Aufgabe 4: Programmverstehen, Fehler Finden (20 Punkte)

Gegeben ist eine Klasse `Schihuette`:

```
class Schihuette {
  static String bezeichnung;  // Bezeichnung einer Schihuette
  static int seeHoehe;        // Meter ueber Meeresspiegel einer Schih.
  boolean inBetrieb = true;   // true, nur wenn Schihuette in Betrieb
  ...
  void setSeeHoehe (int s) {
    seeHoehe = s;
  }

  boolean getSeeHoehe () {
    return seeHoehe;
  }

  ...
  public static void main(String[] arg) {  // zum Testen
    Schihuette h1 = new Schihuette();
    Schihuette h2 = new Schihuette();
    h1.setSeeHoehe(In.readInt());          // Klasse In für Eingabe
    h2.setSeeHoehe(In.readInt());
    Out.println (h1.getSeeHoehe());        // Klasse Out für Ausgabe
    Out.println (h2.getSeeHoehe());
    ...
    if (!h1.inBetrieb && !h2.inBetrieb)
      Out.println ("Beide Schihuetten nicht in Betrieb!");
    else Out.println ("Beide Schihuetten in Betrieb!");
  }
}
```

a) (10 Punkte) Ausführliches Testen dieses Programms ergibt, dass **beide** Aufrufe von `getSeeHoehe` stets den Wert der Schihuette h2 ausgeben, auch wenn für h1 und h2 verschiedene Seehöhen eingegeben werden.
Finden Sie die Fehlerstellen und korrigieren Sie die Fehler.

b) (10 Punkte) Außerdem liefert der Test, ob beide Schihütten in Betrieb sind, ein falsches Ergebnis.
Finden Sie auch hier die Fehlerstelle und korrigieren Sie den Fehler so, dass die beiden gegebenen Ausgabetexte zu den abzufragenden Wahrheitswerten von `inBetrieb` passen.

Aufgabe 5: Theoriefragen (10 Punkte)

Frage	Antwort	Punkte
5.1 **Overloading** Was bedeutet eine "Methode überladen"? Beschreiben Sie das Überladen von Methoden im Allgemeinen.		5

5.2 Geben Sie den **Programmcode** einer Methode an, die, die gegebene Methode add überladet.
 	
`class arithmo {`
`  double d;`
`  void add (int i){`
`  d = d + i;`
`  }`
`...`
`}`
		5

PRÜFUNGSORDNER - ein Service Deiner Fachschaft Informatik!

LVA: EPROG VO NEU Preis:

28.1.03 Gruppe B

Aufgabe 1: Verkettete Liste (35 Punkte)

Gegeben sei eine einfach verkettete, sortierte Liste (Klasse **ReservierungsListe**), mit der die Tischreservierungen in einem Restaurant verwaltet werden. Jeder Knoten enthält eine **Reservierung** bestehend aus fünf Integerwerten (ein Datum Format: JJJJMMTT, eine Beginn-Uhrzeit, eine Ende-Uhrzeit Format: HHMM, eine Tischnummer und eine Telefonnummer). Zusätzlich beinhaltet Reservierung noch einen String, nämlich den Namen der Person, für die der Tisch reserviert wurde. Die Sortierung der Liste ist nach Datum und Name bereits gegeben.

Aufgabenstellung: Implementieren Sie eine Methode „**Suchen**", die überprüft, ob für einen bestimmten Namen und ein bestimmtes Datum eine Reservierung eingetragen ist. Der gesuchte Name und das gesuchte Datum werden als Parameter an die Methode übergeben. Falls eine Reservierung vorliegt, soll der entsprechende Knoten aus der Methode zurück geliefert werden. Es ist davon auszugehen, dass die Liste nur gültige Daten enthält.

Punkteverteilung:
1. Algorithmische Richtigkeit. (25 Punkte)
2. Syntaktische Richtigkeit. (5 Punkte)
3. Variablendeklarationen. (5 Punkte)

Aufgabe 2: Klassifikation (15 Punkte)

Bilden Sie eine Klassifizierung von Artikel und Leistungen, die in einem **Computerfachgeschäft** angeboten werden (z.B. Reparaturen, Upgrades, Software, Hardware, Betriebssysteme, Notebooks, Komplettsysteme, Festplatten, Kaufberatungen, etc.) und zeichnen Sie die zugehörige Klassenhierarchie. Mindestens drei Klassenhierarchiestufen müssen abgebildet sein. Überlegen Sie sich zu jeder Klasse geeignete Attribute (bzw. Felder) mit Typ (z.B. int, etc.). Überlegen Sie sich passende Methoden (insgeamt mindestens vier).

