

Grundlæggende programmering

11 sider i alt

4 timers skriftlig prøve fredag den 9. januar 2004 kl 9:00 – 13:00

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.

1for-sætningen (5%)	2
1.1Hvad udskriver følgende stump kode? (5%).....	2
2Fra do-while til while (5%)	2
2.1Omskriv følgende do-while-løkke til en ækvivalent while-løkke (5%).....	2
3Referencer (5%)	2
3.1Hvad er output? (5%).....	2
4Klassen Billet -- Statiske felter og metoder (15%)	3
4.1Skriv konstruktøren til klassen Billet (10%).....	3
4.2Skriv metoden toString() til klassen Billet (5%).....	4
5Klassen FileUtil -- Exceptions (15%)	5
5.1Skriv metoden check i klassen FileUtil (15%).....	5
6The *+Game -- Hændelser og GUI (15%)	6
6.1Gør klassen interaktiv (15%).....	6
7Bibliotekssystemet (20%) -- arv og polymorfi	7
7.1Skriv klassen Bog (5%).....	7
7.2Skriv klassen CD (10%).....	7
7.3Skriv klassen SerieBog (5%).....	7
8Kunde og sælger (20%) -- datastrukturer	8
8.1Skriv klassen Saelger (10%).....	8
8.2Skriv klassen Kunde (10%).....	8
9Bilag 1: Uddrag af dokumentationen for klassen java.io.FileReader	9
9.1Konstruktøren FileReader.....	9
9.2Metoden read.....	9
10Bilag 2: Kode til EventFrame	10

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%)

1.1 Hvad udskriver følgende stump kode? (5%)

```
char c;
for (int i = 0; i < 4; i++) {
    if (i == 2 || i == 4) c = + ; else c = ;
    System.out.print(c);
}
System.out.println();
```

2 Fra do-while til while (5%)

2.1 Omskriv følgende do-while-løkke til en ækvivalent while-løkke (5%)

```
int i = 0;
do {
    System.out.print(i + " ");
    i++;
} while (i < MAX);
```

3 Referencer (5%)

Betragt følgende stump kode:

```
String a = "Kofoed";
String b = "Olesen";
String c = "Pilmark";
String d = "Reher";

System.out.println
    ("a=" + a + ", b=" + b + ", c=" + c + ", d=" + d);
a = b;
b = "Enevold";
c = null;
d = a;
System.out.println
    ("a=" + a + ", b=" + b + ", c=" + c + ", d=" + d);
```

3.1 Hvad er output? (5%)

Hvad vil ovenstående stump kode udskrive på konsollen?

4 Klassen Billet -- Statiske felter og metoder (15%)

Klassen `Billet` er defineret således:

```
public class Billet {  
  
    public static final boolean UDEN_RABAT = false;  
    public static final boolean MED_RABAT = true;  
    private static final int RABAT_PCT = 15;  
  
    private static int antal = 0;  
    private static int sum = 0;  
    private int nr;  
    private int pris;  
  
    public Billet(int pris, boolean rabat) {  
        ?  
    }  
  
    ? //øvrige metoder  
  
}
```

Uden for klassen findes denne kode:

```
System.out.println(  
    "Antal=" + Billet.getAntal() + ", sum= " + Billet.getSum());  
  
Billet b1 = new Billet(100, Billet.MED_RABAT);  
System.out.println("b1: " + b1.toString());  
  
Billet b2 = new Billet(100, Billet.UDEN_RABAT);  
for (int i = 0; i < 10; i++) {  
    new Billet(100, Billet.UDEN_RABAT);  
}  
System.out.println("b2: " + b2.toString());  
  
System.out.println(  
    "Antal=" + Billet.getAntal() + ", sum= " + Billet.getSum());
```

Når denne kode afvikles vil den resultere i følgende udskrift på konsollen:

```
Antal=0, sum= 0  
b1: Billet nr 1 ud af 1 solgte billetter, pris: 85  
b2: Billet nr 2 ud af 12 solgte billetter, pris: 100  
Antal=12, sum= 1185
```

