12.03.02	Theorietest aus **EINFÜHRUNG IN DAS PROGRAMMIEREN** LVANr.: 183 046		EI7 13:00–14:30
Matrikelnr.:	Nachname:	Vorname:	Gruppe **0**
Kennzahl:			Punkte:

Bitte tragen Sie **zuerst** Ihre **Matrikelnummer, Kennzahl, Nachnamen und Vornamen** in die dafür vorgesehenen Kästchen ein. Es sind alle schriftlichen Unterlagen erlaubt. Es sind keine technischen Hilfmittel erlaubt! Wärend des Tests herrscht **striktes Handy–Verbot**! Schreiben Sie ihre Lösungen auf den Testbogen. Es werden **keine Zusatzblätter** akzeptiert! Mit der Lösung aller Aufgaben können Sie insgesamt **maximal 100 Punkte** erreichen. **ACHTEN** Sie auf die Punkteverteilung bei den Beispielen!

1. Aufgabe: Ausdrücke (18 Punkte)

Im Folgenden sind einige Variable deklariert. Im Anschluß daran werden einige Ausdrücke angeboten. Bestimmen Sie für jeden dieser Ausdrücke, ob er gültig oder ungültig ist. Erläutern Sie ihre Entscheidung.

```
public class ausdruecke
{
  int a ; double b; short c; byte d ; long e ; float f ;
  char g ; boolean h; String i;
    ...
}
```

a) `(a<=++b)` _____

b) `g.charAt(b)` _____

c) `~a` _____

d) f%f _____

e) `i+i` _____

f) `a<<<3` _____

2. Aufgabe: Fehler finden (21 Punkte)

In der folgenden Klasse sind einige Fehler enthalten, die das Programm nicht compilierbar machen. Finden Sie die Fehler und kreuzen sie fehlerhafte Codezeilen an:

```
import eprog.*;
public class testen extends EprogIO
{

    public static String perform ()
    {
      short z1,z2,n1,n2,t,t2;
      int resz, resn,ti1,ti2;
      boolean ok,neg;
      String op;
      neg=false;
      ok=false;
      try {
          resz==0;
          resn=0;
          z1=readShort();
          n1=readShort();
          z2=readShort()
          n2=readShort();
          op=readWord();
          if (!((n1=0) || (n2=0)))
            {
              if (op.equals("/")) {
                resz=z1*n2;
                resn=z2*n1;
                ok=true;
              }
              if (op.equals("*")) {
                resz=z1*z2;
                resn=n1*n2;
                ok=true;
              }
              if (op.equals("+")) {
                ti1=z1*n2;
                ti2=z2*n1;
                resz=ti1++ti2;
                resn=n1*n2;
                ok=true;
              }
              if (op.equals(-)) {
                ti1=z1*n2;
                ti2=z2*n1;
                res=ti1-ti2;
                resn==n1*n2;
                ok=true
              }
            }
          if (ok==false) {
            return "FALSCHE EINGABE";
          } else {

            if ((resz<0) && (resn>=0)) {
                resz=-1*resz;
                neg=true;
            }

            if ((resz>=0) && (resn<0)) {
                resn=*-1;
                neg=true;
            }

            if (resz >= resn) {
                ti2=resn;
            } else {
                ti2=resz;
            }
```

```
            while (ti2>1) {
                if ((resz % ti2==0) && (resn % ti2==0)) {
                  resz/=ti2;
                  resn=/ti2;
                } else
                  {
                      ti2--;
                  }
            }
            if (resz=0) {
                resn=1;
            }
            if (neg!=true) {
                return resz+" "+resn;
            } else {
                return "-"+resz++" "+resn;
            }
          }
      } catch (EprogException e)
          {
            return "?"
          }
    }
}
```

3. Aufgabe: Methoden erstellen (17 Punkte)

Schreiben sie eine Klasse `Matrix`, die von einer **vorgegebenen** Klasse `SimpleMatrix` erbt und folgende funktionelle Erweiterungen besitzt.

* **Potenzen einer Matrix:**

  Aufruf A.exponent(n) entspricht $A=A^n$

  Beispiel: $A^2 = A*A$, $A^3=A * A * A$

  Definition: Ist A eine (m,n)–Matrix und B eine (n,p)–Matrix, dann heisst die (m,p)–Matrix C das Produkt der Matrizen A und B genau dann, wenn für die Elemente der Matrix gilt:

  $c_{ij}= a_{i1} * b_{1j} + a_{i1} * b_{2j} + ... + a_{in} * b_{pj}$ mit $1 \le i \le m$ und $1 \le j \le p$

  Man beachte, dass das Produkt zweier Matrizen nur dann erklärt ist, wenn die linke Matrix ebenso viele Spalten hat, wie die rechte Matrix Zeilen.

* **Gleichheit zweier Matrizen:**

  Aufruf equal(A,B) ergibt 1 falls A=B, sonst 0.

  Zwei Matrizen A=B heißen gleich, falls die Anzahl der Zeilen und Spalten übereinstimmen und die Matrizen koeffizientenweise gleich sind, d.h. $a_{ij}= b_{ij}$.

  Beachten Sie, dass beide Matrizen diesselbe Dimension aufweisen müssen.

