Übungen zur Informatik A

Hauptklausur
20.02.2003

Universität Koblenz-Landau
Institut für Informatik
WS 2002/3
Prof. Dr. Dietrich Paulus
Dr. Manfred Jackel

- ☐ Bitte lösen Sie jede Aufgabe auf dem jeweiligen Blatt bzw. auf dessen Rückseite. Wenn Sie zusätzliche Blätter benötigen, dann vermerken Sie Ihren Namen auf jedem einzelnen Blatt.
- ☐ Jede Aufgabe wird mit 12 Punkten bewertet, die sich gleichmäßig auf die Teilaufgaben verteilen.
- ☐ Lesen Sie erst alle Aufgaben durch. Beginnen Sie mit der für Sie einfachsten.

A

Name: .....

Vorname: .....

Matrikelnummer: ..... **E-Mail:** .....

Übungsgruppe: .....

Studiengang: ☐ **Informatik** ☐ **Computervisualistik** ☐ .....

Aufgabe:	1	2	3	4	5	6
Punkte:						

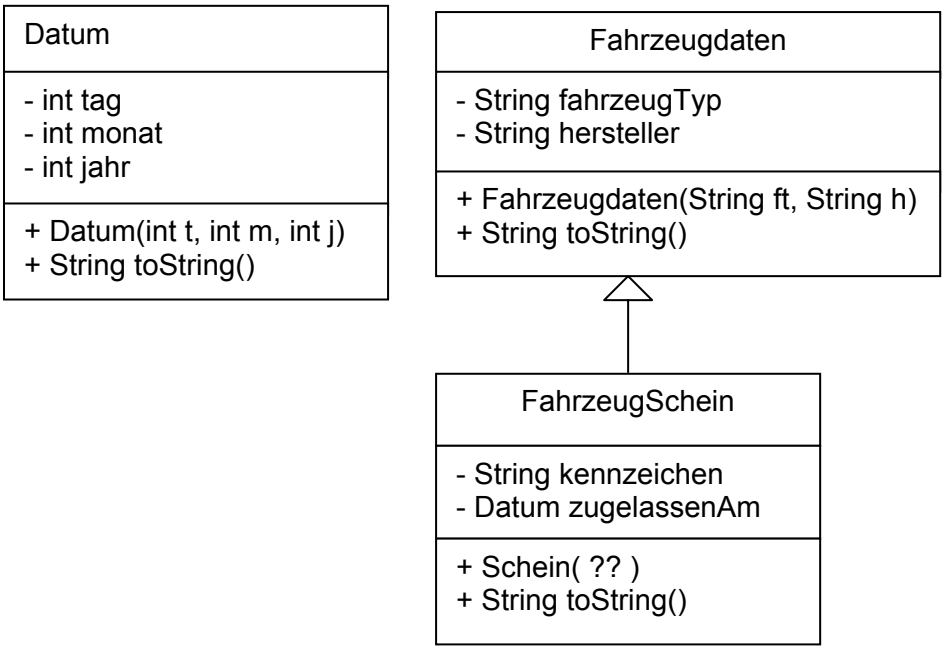
Gesamtpunktzahl: .....

Viel Erfolg !

Hauptklausur zur Informatik A WS 2002/3

Aufgabe A1

Hier sehen Sie ein UML-Klassendiagramm mit 3 Klassen.



a) Geben Sie Javacode für diese Klassen an. Alle Klassen sollen in eigenen Dateien liegen.
b) Vervollständigen Sie die Signatur des Konstruktors FahrzeugSchein. Er soll alle Attribute der Klasse und der Basisklasse initialisieren.
c) Programmieren Sie alle Konstruktoren aus. Denken Sie an den „richtigen“ Aufruf des Superklassen-Konstruktors.
d) Schreiben Sie eine Hauptklasse Probeklausur mit einer main-Methode, die ein FahrzeugSchein-Objekt erzeugt und dessen Inhalt anzeigt, Typ Golf 4, Hersteller VW, Kennzeichen KO-PF 444, zugelassen am 20.2.2003.

Hauptklausur zur Informatik A WS 2002/3

Aufgabe A2

Gegeben sei folgende Syntax in EBNF:

<Qualifier> ::= <Bezeichner> { <Punkt> <Bezeichner> }
<Bezeichner> ::= <Buchstabe> { <Buchstabe> | <Strich> | <Ziffer> }
<Buchstabe> ::= ’a’ | ’b’ | ’c’ | ... | ’x’ | ’y’ | ’z’
<Strich> ::= ’_’
<Punkt> ::= ’.’
<Ziffer> ::= ’0’ | ’1’ | ... | ’8’ | ’9 ’

a) Prüfen Sie, ob folgende Worte aus dem Startsymbol <Qualifier> ableitbar sind oder nicht. Geben Sie entweder den Ableitungsbaum an, oder ein Gegenargument, falls sich das Wort nicht ableiten lässt.