4.1 Skriv konstruktøren til klassen `Billet` (10%)

Konstruktøren skal indeholde kode, der sikrer,

- at feltet `antal` altid indeholder antallet af hidtil oprettede billetter
- at feltet `pris` udregnes som værdien af parameteren `pris` med fradrag af en rabat svarende til `RABAT_PCT`. Hvis fx `RABAT_PCT` er 15 og parameteren `pris` er 200 skal feltet `pris` sættes til 170. Dette repræsenterer billetens pris.
- at feltet `sum` er summen af prisen på alle hidtil oprettede billetter

4.2 *Skriv metoden toString() til klassen Billet (5%)*

Skriv metoden toString() til klassen Billet, således at en returværdi fx kan se således ud:

`Billet nr 2 ud af 12 solgte billetter, pris: 100`

hvis metoden kaldes på det objekt, der i eksemplet ovenfor referes gennem variabelen `b2`.

5 Klassen FileUtil -- Exceptions (15%)

Klassen FileUtil er defineret således:

```
class FileUtil {  
  
    public static int countChars(String fileName) throws IOException  
    {  
        FileReader fileReader = new FileReader(fileName);  
        int i = fileReader.read();  
        int count = 0;  
  
        while (i != 1) {  
            count++;  
            i = fileReader.read();  
        }  
        return count;  
    }  
  
    public static void check(String fileName) {  
        // denne metode udskriver antallet af tegn i filen "filename"  
        ?  
    }  
}
```

Metoden `check` udskriver på konsollen en meddelelse om den fil, hvis navn er parameter til kaldet:

• hvis filen findes skriver metoden antallet af tegn på konsollen

• hvis filen ikke findes, eller hvis der er andre problemer med åbning og læsning af filen udskriver metoden en meddelelse om dette

Antag at filen `yes.txt` findes og at filen `no.txt` ikke findes. Da vil kodestumpen

```
FileUtil.check("yes.txt");  
FileUtil.check("no.txt");
```

på konsollen udskrive

```
Tjekker yes.txt: 101 tegn  
Tjekker no.txt: findes ikke
```

5.1 Skriv metoden `check` i klassen FileUtil (15%)

Metoden `check` skal ikke selv åbne den fil, der skal undersøges, men benytte sig af at metoden `countChars` kan gøre dette. Denne metode indeholder metodekald til et objekt af klassen `FileReader`, beskrivelsen af disse metode kan du om nødvendigt se i bilag 1.

6 The *+Game -- Hændelser og GUI (15%)

I bilag 2 findes koden til en klasse `EventFrame`, der viser vinduet "The *+game". Dette spil går ud på at få `current`-tallet til at ramme `target`-tallet:

• hvis der klikkes på knappen "`*2 +1`" vil `current`-tallet blive ganget med 2 og få lagt 1 til

• hvis der klikkes der på knappen "`*2 +0`" vil `current`-tallet blot blive ganget med 2

• hvis det klikkes på knappen "reset" vil der blive genereret et nyt `target`-tal mellem 0 og 100 og `current`-tallet bliver sat til 0.

Hvis `current`-tallet rammer eller bliver større end `target`-tallet er spillet slut og det skal ikke reagere på klik på de to venstre knapper.

Spillet er i koden repræsenteret ved klassen `Game`, der ser således ud:

```
public class Game {

    private int current;
    private int target;

    public Game() { reset(); }

    public void muladd() { current = current * 2 + 1; }

    public void mul() { current = current * 2; }

    public void reset() {
        target = (int) (Math.random() * 100);
        current = 0;
    }

    public int getCurrent() { return current; }

    public int getTarget() { return target; }

    public boolean over() {return current >= target; }

    public boolean hit() {return current == target; }

}
```

6.1 Gør klassen interaktiv (15%)

Den kode til `EventFrame`, der er vist i bilag 2, er ikke interaktiv, det vil sige at den ikke reagerer på klik på tasterne. Til at gøre den interaktiv får den til at reagere på klik som beskrevet ovenfor.



