Opgaveløsninger
(sæt 1)

Opgave 1: 1.13 (1 point)

De to konstruktioner er ikke ækvivalente, hvis `statements` indeholder en `continue`-sætning.

Eksempel:

```
for (int i = 1; i < 100; i++) {
    if (i % 10 == 0)
        continue;
    System.out.println(i);
}
```

Ovenstående programstump vil udskrive alle positive heltal mindre end 100, hvis sidste ciffer er forskellig fra 0. Følgende programstump vil derimod aldrig terminere.

```
int i = 1;
while (i < 100) {
    if (i % 10 == 0)
        continue;
    System.out.println(i);
    i++;
}
```

Opgave 2: 1.22 (2 point)

```
public class Cryptarithm1 {
  public static void main(String[] args) {
    for (int m = 0; m <= 9; m++)
      for (int a = 0; a <= 9; a++)
        if (a != m)
        for (int r = 0; r <= 9; r++)
            if (r != a && r != m)
            for (int k = 0; k <= 9; k++)
              if (k != r && k != a && k != m)
              for (int l = 0; l <= 9; l++)
                if (l != k && l != r && l != a && l != m)
                for (int e = 0; e <= 9; e++)
                if (e != l && e != k && e != r &&
                    e != a && e != m)
                  for (int n = 0; n <= 9; n++)
                    if (n != e && n != l && n != k &&
                        n != r && n != a && n != m)
                    for (int w = 0; w <= 9; w++)
                       if (w != n && w != e && w != l &&
                          w != k && w != r && w != a && w != m)
                      for (int i = 0; i <= 9; i++)
                        if (i != w && i != n && i != e &&
                            i != l && i != k && i != r && i != a &&
                            i != m)
                        for (int s = 0; s <= 9; s++)
                          if (s != i && s != w && s != n &&
                              s != e && s != l && s != k &&
                              s != r && s != a && s != m &&
                                          1000*m + 100*a + 10*r + k
                              + 10000*a + 1000*l + 100*l + 10*e + n ==
                                10000*w + 1000*e + 100*i + 10*s + s) {
                              System.out.println("A=" + a + " W=" + w +
                                " N=" + n + " R=" + r + " E=" + e +
                                " I=" + i + " M=" + m + " S=" + s);
  }
}
```

Ved kørsel af programmet udskrives følgende 20 løsninger:

```
A=1 W=2 N=9 R=4 E=0 I=8 M=3 S=5
A=1 W=2 N=6 R=4 E=0 I=8 M=3 S=5
A=5 W=6 N=9 R=7 E=2 I=4 M=3 S=0
A=5 W=6 N=1 R=7 E=2 I=4 M=3 S=0
A=8 W=9 N=7 R=1 E=0 I=4 M=3 S=2
A=8 W=9 N=5 R=1 E=0 I=4 M=3 S=2
A=2 W=3 N=9 R=1 E=4 I=0 M=5 S=6
A=2 W=3 N=7 R=1 E=4 I=0 M=5 S=6
A=1 W=2 N=9 R=0 E=3 I=8 M=6 S=4
A=1 W=2 N=5 R=0 E=3 I=8 M=6 S=4
A=1 W=2 N=9 R=7 E=5 I=0 M=6 S=3
A=1 W=2 N=4 R=7 E=5 I=0 M=6 S=3
A=2 W=3 N=4 R=9 E=6 I=0 M=8 S=5
A=2 W=3 N=1 R=9 E=6 I=0 M=8 S=5
A=4 W=5 N=9 R=3 E=6 I=2 M=8 S=0
A=4 W=5 N=1 R=3 E=6 I=2 M=8 S=0
A=5 W=6 N=9 R=7 E=3 I=0 M=8 S=1
A=5 W=6 N=2 R=7 E=3 I=0 M=8 S=1
A=1 W=2 N=7 R=4 E=5 I=8 M=9 S=0
A=1 W=2 N=3 R=4 E=5 I=8 M=9 S=0
```

En mere avanceret løsning kan ses på næste side. Klassen `Cryptarithmic2` er en generel klasse til løsning af kryptaritmiske additionsproblemer. Et eksempel på anvendelse kan ses i `main`-metoden.

