

# Grundlæggende programmering

8 sider

4 timers skriftlig prøve den 11. januar 2003

Alle skriftlige hjælpemidler er tilladte til denne eksamen, men ikke elektroniske.

Besvarelsen bedømmes efter 13-skalaen, vægten af de enkelte opgaver og spørgsmål er angivet i parentes i overskriften.

Det tilrådes, at du læser hele opgavesættet igennem inden du begynder løsningen af det.

<b>1for-sætningen (5%)</b> .....	<b>2</b>
1.1Hvad er uddata? (5%) .....	2
<b>2switch-sætningen (5%)</b> .....	<b>2</b>
2.1Hvad er uddata? (5%).....	2
<b>3Metoder og statiske felter (15%)</b> .....	<b>3</b>
3.1Skriv konstruktøren (8%).....	3
3.2Skriv metoden printList (7%).....	3
<b>4Referencer (10%)</b> .....	<b>4</b>
4.1Hvad er uddata? (10%).....	4
<b>5Hændelser og GUI (15%)</b> .....	<b>5</b>
5.1Tilføj en knap (5%).....	5
5.2Gør knappen interaktiv (10%).....	5
<b>6Medlemsadministrationssystemet (25%)</b> .....	<b>6</b>
6.1Skriv klassen Member (5%).....	6
6.2Skriv klassen Team (10%).....	6
6.3En anden repræsentation af Team (10%).....	7
<b>7PC-oversigten (25%)</b> .....	<b>7</b>
7.1Udarbejd og vis en klassestruktur for programmet (9%).....	7
7.2Udarbejd en udskrift for en enkelt PC'erne (8%).....	7
7.3Klassen PCList(8%).....	8

I opgavesættet er det brugt følgende notation og formulering:

ÿ "?" repræsenterer kode, der ikke er vist – du kan bruge samme notation i din bevarelse

ÿ "udskrivning på konsollen" betyder udskrift ved hjælp af `print-` og `println-` metoderne fra `System.out`-objektet

## 1 for-sætningen (5%)

En metode `print()` ser således ud:

```
public void print(int max) {
    char c;
    for (int i = 0; i < max; i++) {
        for (int j = 0; j < max; j++) {
            if (j < i) c = ' '; else c = '*';
            System.out.print(c);
        }
        System.out.println();
    }
}
```

### 1.1 Hvad er uddata? (5%)

Hvad udskriver metoden på konsollen, hvis den kaldes med `print(4)`;

## 2 switch-sætningen (5%)

Metoderne `printGrade` og `level` ser således ud:

```
public void printGrade (String name, int grade) {
    System.out.println
        (name + ": " + grade + " [" + level(grade) + "]);
}

public String level(int grade) {
    String s = "";
    switch (grade) {
        case 0:
        case 3:
        case 5: s = "not passed"; break;
        case 6: s = "just ";
        case 7:
        case 8:
        case 9: s += "passed"; break;
        case 10:
        case 11:
        case 13: s = "well passed"; break;
        default: return "not a grade";
    }
    return s;
}
```

### 2.1 Hvad er uddata? (5%)

Hvad er uddata til følgende kald af metoden `printGrade`?

```
printGrade("Peter", 8);
printGrade("Paul", 3);
printGrade("Mary", 1);
printGrade("Joe", 6);
```

### 3 Metoder og statiske felter (15%)

En klasse `Car` er defineret således:

```
class Car {  
  
    private static final int MAX_CARS = 100;  
    private static int noOfCars = 0;  
    private static Car[] car = new Car[MAX_CARS];  
    private int carNo;  
    private String cartype;  
  
    public Car(String cartype) {  
        ?  
    }  
  
    public static void clear() {  
        noOfCars = 0;  
    }  
  
    public String toString() {  
        ?  
    }  
  
    public static void printList() {  
        ?  
    }  
}
```

Hvis man fra en metode uden for klassen `Car` udfører følgende kode

```
Car.clear();  
new Car("VW Passat");  
new Car("Volvo Amazon");  
new Car("Ford Galaxy");  
new Car("Peugot 505");  
Car.printList();
```

vil det resultere i dette uddata:

```
Car no 0: VW Passat  
Car no 1: Volvo Amazon  
Car no 2: Ford Galaxy  
Car no 3: Peugeot 505
```

#### 3.1 Skriv konstruktøren (8%)

Skriv konstruktøren til klassen `Car`, således at ethvert af objekt af klassen gemmes i det statiske array `car`. Feltet `noOfCars` skal til hver en tid indeholde antallet af `car`-objekter i array'et `car`. Det kan antages, at værdien af `noOfCars` aldrig overskrider værdien af `MAX_CARS`.