Punkteverteilung:
1. Klassen. (5 Punkte)
2. Attribute. (5 Punkte)
3. Methoden. (5 Punkte)

Aufgabe 3: Array: Palindrom (20 Punkte)

DUDEN: Palindrom das; Wort[folge] od. Satz, die vorwärts wie rückwärts gelesen [den gleichen] Sinn ergeben, z. B. Reittier; Rentner; Reliefpfeiler; Otto; Anna; Hannah; etc.

Aufgabenstellung: Gegeben ist ein eindimensionales Array vom Datentyp char dessen Elemente ein Wort ergeben. Schreiben Sie eine Methode, die überprüft, ob es sich um ein Palindrom handelt und geben Sie das Ergebnis als Returnwert vom Typ boolean zurück.

Punkteverteilung:
1. Algorithmische Richtigkeit. (15 Punkte)
2. Syntaktische Richtigkeit. (5 Punkte)

Aufgabe 4: Programmverstehen, Fehler Finden (20 Punkte)

Gegeben ist eine Klasse `Schilift`:

```
class Schilift {
  static String bezeichnung;      // Bezeichnung eines Schiliftes
  static int seeHoehe;            // Meter ueber Meeresspiegel
  boolean inBetrieb = true;       // true, nur wenn Schilift in Betrieb
  ...
  void setBezeichnung(String s) {
    bezeichnung = s;
  }

  String getBezeichnung () {
    return bezeichnung;
  }

  ...
  public static void main(String[] arg) {  // zum Testen
    Schilift s1 = new Schilift();
    Schilift s2 = new Schilift();
    s1.setBezeichnung(In.readString());     // Klasse In für Eingabe
    s2.setBezeichnung(In.readString());
    Out.println (s1.getBezeichnung());      // Klasse Out für Ausgabe
    Out.println (s2.getBezeichnung());
  ...
    if (!s1.inBetrieb || !s2.inBetrieb)
      Out.println ("Beide Schilifte nicht in Betrieb!");
    else Out.println ("Beide Schilifte in Betrieb!");
  }
}
```

Aufgabenstellung:

a) (10 Punkte) Ausführliches Testen dieses Programms ergibt, dass **beide** Aufrufe von `getBezeichnung` stets die Bezeichnung des Schilifts s2 ausgeben, auch wenn für s1 und s2 verschiedene Bezeichnungen eingegeben werden. Finden Sie die Fehlerstellen und korrigieren Sie die Fehler.

b) (10 Punkte) Außerdem liefert der Test, ob beide Schilifte nicht in Betrieb sind, ein falsches Ergebnis. Finden Sie auch hier die Fehlerstelle und korrigieren Sie den Fehler so, dass die beiden gegebenen Ausgabetexte zu den abzufragenden Wahrheitswerten von `inBetrieb` passen.

Aufgabe 5: Theoriefragen (10 Punkte)

Frage	Antwort	Punkte
5.1 **Überschreiben (Redefinition)** Was *bedeutet* eine "Methode überschreiben"? Beschreiben Sie das Überschreiben von Methoden im Allgemeinen.		5

5.2 Geben Sie einen **Programmcode** an, der, zeigt, wie die gegebene Methode `add` überschrieben wird.
`class arithmo {`
`  double d;`
`  void add (int i){`
`  d = d + i;`
`  }`
`...`
`}`
		5

22.01.2002	Theorietest aus **EINFÜHRUNG IN DAS PROGRAMMIEREN** LVANr.: 183 046		Audi Max / EI7 13:00-14:30
Matrikelnr.: [redacted]	Nachname: [redacted]	Vorname: [redacted]	Gruppe **B**
Kennzahl: E033 534			Punkte:

12
19
18
18
20
64
81
(1)

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) (12)