7 Bibliotekssystemet (20%) -- arv og polymorfi

Et bibliotek ønsker at udvikle et it-system der gør det muligt for lånere at søge efter materiale. Materiale, der kan udlånes, udgøres af bøger og CD'er. Alt materiale har et id-nummer. Desuden har en bog en forfatter og en titel, mens en CD har en en titel og én eller flere kunstnere.

I it-systemet tænkes alt materiale beskrevet i den abstrakte klasse `Materiale`. Denne klasse stiller en metode `boolean match(String s)` til rådighed. Det er tanken, at denne metode returnerer værdien `true` hvis det pågældende stykke materiale matcher strengen `s`, ellers returnerer den værdien `false`. En bog matcher en streng `s` hvis enten bogens titel eller forfatter er lig med `s`. For eksempel vil en bog med titel "Eventyret om Ringen" og forfatter "J.R.R.Tolkien" matche både søgestrengen "Eventyret om Ringen" og strengen "J.R.R.Tolkien".

Tilsvarende matcher en CD en streng `s` hvis enten CD'ens titel eller en af dens kunstnere er lig med `s`. Hvis titlen er "Classic Jazz" og kunstnerne er "Armstrong", "Ellington" og "Basie", vil enhver af disse fire strenge matche CD'en.

Ud over den abstrakte klasse `Materiale` tænkes it-systemet blandt andet også at indeholde klasserne `Bog` og `CD`, således at `Materiale` er abstrakt super-klasse for klasserne `Bog` og `CD`. `Materiale` er defineret således:

```
abstract class Materiale {
    private int id;

    public Materiale(int id) {
        this.id = id;
    }

    public abstract boolean match(String s);

    public String toString() {
        return "ID: " + id;
    }
}
```

7.1 Skriv klassen `Bog` (5%)

Skriv klassen `Bog`. Den skal definere to felter, `forfatter` og `titel`, af typen `String` og de skal være `private`. Den skal også definere en passende konstruktør samt metoden `match()`.

7.2 Skriv klassen `CD` (10%)

Klassen `CD` skal tilbyde en metode til at tilføje en kunstner, således at op til 12 kunstnere kan blive tilknyttet en `CD`. Desuden skal klassen definere de nødvendige felter, en passende konstruktør samt metoden `match()`.

7.3 Skriv klassen `SerieBog` (5%)

Det forudses imidlertid at brugerne af søgefunktionen ofte ønsker at kunne finde bøger, der indgår i en serie, ved at søge efter navnet på serien. For eksempel bør man kunne søge efter "Ringenes Herre" (en serie) og få vist alle bøger i serien, herunder bogen med titel "Eventyret om Ringen" og forfatter "J.R.R.Tolkien".

Derfor udvides systemet med `SerieBog`, en underklasse af klassen `Bog` med to nye felter

• `serie`, som er en streng der angiver navnet på serien hvori den aktuelle bog indgår, for eksempel "Ringenes Herre"

• `nr`, som er et tal, der angiver nummeret på den aktuelle bog i serien.

Skriv klassen `SerieBog`, herunder felter, konstruktør, og metoden `match`.

8 Kunde og sælger (20%) -- datastrukturer

I en skitse til system, der skal hjælpe med at udregne provision til sælgere ud fra deres omsætning, er der indtil videre defineret to klasser: `Sælger` og `Kunde`. En kunde er i dette system blot kendetegnet ved sit firmanavn, en sælger er kendetegnet ved sit kaldenavn. I øvrigt gælder, at hver sælger højst har 10 kunder.

