Her er et eksempel på et output fra et kald af `printNumSubordinates` med en tabel med 10000 elementer.

```
Employee 1 has 0 subordinates.
Employee 2 has 4 subordinates.
Employee 3 has 1 subordinates.
Employee 4 has 2 subordinates.
Employee 5 has 0 subordinates.
...
Employee 10000 has 0 subordinates.
```

Bemærk, I eksemplet har vi udeladt 9994 linjer (angivet med ...) – din kode skal producere alle linjer.

Hint: Brug en tabel `subords` af længde 10001 sådan at `subords[i]` bruges til at tælle hvor mange underordnede den ansatte med id i har.

Spørgsmål 4.4.2

Vi ønsker at metoden `printNumSubordinates` fra spørgsmål 4.4.1 skal udskrive linjerne i sorteret rækkefølge, med den ledede ansatte med flest underordnede først. F.eks.

```
Employee 6433 has 87 subordinates.
Employee 127 has 55 subordinates.
Employee 14 has 37 subordinates.
Employee 8814 has 22 subordinates.
```

og så videre (10000 linjer i alt).

Beskriv og motiver hvilken sorterings algoritme du vil bruge. Der ønskes ikke Java kode i denne besvarelse.