#### 3.2 Skriv metoden `printList` (7%)

Skriv metoden `printList`, der gennemløber hele array'et `car` og udskriver en liste som den viste.

## 4 Referencer (10%)

Klassen Q4 er erklæret på følgende måde:

```
class Q4 {  
  
    private String id ;  
  
    public Q4(String id) {  
        this.id = id;  
    }  
  
    public void setId(String id) {  
        this.id = id;  
    }  
  
    public String getId() {  
        return id;  
    }  
  
    public String toString() {  
        return "Id = " + id;  
    }  
}
```

### 4.1 Hvad er uddata? (10%)

Hvad skrives der på konsollen, hvis man kører følgende kode:

```
Q4 q1 = new Q4("John");  
Q4 q2 = new Q4("Bob");  
Q4 q3 = new Q4("Ed");  
  
System.out.println("out1: " +q1 + ", " + q2 + ", " + q3);  
q2 = q1;  
q1.setId("Jackie");  
q3.setId(q2.getId());  
q1 = new Q4("Bill");  
System.out.println("out2: " + q1 + ", " + q2 + ", " + q3);
```

## 5 Hændelser og GUI (15%)

Klasserne CounterWindow og CenterPanel ser således ud:

```
import javax.swing.*;
import java.awt.*;

public class CounterWindow extends JFrame {

    private Container pane;
    private CenterPanel centerPanel = new CenterPanel(3);

    public CounterWindow() {
        super("Vindue 1");
        pane = getContentPane();

        pane.add(centerPanel, "Center");
        setBounds(100, 50, 200, 100);
        setVisible(true);
    }
}

public class CenterPanel extends JPanel {

    private int value = 0;

    public CenterPanel(int start) {
        value = start;
    }

    public void inc() {
        value++;
    }

    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        int w = getWidth(); // gets the JPanels width
        int h = getHeight(); // gets the JPanels height
        g.drawString(""+value, w/2, h/2);
    }
}
```

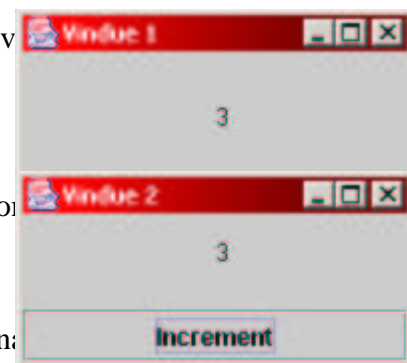
Med kaldet `new CounterWindow` får man et vindue som vist ovenfor, kaldet *Vindue 1*.

### 5.1 Tilføj en knap (5%)

Tilføj en knap til *Vindue 1* så vinduet kommer til at se ud omtrent som vist ovenfor.

### 5.2 Gør knappen interaktiv (10%)

Gør knappen interaktiv, således at hver gang brugeren klikker på knappen kommer tallet i vinduet over knappen til at være én højere end tidligere, fx fra 3 til 4.



## 6 Medlemsadministrationssystemet (25%)

Et system til medlemsadministration i en fodboldklub har blandt andet følgende klasser:

```
class Member {  
  
    private String name;  
    private short birthYear;  
    private Team team;  
  
    public Member(?) {  
        ?  
    }  
  
    public void setTeam(?) {  
        ?  
    }  
  
}  
  
class Team {  
    ?  
  
    public void addMember(Member member) {  
        ?  
    }  
  
    public void printTeamList(?) {  
        ?  
    }  
  
}
```

### 6.1 Skriv klassen *Member* (5%)

Skriv til klassen `Member` parameterlisten og kroppen til

- konstruktøren, sådan at alle felter bliver initialiseret (dette indebærer, at medlemmet skal have et navn og et fødselsår, men ikke at det er tilknyttet noget hold)
- metoden `setTeam`, sådan at medlemmet bliver tilknyttet et hold. Relationen mellem medlem og hold skal være tovejs, så metoden skal kalde holdets `addMember`-metode. Du må gerne antage at medlemmet ikke i forvejen er på et team.