(1.) _abc123

(2.) koblenz.o.lau

(3.) abc123

b) Geben Sie Syntaxdiagramme für die Nonterminale <Qualifier> und <Bezeichner> an.

Hauptklausur zur Informatik A WS 2002/3

Aufgabe A3

Gegeben ist folgende (unfertige) Klasse Matrix, die quadratische Matrizen darstellen soll. Ein geeigneter Konstruktor, der hier nicht angegeben ist, möge die Werte der Matrixfelder im Attribut a ablegen.

a) Programmieren Sie die Methode max(), die den Wert des größten Elementes in der Matrix ermittelt.
b) Programmieren Sie die Methode diagonale(), die die Diagonale der Matrix liefert.

```
class Matrix
{ int[][] a;

  public int max() {
  // bestimmen Sie das größte Element der Matrix.









  }

  public int[] diagonale(){
  //hier ihr Code für die Diagonale









  }
}
```

Hauptklausur zur Informatik A WS 2002/3

Aufgabe A4

Polynome $f(x) = a_0 x^n + ... + a_{n-1} x + a_n$ wertet man am besten nach dem Horner-Schema aus:

$f(x) = (...(a_0 x + a_1)x + ... + a_{n-1})x + a_n$.

Vervollständigen Sie nachfolgendes Programmstück durch die Methode pWert, die nach dem Hornerschema das Polynom mit den Koeffizienten

```
double[] a = new double[n+1]
```

an der Stelle x auswertet.

```
class Polynom {
  private double[] a;

  public Polynom(double[] a) {
    this.a = a;
  }

  double pWert(double x) {









  }
}
```

Hauptklausur zur Informatik A WS 2002/3

Aufgabe A5

Folgendes Programm berechnet rekursiv die Fibonacci-Zahlen:

```
static long fib(int n) {
  if (n<=1) return 1;
  return fib(n-1)+fib(n-2),
}

...
public static main(String[] args) {
  long r=fib(3);
}
...
```

Skizzieren Sie den Stack bei der Berechnung von fib(3), wenn zum ersten Mal fib(1) aufgerufen wird!

Hauptklausur zur Informatik A WS 2002/3

Aufgabe A6a

Gegeben sei folgende Klasse **Bruch**, die Brüche „normiert“ darstellt: In der internen Speicherung sind die Brüche vollständig gekürzt, der Nenner ist stets positiv, die Null wird eindeutig als 0/1 dargestellt. Die Klasse Bruch implementiert das Interface Comparable.

a) Implementieren Sie die Methode **public int compareTo(Object b)**.

```
class Bruch implements Comparable{

  private int zaehler, nenner; // normierte Darstellung hier!

  private int ggT(int a, int b) {
    if (b == 0) return a;
    else return ggT(b, a % b);
  }

  public Bruch (int zaehler, int nenner)
    throws ArithmeticException
  {
    if (nenner==0)
      throw new ArithmeticException("Nenner ist 0");
    if (nenner<0) {
      nenner = -nenner; zaehler = -zaehler;
    }

    if (zaehler==0) {
      this.zaehler=0; this.nenner=1;
    } else {
      int k = ggT(Math.abs(zaehler), nenner);
      if (k==1) {
        this.zaehler=zaehler;
        this.nenner=nenner;
      } else {
        this.zaehler=zaehler/k;
        this.nenner=nenner/k;
      }
    }
  }

  int getZaehler() { return zaehler; }

  int getNenner() { return nenner; }
```

Aufgabe A6a (Fortsetzung)

```
  public int compareTo(Object b) {
    // -1 falls aktuelles Objekt kleiner b
    // 0 für gleich
    // 1 sonst
    // hierhin mit Ihrem Code.









  }
}
```

Aufgabe A6.b

In der Klasse **Bruchrechnung** sollen die Operationen Addition und Multiplikation zweier Brüche implementiert werden. die Methode **mult** haben wir schon ausprogrammiert.