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) (f<<f) Ungültig ✓; Bit Operationen sind auf float nicht anwendbar ✓

b) i.charAt(d) ungültig, charAt verlangt ~~ist~~ als Parameter ~~gültig, charAt der String anwendbar (Methode von String), da~~

c) a>>>c gültig, Bitshift auf int zulässig, short zulässig für Konstante für shift ✓

d) d&c ~~gültig~~ ungültig, ungleiche ~~Länge~~ an bits

e) i+i gültig ✓ + Operator ist für String definiert ✓

f) g++ ~~ist~~ gültig ✓, erhöht Wert in Unicode ✓

2. Aufgabe: Fehler finden (21 Punkte) (14)

a) Folgende Klasse soll die Faktorielle einer Zahl berechnen. Dabei sind dem Programmierer ein paar Fehler unterlaufen. Einerseits erzeugt das Programm Fehler bei der Übersetzung (Compilererror), andererseits ist ein Fehler im Algorithmus. Ihre Aufgabe ist es die Fehler zu finden und in die linke Spalte, die korregierte Version der Programmzeile zu schreiben. Zur Erinnerung : n! = (n-1)!*n

```
import eprog.*;

class Fakt{
    staticstatic void main (String argv[]) {
        int n=0;
        try {
            n=EprogIO.readInt();
        }
        catch (Exception e) {
            EprogIO.println("?");
        }
        EprogIO.println(a(n));
    }

    static int a(int a)
    {
        return (a>0)?c(a,a(--a));1;
    }

    static int b(int a, int b) {
        return (a==1)?b++:b(--a,++b);
    }

    static int c(int a, int b) {
        return (b==0)?0:b(a,c(a,--b))
    }
}
```

public static void main (String args[]){ +0

return (a>0)? c(a, a(--a)) : 1; +1

+1

return (b=0) ? 0 : (a,c(a,--b)); +1

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

```
public class Rational
{
    private long numerator = 0;
    private long denominator = 1;

    public Rational()
    {
      this(0, 1);
    }

    public Rational(long numerator, long denominator)
    {
      long gcd = Gcd(numerator, denominator);
      this.numerator = numerator/gcd;
      this.denominator = denominator/gcd;
    }

    private long gcd(long n, long d)
    {
      long t1 = Math.abs(n);
      long t2 = Math.abs(d);
      long remainder = t1%t2;

      while (remainder != 0)
          {
            t1 := t2;
            t2 := remainder;
            remainder := t1%t2;
          }

      return t2;
    }

    public long getNumerator()
    {
      return n;
    }

    public long getDenominator()
    {
      return d;
    }

    public Rational add(Rational secondRational)
    {
      long n = numerator*secondRational.getDenominator() +
          denominator*secondRational..getNumerator();
      long d = denominator*secondRational.getDenominator();
      return new Rational(n, d);
    }

    public Rational subtract(Rational secondRational)
    {
      long n = numerator*secondRational.getDenominator()
          - denominator*secondRational.getNumerator();
      long d = denominator*secondRational.getDenominator();
      return new rational(n, d);
    }

    public Rational multiply(Rational secondRational)
    {
      long n = numerator*secondRational.getNumerator();
```

```
        long d = denominator*secondRational.getDenominator();
        return new Rational(ne, d);
    }

    public Rational divide(Rational secondRational)
      throws RuntimeException
    {
      if (secondRational.getNumerator() == 0)
          throw new RuntimeException("Denominator cannot be zero");

      long n = numerator*secondRational.getdenominator();
      long d = denominator*secondRational.getNumerator();
      return new Rational(n; d);
    }

    public String toString()
    {
      if (denominator = 1)
          return numerator + "";
      else
          return numerator + "/" + denominator;
    }
}
```

3. Aufgabe: Methoden erstellen (17 Punkte)

Ein Programmierer hat eine Klasse `SimpleMatrix` entsprechend den ihm vorliegenden Anforderungen programmiert und 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:

`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.

Leider reicht die bislang programmierte Funktionalität der Klasse `SimpleMatrix` nicht aus. Schreiben sie eine Klasse `Matrix`, die von der hier beschriebenen Klasse erbt und folgende funktionelle Erweiterungen besitzt.

* **Subtraktion einer Matrix:**
  Aufruf A.subtract(B) entspricht A=A–B.

  Definition: Die (m,n)–Matrix C heisst genau dann die Differenz der Matrizen A und B, wenn für die Elemente von C gilt:

  $$c_{ij} = a_{ij} - b_{ij} \quad ; \text{mit } 1 \le i \le m \text{ und } 1 \le j \le n$$

  Man beachte, dass das Differenz zweier Matrizen nur dann erklärt ist, wenn beide Matrizen diesselben Dimensionen besitzen.

* **Matrix symmetrisch machen:**
  Aufruf A.symmetric() entspricht A^S.
  ($a^S_{ij} = a_{ji}$ wenn $i \le j$, sonst $a^S_{ij} = a_{ij}$; mit $1 \le i,j \le m$)

  Man beachte, dass die Matrix quadratisch sein muss.