### 6.2 Skriv klassen *Team* (10 %)

Skriv klassen `Team`, herunder

- erklæringerne til felterne, sådan at referencerne til holdets medlemmer indeholdes i et array (der er højst 18 medlemmer på et hold)
- konstruktøren, hvis den er nødvendig
- metoden `addMember`
- metoden `printTeamList`

### 6.3 En anden repræsentation af Team (10%)

Genskriv klassen **Team**, således at et hold bliver repræsenteret ved en *kædet liste* (det samme som en *hægtet liste* eller en *linked list*) i stedet for et array. Genskriv om nødvendigt også klassen **Member**.

## 7 PC-oversigten (25%)

I en virksomhed er der brug for en oversigt over virksomhedens PC'ere. Disse findes i tre *varianter*, nemlig som bærbare, som stationære og som servere.

Hver PC har en navn, en pris og et anskaffelsesår. Herudover kan stationære og bærbare PC'ere være tilnyttet en medarbejder, nogle PC'ere er dog fast monteret et sted, fx i receptionen, i et kursuslokale eller i demonstrationlokalerne, hvor de så benyttes af forskellige personer. Servere er altid fast monteret i et lokale. Nogle medarbejdere har både en stationær og en bærbar PC.

Systemadministratoren påtænker at udarbejde et program, der kan hjælpe ham med at få overblik over virksomhedens PC'ere. Han har især brug for

- en oversigt over samtlige PC'ere ordnet efter anskaffelsesår
- en oversigt over alle brugere, som viser hvilke PC'ere de har tilknyttet
- en oversigt over alle lokaler, der viser hvilke PC'ere de har fastmonteret.

Til at konstruere sit system har systemadministratoren allerede udarbejdet klasserne **Bruger** og **Lokale** (ikke vist) samt klassen **PCList**, der ser således ud:

```
class PCList {
    //Felter erklæres her
    public PCList() { ? }
    public void add(PC pc) { ? }
    public void printAll() { ? }
}
```

Metoden **Add** vil tilføje en PC til listen, mens metoden **printAll** udskriver oplysninger om de PC'er, der findes i listen.

### 7.1 Udarbejd og vis en klassestruktur for programmet (9%)

Udarbejd en klassestruktur, der som minimum indeholder klasserne **PC**, **Bærbar**, **Stationær**, **Server**, **Bruger**, **Lokale** og **PCList**, hvor

Vis klassestrukturen enten som UML-diagrammer eller som Javakode, der viser klassernes hoved samt deres felter og deres metoders hoveder.

### 7.2 Udarbejd en udskrift for en enkelt PC'erne (8%)

Udarbejd den kode, der gør det muligt at udskrive oplysningerne for én PC med navn, pris, anskaffelsesår og variant, samt oplysning om hvilken bruger eller lokale, PC'en er tilknyttet eller monteret. Tip: brug klassernes **toString**-metoder.

Oversigten skal udskrives på konsollen og kan for en server i lokale 7 fx se således ud:

Server: Server1, pris: 20000, anskaffet: 1999, lokale: 7
--

mens den for en bærbar, som er tilknyttet brugeren Jens fx kan se således ud:

Bærbar: Bærbar1, pris: 25000, anskaffet: 2002, bruger: Jens
---

Hvis samme Jens også har en stationær PC, kunne udskriften for den stationære PC fx se således ud:

Stationær: Stationær1, pris: 15000, anskaffet: 2000, bruger: Jens
---

### 7.3 Klassen *PCList*(8%)

Færdiggør klassen `PCList` ved skrive den manglende kode til felter og metoder.