Verwenden Sie bei allen Funktionen das Exception–Handling analog zur Vorgabe.
Die Klasse `SimpleMatrix` ist wie folgt dokumentiert (Das Programm ermöglicht es, **`int`** Werte von einem Benutzer zu erfragen und diese in Matrizen einzutragen, auszulesen oder Werte einer anderen Matrix aufzuaddieren). **Diese Klasse ist vorgegeben und daher für die Lösung der Aufgabe nicht auszuprogrammieren!**

`public SimpleMatrix( int rows, int columns )`
Es wird eine Matrix für Ganzzahlen in der gegebenen Größe erstellt. Ihre Felder sind mit 0 intialisiert.

`public int getRows( )`
liefert die Anzahl der Reihen der Matrix

`public int getColumns()`
liefert die Anzahl der Spalten der Matrix

`public void setElement( int row, int column, int value ) throws OutOfMatrixDimensionException`
Das referenzierte Element der Matrix nimmt den Wert „value`` an. Bei Angabe falscher Reihen– oder Spaltenzahlen wird eine `OutOfMatrixDimensionException` erzeugt und ausgelöst.

`public int getElement( int row,int column ) throws OutOfMatrixDimensionException`
Der Wert des referenzierten Elements wird zurückgeliefert. Bei Angabe falscher Reihen– oder Spaltenzahlen wird eine `OutOfMatrixDimensionException` erzeugt und ausgelöst.

`public void add( SimpleMatrix otherMatrix ) throws MismatchMatrixDimensionsException`
Die Argument–Matrix wird zur Matrix hinzuaddiert. Sollten die Matrixausdehnung für diese Operation nicht konform sein, wird eine `MismatchMatrixDimensionsException` Ausnahme ausgelöst.

`public void print()`
Einfach formatierte Ausgabe der Matrix auf die Standardausgabe.

4. Aufgabe: Programm nachvollziehen (24 Punkte)

Gegeben ist untenstehendes Programm. Geben Sie bei der Eingabe in die Variablen a, b, c die letzten drei Stellen Ihrer Matrikelnummer ein (z.B. Matrikelnummer ist 1234567, nach der Eingabe haben die Variablen folgenden Wert: a = 5, b = 6, c = 7). Führen Sie dieses Programm aus, und geben Sie an, welche Ausgaben das Programm liefert. Tragen Sie diese in folgende Tabelle ein.

Ausgabetabelle

1. Spalte	2. Spalte	3. Spalte	4. Spalte

```
import eprog.*;

public class nachvollziehen
{
    final static int Y = 0;
    final static int Z = 6;
    static int a=0,b=0;

    public static int A(int x,int y)
    {
      int c=0,b=0;
      {
          int z=5;
          c=z;
          b=y;
      }
      return a;
    }

    public static int B(int x,int y)
    {
      int c=2,a=0;
        {
          int z=c;
          x=b;
      }
      c=x;
      a=y;
      return x;
    }

    public static int C(int x,int y)
    {
      int c=b;
      a=++x+y;
      b=++x;
      return c++;
    }
```

```
    public static void D(int x,int y,int z)
    {
      x=++a;
        b+=++y+z;
      z=x;
        y=C(x,x);
    }

    public static void E(int x,int y,int z)
    {
        x=y++;
        z=y++;
        y+=a+++b;
    }

    public static void F(int x,int y,int z)
    {
      a=++x+z++;
    }

    public static int G(int x,int y)
    {
      int a=0;
      a=+x;
      b=y;
      return y;
    }

    public static void main(String[] args)
    {
        int c=0,d=0;
        try
           {
             a=EprogIO.readInt();
             b=EprogIO.readInt();
             c=EprogIO.readInt();
           }
      catch (EprogException e)
           {
           }
      for (d=Y;++d<Z;EprogIO.println(d++))
           {
             switch (d)
                 {
                 case 0: a=A(b,c); break;
                 case 1: b=B(a,c); break;
                 case 2: c=C(c,b); break;
                 case 3: D(a,b,c); break;
                 case 4: E(b,a,c); break;
                 case 5: F(b,c,a); break;
                 case 6: c=G(a,c); break;
                 }
             EprogIO.print(a+"\t"+b+"\t"+c+"\t");
           }
    }
}
```

5. Aufgabe: Klassendesign (20 Punkte)

a) Design einer Klasse Pyramid
Die 2 Parameter zur Beschreibung einer quadratischen geraden Pyramide (z.B. Aus Pappkarton) sind 1) **Länge** und 2) die **Höhe**. Mit diesen 2 Parametern können nun weitere Merkmale einer rechteckigen Pyramide berechnet werden:

– **Grundfläche**: Länge²
– **Volumen:** Länge²*Höhe/3
– **Mantelfläche:** 2*Länge*Quaratwurzel(Länge²/4*Höhe²)

Programmieren Sie eine Klasse Pyramid, deren Instanzvariablen die oben beschriebenen Parameter abbilden (double), einen geeigneten Konstruktor sowie die Methoden zur Berechnung der weiteren Merkmale enthält.

Beispiel:

```
class Pyramid
    {
     // Instance Variables
     ...
     // Constructors
     ...
     // Methods
     ...
    }
```

b) Um das Design zu überprüfen, schreiben Sie ein Programm PyramidTester welches eine Pyramide mit den Werten 5.0 und 6.0 erzeugt und anschliessend die Grundfläche, Volumen sowie die Mantelfläche ausgibt.