Verwenden Sie das Exception–Handling bei allen Funktionen analog zur Vorgabe.

4. Aufgabe: Programm nachvollziehen (24 Punkte) (18)

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.

-2

Ausgabetabelle

1. Spalte	2. Spalte	3. Spalte	4. Spalte
1	0	5	0
2	6	2	5
3	15	19	6

```
import eprog.*;

public class nachvollziehen
{
    final static int Y = 1;
    final static int Z = 4;
    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) (20)

a) Design einer Klasse Box:
Die 3 Parameter zur Beschreibung einer Schachtel (z.B. Aus Pappkarton) sind 1) **Länge**, 2) die **Breite** und 3) die **Höhe**. Mit diesen 3 Parametern können nun weitere Merkmale einer Schachtel berechnet werden:

- **Grundfläche**: Länge*Breite
- **Volumen:** Länge*Breite*Höhe
- **Oberfläche:** 2*((Länge*Breite)+(Länge*Höhe)+(Breite*Höhe))

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

Beispiel:
```
class Box
    {
    // Instance Variables
    ...
    // Constructors
    ...
    // Methods
    ...
    }
```

b) Um das Design zu überprüfen, schreiben Sie ein Programm BoxTester welches eine Schachtel mit den Werten 2.5, 5.0 und 6.0 erzeugt und anschliessend die Grundfläche, Volumen sowie die Oberfläche ausgibt.

a)
```
public class Box
{ double Laenge, ✓
  double Breite, ✓
  double Hoehe, ✓
  public Box(double a, double b, double c)
  { Laenge = a; ✓
    Breite = b; ✓
    Hoehe = c; } ✓
  public double GF()
  { return Laenge * Breite; ✓
  public double V()
    return Laenge * Breite * Hoehe; ✓
  public double OF()
    return 2 * ((Laenge * Breite) + (Laenge * Hoehe) + (Breite * Hoehe)); ✓
}
```

b)
```
public class BoxTester
{ public static void main(String [] args)
  { double a = 2.5, b = 5.0, c = 6.0;
    testBox = new Box(a, b, c); ✓
    double x = testBox.GF(); ✓
    Ergebnis.println(x); ✓
    double y = testBox.V();
    Ergebnis.println(y);
```

PRÜFUNGSORDNER - ein Service Deiner Fachschaft Informatik!

LVA: EPROG VO NEU Preis:

28.1.03 Gruppe A

Aufgabe 1: Verkettete Liste (35 Punkte)

Gegeben sei eine einfach verkettete, sortierte Liste (Klasse **SitzplatzListe**), mit der die Sitzplätze in einem Kino verwaltet werden. Jeder Knoten enthält eine **Sitzplatzreservierung** bestehend aus:
- **vier Integerwerten**: ein Datum Format: JJJJMMTT, eine Beginn-Uhrzeit: HHMM, eine zweistellige Saalnummer, und eine max. 10stellige Nummer, die den Film bezeichnet,
- **einem Array**, in dem die reservierten Sitzplätze mit Reihennummer und Sitzplatznummer gespeichert sind. Pro Person können max. 5 Plätze reserviert werden,
- **einem String**, der Name der Person, für die die Sitzplätze reserviert wurden.

Die Sortierung der Liste ist nach Datum und Filmnummer bereits gegeben.

Aufgabenstellung: Implementieren Sie eine Methode „**Suchen**", die überprüft, ob für einen bestimmten Namen und ein bestimmtes Datum eine Reservierung für einen bestimmten Film eingetragen ist. Der gesuchte Name, das gesuchte Datum und der gesuchte Film werden als Parameter an die Methode übergeben. Falls eine Sitzplatzreservierung vorliegt, soll der entsprechende Knoten aus der Methode zurück geliefert werden. Es ist davon auszugehen, dass die Liste nur gültige Daten enthält.

Punkteverteilung:
1. Algorithmische Richtigkeit. (25 Punkte)
2. Syntaktische Richtigkeit. (5 Punkte)
3. Variablendeklarationen. (5 Punkte)

Aufgabe 2: Klassifikation (15 Punkte)

Bilden Sie eine Klassifizierung von Artikel und Leistungen, die in einem **Elektronikfachmarkt** angeboten werden (z.B. Videofilme, Musik-CDs, Mikrowellen, Fernsehgeräte, Telefone, Computer, Software, CD-Rohlinge, Videokassetten, Reparaturen, Kaufberatungen, etc.) und zeichnen Sie die zugehörige Klassenhierarchie. Mindestens drei Klassenhierarchiestufen müssen abgebildet sein. Überlegen Sie sich zu jeder Klasse geeignete Attribute (bzw. Felder) mit Typ (z.B. int, etc.). Überlegen Sie sich passende Methoden (insgeamt mindestens vier).

Punkteverteilung:
1. Klassen. (5 Punkte)
2. Attribute. (5 Punkte)
3. Methoden. (5 Punkte)

Aufgabe 3: Array: Palindrom (20 Punkte)

DUDEN: Palindrom das; Wort[folge] od. Satz, die vorwärts wie rückwärts gelesen [den gleichen] Sinn ergeben, z. B. Reittier; Rentner; Reliefpfeiler; Otto; Anna; Hannah; etc.

Aufgabenstellung: Gegeben ist ein eindimensionales Array vom Datentyp char dessen Elemente ein Wort ergeben .Schreiben Sie eine Methode, die überprüft, ob es sich um ein Palindrom handelt und geben Sie das Ergebnis als Returnwert vom Typ boolean zurück.

Punkteverteilung:
1. Algorithmische Richtigkeit. (15 Punkte)
2. Syntaktische Richtigkeit. (5 Punkte)

Aufgabe 5: Theoriefragen

(10 Punkte)

Frage	Antwort	Punkte
5.1 **Polymorphie.** Was ist Polymorphie? Welchen Zweck erfüllt sie? Geben Sie ein Beispiel für Polymorphie!		**5**

5.2 **Zuweisungskompatibilität.**
Gegeben ist eine Objektklasse C. Objekte welcher Klassen dürfen einem Objekt der Klasse C mittels einer Zuweisung (=) zugewiesen werden?
 	
`class C {}`
`...`
`C cvar;`
`...`
`cvar = ???;`
		5

Aufgabe 4: Programmverstehen, Fehler Finden (20 Punkte)

Gegeben ist folgende Methode berechneZ:

```
public static int berechneZ (int[] a) {
  int z, m = a[0], i = 1;
  while (i < a.length && a[i] == m)
    i++;
  if (i == a.length) return m;
  if (a[i] < m)
    {z = m; m = a[i];}
  else z = a[i];
  i++;
  while (i < a.length) {
    if (a[i] < m) { z = m; m = a[i]; }
    else if (a[i] > m && a[i] < z) z = a[i];
    i++;
  }
  return z;
}
```

4.1 (10 Punkte) Was liefert der folgende Aufruf der Methode berechneZ, wenn Sie hier beispielhaft das Array a mit den Ziffern Ihrer eigenen Matrikelnummer initialisieren?

int a[] = {_,_,_,_,_,_,_,_}; // hier die Ziffern Ihrer Matrikelnummer eintragen

int result = berechneZ(a);

result = ☐

4.2 (5 Punkte) Bei welchen Parameterwerten würde ein Aufruf von `berechneZ` zu einem Absturz zur Laufzeit (bzw. Ausnahme) führen?

4.3 (5 Punkte) Welche Wirkung hat die Methode `berechneZ` im Allgemeinen? Was liefert `berechneZ` bei beliebigem Eingangsparameter a?

3.3.03 Gruppe B

Aufgabe 1: Bücherverwaltung (35 Punkte)

Gegeben sei eine einfach verkettete, sortierte Liste (Klasse **BuchBestand),** mit der die Bücher in einer Buchhandlung verwaltet werden. Jeder Knoten beschreibt ein **Buch:**

```
class Buch {
        int ISBN;
        String Titel;
        int Preis;
        int Jahr;
        int Stueck;
        Buch next;

        Buch(int ISBN, String Titel, int Preis, int Jahr, int Stueck, Buch n)
       { this.ISBN = ISBN; this.Titel = Titel; this.Preis = Preis;
         this.Jahr = Jahr; this.Stueck = Stueck; next = n; }
}
```

Die Liste ist nach der ISBN Nummer aufsteigend sortiert.

Aufgabenstellung: Implementieren Sie eine Methode „**Loeschen**", die ein Buch mit einer bestimmten ISBN Nummer aus der Liste entfernt. Falls die Stückanzahl bei diesem Buch größer 1 ist, soll die Stückanzahl um eins verringert werden. Falls die Stückanzahl gleich 1 ist, soll der Knoten gelöscht werden.

Punkteverteilung:
1. Algorithmische Richtigkeit. (30 Punkte)
2. Syntaktische Richtigkeit. (5 Punkte)

Aufgabe 2: Schleifen (15 Punkte)

Gegeben ist die folgende while-Schleife:

```
int i = 0; j = a.lenght()-1;

while (i < a.length() && j > i && a.charAt(i) == a.charAt(j)) {
                        i++; j--;
        }

a ist ein String.
```

Aufgabenstellung:
2.1 (5 Punkte) Beschreiben Sie die Funktionsweise dieser Schleife. Welches Ergebnis liefert diese Schleife?
2.2 (5 Punkte) Wandeln Sie die gegebene Schleife in eine do while Schleife um.
2.3 (5 Punkte)Wandeln Sie die gegebene Schleife in eine for Schleife um.

Aufgabe 3: Array: Ziffernsummen (20 Punkte)

Gegeben sei ein eindimensionales int Array mit ganzen, positiven Zahlen.

Aufgabenstellung: Schreiben Sie eine **Methode**, die die **Ziffernsumme** der Elemente dieses Arrays **berechnet**. Das Array wird als Parameter an die Methode übergeben. Der Rückgabewert ist die errechnete Summe.

Bsp: Das Array beinhaltet die Elemente 13, 2, 15, 198, 7. Der Rückgabewert ist: 37

Punkteverteilung:
1. Algorithmische Richtigkeit. (15 Punkte)
2. Syntaktische Richtigkeit. (5 Punkte)

Aufgabe 5: Theoriefragen

(10 Punkte)

Frage	Antwort	Punkte

5.1 Die folgende Methode wird mit swap(x,y) aufgerufen (x und y sind int Variable). Was passiert dabei? Warum tauscht dieses Programm nicht die Inhalte der Variablen x und y?
 	
`static void swap (int a; int b)`
`{`
`  int h = a; a = b; b = h;`
`}`
		5

5.2 Was ist der **Konstruktor super**? Wie wird er eingesetzt? Welchen Zweck erfüllt er?		**5**

Aufgabe 4: Programmverstehen, Fehler Finden (20 Punkte)

Gegeben ist folgende Methode berechneZ:

```
public static int berechneZ (int[] a) {
  int z, m = a[0], i = 1;
  while (i < a.length && a[i] == m)
    i++;
  if (i == a.length) return m;
  if (a[i] > m)
    {z = m; m = a[i];}
  else z = a[i];
  i++;
  while (i < a.length) {
    if (a[i] > m) { z = m; m = a[i]; }
    else if (a[i] < m && a[i] > z) z = a[i];
    i++;
  }
  return z;
}
```

4.1 (10 Punkte) Was liefert der folgende Aufruf der Methode berechneZ, wenn Sie hier beispielhaft das Array a mit den Ziffern Ihrer eigenen Matrikelnummer initialisieren?

int a[] = {_,_,_,_,_,_,_}; // hier die Ziffern Ihrer Matrikelnummer eintragen

int result = berechneZ(a);

result = [ ]

4.2 (5 Punkte) Bei welchen Parameterwerten würde ein Aufruf von `berechneZ` zu einem Absturz zur Laufzeit (bzw. Ausnahme) führen?

4.3 (5 Punkte) Welche Wirkung hat die Methode `berechneZ` im Allgemeinen? Was liefert `berechneZ` bei beliebigem Eingangsparameter a?

3.3.'03 Gruppe A

Aufgabe 1: Bücherverwaltung (35 Punkte)

Gegeben sei eine einfach verkettete, sortierte Liste (Klasse **BuchBestand),** mit der die Bücher in einer Buchhandlung verwaltet werden. Jeder Knoten beschreibt ein **Buch:**

```
class Buch {
        int ISBN;
        String Titel;
        int Preis;
        int Jahr;
        int Stueck;
        Buch next;

        Buch(int ISBN, String Titel, int Preis, int Jahr, int Stueck, Buch n)
       { this.ISBN = ISBN; this.Titel = Titel; this.Preis = Preis;
         this.Jahr = Jahr; this.Stueck = Stueck; next = n; }
}
```

Die Liste ist nach der Buchnummer aufsteigend sortiert.

Aufgabenstellung: Implementieren Sie eine Methode „**Einfuegen**", die Bücher in die Liste sortiert einträgt. Wenn ein Buch mit einer bestimmten ISBN Nummer bereits in der Liste existiert, soll die Stückanzahl erhöht werden. Wenn dieses Buch noch nicht existiert, soll ein neuer Knoten eingetragen werden.

Punkteverteilung:
1. Algorithmische Richtigkeit. (30 Punkte)
2. Syntaktische Richtigkeit. (5 Punkte)

Aufgabe 2: Schleifen (15 Punkte)

Gegeben ist die folgende while-Schleife:

```
int i = 0; j = 0;

while (i < a.length() && j < b.length() && a.charAt(i) == b.charAt(j)) {
                        i++; j++;
        }

a und b sind Strings.
```

Aufgabenstellung:
2.1 (5 Punkte) Beschreiben Sie die Funktionsweise dieser Schleife. Welches Ergebnis liefert diese Schleife?
2.2 (5 Punkte) Wandeln Sie die gegebene Schleife in eine do while Schleife um.
2.3 (5 Punkte)Wandeln Sie die gegebene Schleife in eine for Schleife um.

Aufgabe 3: Array: Ziffernsummen (20 Punkte)

Gegeben sei ein eindimensionales int Array mit ganzen, positiven Zahlen.

Aufgabenstellung: Schreiben Sie eine **Methode**, die die **Ziffernsumme** der Elemente dieses Arrays **berechnet**. Das Array wird als Parameter an die Methode übergeben. Der Rückgabewert ist die errechnete Summe.

Bsp: Das Array beinhaltet die Elemente 13, 2, 15, 198, 7. Der Rückgabewert ist: 37

Punkteverteilung:
1. Algorithmische Richtigkeit. (15 Punkte)
2. Syntaktische Richtigkeit. (5 Punkte)

Aufgabe 5: Programm nachvollziehen (20 Punkte)

Gegeben ist untenstehendes Programm. Geben Sie bei der Eingabe in die Variablen a,b,c die letzten vier Stellen Ihrer Matrikelnummer ein (z.B. Matrikelnummer ist 1234567, nach der Eingabe haben die Variablen folgenden Wert: a = 4, b= 5, c=6, d=7). Führen Sie dieses Programm aus, und geben Sie an, welche Ausgaben das Programm liefert:

```
import eprog.*;

class O {
  public int a;
  public void m1 (int a) { this.a = a; }
}

class U extends O {
  public int a;
  public static int b;
  public void m1 (int x) { a = x; }
  public static int m2 (int x, int y) {
      b = x+++y;
     return x ;
  }
  public void m3 (int a) { b=a; a++; }
}

public class Test extends EprogIO {
  static int a=0, b=0;
  public static void main(String[] args) {
     O o = new O();
     U u1 = new U();
     U u2 = new U();
     O ou = u1;
     String s = " , ";
     int c=0, d=0;
      try {
            a = readInt();  // Tausenderstelle der Matrikelnummer
            b = readInt();  // Hunderterstelle der Matrikelnummer
            c = readInt();  // Zehnerstelle der Matrikelnummer
            d = readInt();  // Einerstelle der Matrikelnummer
      } catch (EprogException e) { }


     println("1: " + o.a + s + u1.a + s + u2.a + s + ou.a); // 1:  ,  ,  ,
     o.a = a; ((O)u1).m1(b);
     println("2: " + o.a + s + u1.a + s + u2.a + s + ou.a); // 2:  ,  ,  ,
     u2.a = U.m2(b,c);
     println("3: " + o.a + s + u1.a + s + u2.a + s + ou.a); // 3:  ,  ,  ,
     ((U)ou).a = u2.b; ou.a++;
     println("4: " + o.a + s + u1.a + s + u2.a + s + ou.a); // 4:  ,  ,  ,
     u1.m3(u1.b);
     println("5: " + o.a + s + u1.a + s + u2.a + s + ou.a); // 5:  ,  ,  ,
     ((O)u1).a = d;
     println("6: " + o.a + s + u1.a + s + u2.a + s + ou.a); // 6:  ,  ,  ,
  }
}
```

Aufgabe 4: Klassenerweiterung (20 Punkte)

Gegeben ist die folgende Klasse `Punkt`:

```
public class Punkt {
      private float x,y;
      public Punkt(float neu_x, float neu_y)   { x = neu_x; y = neu_y ;}
      public distanz_zu(Point dort) {
            float xdiff = x - dort.x;
            float ydiff = y - dort.y;
            return Math.sqrt(xdiff * xdiff + ydiff * ydiff)
      }
}
```

Erweitern Sie die Klasse `Punkt` zu einer Klasse `Punkt3D`, die zusätzlich eine Variable für die z-Koordinate des Punktes im Raum enthält. Erweitern Sie weiters die Methode `distanz_zu`, so dass der Abstand zu einem anderen Punkt im Raum berechnet werden kann, wobei Sie hier die Methode der Oberklasse Punkt aufrufen sollen.

Hinweise: Der Abstand d zweier Punkte *p1* und *p2* im Raum wird bei gegebenen Abstand d_{xy} in der x-y Ebene wie folgt berechnet: $d = sqrt((p2.z - p1.z)^2 + d_{xy}^2)$. Die Klasse `Punkt3D` benötigt einen Konstruktor.

Aufgabe 3: Exception Handling (15 Punkte)

Gegeben ist folgend Klasse:

```
import eprog.*;

public class Test

{ public static void main (String [] args)
    {
    double Zahl1;
    double Zahl2;
    double Ergebnis1;
    double Ergebnis2;

    Zahl1=EprogIO.readDouble();
    Zahl2=EprogIO.readDouble();
    Ergebnis1=Zahl1/Zahl2;
    Ergebnis2=Zahl2/Zahl1;

    EprogIO.print(Ergebnis1);
    EprogIO.print(" ");
    EprogIO.print(Ergebnis2);
    EprogIO.println("");
    }
}
```

Leider wurde von dem Programmierer die Ausnahmebehandlung völlig vergessen. Ergänzen Sie das Programm um diese. Fangen Sie alle möglichen Ausnahmen ab und generieren Sie eine entsprechende Fehlermeldung.

Aufgabe 2: Methoden erstellen (30 Punkte)

Routing gehört zu den wichtigsten Aufgaben im Internet. Dabei wird für eine Internetadresse (IP) festgelegt wie sie erreicht werden kann. Wir möchten ein stark vereinfachtes Beispiel für Routing implementieren. Grundsätzlich funktioniert Routing indem Routen wie folgt festgelegt werden:

Für eine IP-Zieladresse (destination) gibt es ein IP-Gateway (gateway).
Dadurch weiss der Router wohin er ein IP-Paket für ein bestimmtes Ziel schicken soll.
Alle diese Routen werden in einen RoutingTabelle eingetragen. Zusätzlich wird zu jeder Route eine Metrik (metric) gespeichert. Die Metrik ist ein ganzzahliger Wert der angibt, wie schnell ein Ziel über dieses Gateway erreicht werden kann. (Je kleiner die Metrik umso schneller/besser ist die Route)

Folgende Klassen sind gegeben:

```
public class IP
      {
            // Constructor
            public IP(String Address);
                  // Erstelle ein neues Objekt IP mit der Adresse,
                  // die in Address als String übergeben wird
            // Methoden
            public String showIP(); // liefert die IP-Adresse als String
      }
```

Die Klasse IP speichert einfach IP-Adressen.

```
class Route
      {
            // Instanz Variablen
            public IP destination;
            public IP gateway;
            public byte metric;
      }
```

Die Klasse Route speichert einfach Routen.

Erstellen sie die Klasse RoutingTab. Sie speichert eine Routing Tabelle mit bis zu hundert Einträgen.

Erstellen Sie folgende Methoden für die Klasse RoutingTab:

```
insertRoute(Route neueRoute);
```

Die Methode soll die neueRoute in die Tabelle eintragen. Allerdings muss überprüft werden ob es bereits einen Eintrag zu dem Ziel dieser Route gibt. Wenn nein, wird die Route einfach eingetragen. Wenn ja muss überprüft werden ob die Metrik der neuen Route kleiner ist als der Eintrag in der Routing Tabelle. Wenn auch das zutrifft wird die neue Route eingetragen und die alte Route mit dem selben Ziel verworfen. Sonst wird die neue Route verworfen.

25.6.2003	Theorietest aus **Einführung in das Programmieren (AU)** LVANr: 183.046		**EI 7 90 Min.**
MatrikelNr:	**Nachname:**	**Vorname:**	**Punkte:**
Kennzahl:			

Bitte tragen Sie **zuerst** Ihre **Matrikelnummer, Studienkennzahl, Nachnamen** und **Vornamen** in die dafür vorgesehenen Kästchen ein. Es sind alle schriftlichen Unterlagen erlaubt. Es sind **keine technischen Hilfsmittel** erlaubt. Während 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 **maximal 100** Punkte erreichen. Achten Sie auf die **Punkteverteilung der Aufgaben**.

Aufgabe 1: Ausdrücke (15 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)` F

c) `i+d` F

d) `f%f` ✓

e) `(d>>e)` F

f) `a<<<3` ✓

g) `"".length()` ✓

h) `g++` F