8.1 Skriv klassen `Sælger` (10%)

Klassen `Sælger` skal indeholde en datastruktur med referencer til sælgerens kunder, desuden skal klassen ud over konstruktøren have to metoder mere:

• `public boolean addKunde (Kunde kunde)`

- denne metode returnerer `true`, hvis det lykkedes at tilføje kunde-objektet til sælgeren, ellers returnerer den `false`

• `public String kundeliste()`

- denne metode returnerer en tegnstring med en beskrivelse alle sælgerens kunder, idet det kan antages, at klassen `Kunde` indeholder en brugbar `toString`-metode. I den returnerede tegnstring skal de enkelte beskrivelser være adskilt af et linieskift (tegnet `'\n'`).

8.2 Skriv klassen `Kunde` (10%)

Klassen `Kunde` skal indeholde en datastruktur, der refererer til objekter af klassen `Salg`, der er defineret således

```
class Salg {  
  
    public final String id;  
    public final long beløb;  
  
    public Salg(String id, long beløb) {  
        this.id = id;  
        this.beløb = beløb;  
    }  
}
```

Hvert element af klassen `Salg` repræsenterer således et salg til en kunde. Bemærk, at der ikke på forhånd kan siges noget om antallet af salg pr kunde.

Klassen `Kunde` skal udover konstruktøren indeholde metoderne

• `public void addSalg(Salg s)`

- denne metode tilføjer et salg til den pågældende kunde

• `public long omsætning()`

- denne metode skal returnere summen af alle de salgsbeløb, der er registreret på kunden

9 Bilag 1: Uddrag af dokumentationen for klassen java.io.FileReader

9.1 Konstruktøren FileReader

```
public FileReader(String fileName)  
    throws FileNotFoundException
```

Creates a new `FileReader`, given the name of the file to read from.

Parameters:

`fileName` – the name of the file to read from

Throws:

`FileNotFoundException` – if the specified file is not found

9.2 Metoden read

```
public int read()  
    throws IOException
```

Read a single character.

Overrides:

[read](#) in class [Reader](#)

Returns:

The character read, or `-1` if the end of the stream has been reached

Throws:

[IOException](#) – If an I/O error occurs

10 Bilag 2: Kode til EventFrame

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

public class EventFrame extends JFrame {

    private Game game = new Game();

    private JButton muladdButton = new JButton("*2 +1");
    private JButton mulButton = new JButton("*2 +0");
    private JButton resetButton = new JButton("Reset");
    private JLabel targetField = new JLabel();
    private JLabel currentField = new JLabel();
    private JTextField resultField = new JTextField();

    public EventFrame () {
        super("The *+game");

        JPanel buttonPanel = new JPanel(new GridLayout(1, 3));
        buttonPanel.add(muladdButton);
        buttonPanel.add(mulButton);
        buttonPanel.add(resetButton);

        JPanel labelPanel = new JPanel(new GridLayout(2, 1));
        labelPanel.add(targetField);
        labelPanel.add(currentField);

        resultField.setEditable(false);

        JPanel southPanel = new JPanel(new FlowLayout(FlowLayout.CENTER));
        southPanel.add(buttonPanel);
        JPanel northPanel = new JPanel(new FlowLayout(FlowLayout.CENTER));
        northPanel.add(resultField);
        JPanel centerPanel = new JPanel(new
FlowLayout(FlowLayout.CENTER));
        centerPanel.add(labelPanel);

        Container pane = getContentPane();
        pane.add(northPanel, "North");
        pane.add(southPanel, "South");
        pane.add(centerPanel, "Center");

        setBounds(50, 50, 220, 150);
        game.reset();
        refreshDisplay();
        show();
    }

    private void refreshDisplay() {
        targetField.setText("" + game.getTarget());
        currentField.setText("" + game.getCurrent());
        if (game.over()) {
            if (game.hit()) {
                resultField.setText("Spillet vundet");
                resultField.setBackground(Color.green);
            } else {
                resultField.setText("Spillet tabt");
            }
        }
    }
}
```

```
        resultField.setBackground(Color.red);
    }
} else {
    resultField.setText("Fortsæt ...");
    resultField.setBackground(Color.yellow);
}
}
```