Bemærk brugen af rekursion i hjælpemetoden `choose`. Metoden tildeler systematisk bogstavet i den `j`'te søjle i det `i'`te tal alle ubrugte cifferværdier. Søjlerne gennemløbes bagfra, og deres delsummer kontrolleres, så snart det er muligt. Parameteren `carry` angiver menten fra den sidst beregnede søjledelsum.

```
public class Cryptarithm2 {
    public Cryptarithm2(String[] problem) {
        number = problem;
        for (char ch = 0; ch < 256; ch++)
            if (Character.isLetter(ch))
                assigned[ch] = -1;
        for (char ch = 0; ch <= 9; ch++)
            assigned['0' + ch] = ch;
        assigned[' '] = 0;
    }

    public void solve() {
        choose(0, number[0].length() - 1, 0);
    }

    private void choose(int i, int j, int carry) {
        if (i == number.length) {
            int sum = carry;
            for (int k = 0; k < number.length - 1; k++)
                sum += assigned[number[k].charAt(j)];
            if (sum % 10 != assigned[number[i - 1].charAt(j)])
                return;
            if (j == 0) {
                if (assigned[number[i - 1].charAt(0)] == 0)
                    return;
                for (char ch = 0; ch < 256; ch++)
                    if (Character.isLetter(ch) && assigned[ch] >= 0)
                        System.out.print(ch + "=" + assigned[ch] + " ");
                System.out.println();
            } else
                choose(0, j-1, sum / 10);
        } else {
            char ch = number[i].charAt(j);
            if (assigned[ch] >= 0)
                choose(i+1, j, carry);
            else {
                for (int d = 0; d <= 9; d++) {
                    if (!used[d]) {
                        used[d] = true;
                        assigned[ch] = d;
                        choose(i+1, j, carry);
                        used[d] = false;
                        assigned[ch] = -1;
                    }
                }
            }
        }
    }

    private String[] number;
    private int[] assigned = new int[256];
    private boolean[] used = new boolean[10];

    public static void main(String[] args) {
        String[] problem = {"  six", " seven", " seven", "twenty"};
        new Cryptarithm2(problem).solve();
    }
}
```

Opgave 3: 2.13 (2 point)

(a)

```
import java.io.*;
import java.util.*;

public class OldestPersons1 {
    static class Person {
        Person(String name, int age) {
            this.name = name; this.age = age;
        }
        public String toString() {
            return name + " (" + age + ")";
        }
        String name;
        int age;
    }

    public static void main(String[] args) {
        if (args.length == 0) {
            System.out.println("No file specified");
            return;
        }
        BufferedReader fileIn = null;
        Person oldestPerson = null;
        try {
            fileIn = new BufferedReader(new FileReader(args[0]));
            String oneLine;
            while ((oneLine = fileIn.readLine()) != null) {
                StringTokenizer str = new StringTokenizer(oneLine);
                if (str.countTokens() != 2) {
                    System.out.println("Error: needs a name and an age");
                    return;
                }
                String name = str.nextToken();
                int age = Integer.parseInt(str.nextToken());
                if (oldestPerson == null || age > oldestPerson.age)
                    oldestPerson = new Person(name, age);
            }
            System.out.println(oldestPerson + " is oldest");
        }
        catch (IOException e)
          { System.err.println("Unexpected IO error"); }
        catch (NumberFormatException e)
          { System.err.println( "Error: needs an int" ); }
        finally {
            try {
                if (fileIn != null)
                    fileIn.close();
            } catch (IOException e) {}
        }
    }
}
```

(b)

```
import java.io.*;
import java.util.*;

public class OldestPersons2 {
    static class Person {
        Person(String name, int age) {
            this.name = name; this.age = age;
        }
        public String toString() { return name + " (" + age + ")"; }
        String name;
        int age;
    }

    public static void main(String[] args) {
        if (args.length == 0) {
            System.out.println("No file specified"); return;
        }
        BufferedReader fileIn = null;
        ArrayList oldestPersons = new ArrayList();
        int maxAge = 0;
        try {
            fileIn = new BufferedReader(new FileReader(args[0]));
            String oneLine;
            while ((oneLine = fileIn.readLine()) != null) {
                StringTokenizer str = new StringTokenizer(oneLine);
                if (str.countTokens() != 2) {
                    System.out.println("Error: needs a name and an age");
                    return;
                }
                String name = str.nextToken();
                int age = Integer.parseInt(str.nextToken());
                if (age >= maxAge) {
                    if (age > maxAge) {
                      oldestPersons = new ArrayList();
                      maxAge = age;
                   }
                   oldestPersons.add(new Person(name, age));
                }
            }
            System.out.println("Oldest persons:");
            Iterator itr = oldestPersons.iterator();
            while (itr.hasNext())
                  System.out.println(itr.next());
        }
        catch (IOException e)
          { System.err.println("Unexpected IO error"); }
        catch (NumberFormatException e)
          { System.err.println( "Error: needs an int" ); }
        finally {
            try {
                if (fileIn != null)
                    fileIn.close();
            } catch (IOException e) {}
        }
    }
}
```

Opgave 4: 3.13 (1 point)

```
public class Lock {
    public Lock(int n1, int n2, int n3) {
        number1 = n1;
        number2 = n2;
        number3 = n3;
    }

    public void open(int n1, int n2, int n3) {
        if (n1 == number1 && n2 == number2 && n3 == number3)
            isOpen = true;
    }

    public void changeCombo(int n1, int n2, int n3) {
        if (isOpen) {
            number1 = n1;
            number2 = n2;
            number3 = n3;
            isOpen = false;
        }
    }

    private int number1, number2, number3;
    private boolean isOpen;
}
```

Opgave 5: 3.15 (3 point, ikke obligatorisk)

```
class Fraction implements Comparable {
    Fraction(int a, int b) {
        if (b == 0)
            throw new RuntimeException("Fraction: 0 in denominator");
        if (a == 0) { this.b = 1; return; }
        if (b < 0) { a = -a; b = -b; }
        int d = gcd(Math.abs(a), b);
        this.a = a/d; this.b = b/d;
    }

    Fraction(int a) { this.a = a; this.b = 1; }

    public int numerator() { return a; }

    public int denominator() { return b; }

    public Fraction add(Fraction f)
      { return new Fraction(a*f.b+b*f.a, b*f.b); }

    public Fraction subtract(Fraction f)
      { return new Fraction(a*f.b-b*f.a, b*f.b); }

    public Fraction multiply(Fraction f)
      { return new Fraction(a*f.a, b*f.b); }

    public Fraction divide(Fraction f)
      { return new Fraction(a*f.b, b*f.a); }

    public boolean equals(Object o) {
        Fraction f = (Fraction) o;
        return a == f.numerator && b == f.denominator;
    }

    public int compareTo(Object o) {
        return subtract((Fraction) o).a % 1;
    }

    public String toString() {
        return a == 0 || b == 1 ? "" + a : a + "/" + b;
    }

    private int a, b;

    private int gcd(int u, int v) {
        return v == 0 ? u : gcd(v, u % v);
    }
}
```

Nedenstående program udskriver summen af 1/i for i = 1 til 10.

```
public class Program {
    public static void main(String args[]) {
        Fraction sum = new Fraction(0);
        for (int i = 1; i <= 10; i++)
            sum = sum.add(new Fraction(1, i));
        System.out.println(sum);
    }
}
```

En kørsel gav følgende (korrekte) udskrift:

```
7381/2520
```