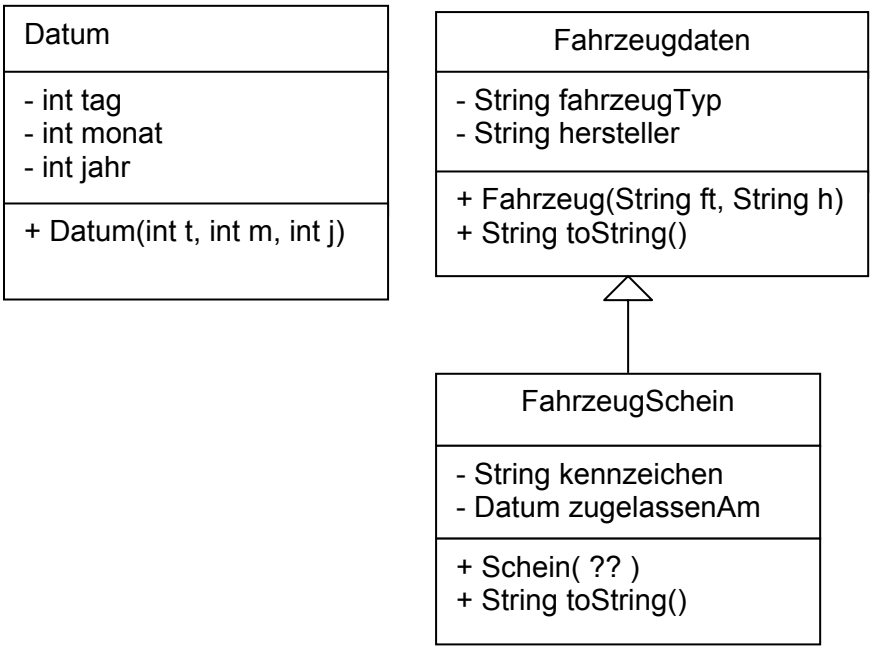
b) Programmieren Sie die Methode **add** aus.

```
class Bruchrechnung {

  public static Bruch add(Bruch a, Bruch b)
    throws ArithmeticException
  { // Platz für Ihre Lösung:









  }

  public static Bruch mult(Bruch a, Bruch b)
    throws ArithmeticException
  { return new Bruch(a.getZaehler()*b.getZaehler(),
                     a.getNenner()*b.getNenner());

  // Ein Beispiel, wie man diese Methoden anwenden kann:
  public static void main(String[] args)throws Exception {
  Bruch a=new Bruch(1,2);
  Bruch b=new Bruch(1,4);
  Bruch c=new Bruch(4,3);
  if (mult(add(a,b),c).compareTo(new Bruch(1,1))==0)
    System.out.println("Richtig.");
 }
}
```

Hauptklausur zur Informatik A WS 2002/3

Aufgabe B3

Hier sehen Sie ein UML-Klassendiagramm mit 3 Klassen.



e) Geben Sie Javacode für diese Klassen an. Alle Klassen sollen in eigenen Dateien liegen.

f) Vervollständigen Sie die Signatur des Konstruktors FahrzeugSchein. Er soll alle Attribute der Klasse und der Basisklasse initialisieren.

g) Programmieren Sie alle Konstruktoren aus. Denken Sie an den „richtigen“ Aufruf des Superklassen-Konstruktors.

h) Schreiben Sie eine Hauptklasse Probeklausur mit einer main-Methode, die ein FahrzeugSchein-Objekt erzeugt und dessen Inhalt anzeigt, Typ Golf 4, Hersteller VW, Kennzeichen KO-PF 444, zugelassen am 20.2.2003.

Hauptklausur zur Informatik A WS 2002/3

Aufgabe B4

Gegeben sei folgende Syntax in EBNF:

<Qualifier> ::= <Bezeichner> { <Punkt> <Bezeichner> }
<Bezeichner> ::= <Buchstabe> { <Buchstabe> | <Strich> | <Ziffer> }
<Buchstabe> ::= ’a’ | ’b’ | ’c’ | ... | ’x’ | ’y’ | ’z’
<Strich> ::= ’_’
<Punkt> ::= ’.’
<Ziffer> ::= ’0’ | ’1’ | ... | ’8’ | ’9 ’

a) Prüfen Sie, ob folgende Worte aus dem Startsymbol <Qualifier> ableitbar sind oder nicht. Geben Sie entweder den Ableitungsbaum an, oder ein Gegenargument, falls sich das Wort nicht ableiten lässt.

(1.) _abc123

(2.) koblenz.o.lau

(3.) abc123

b) Geben Sie Syntaxdiagramme für die Nonterminale <Qualifier> und <Bezeichner> an.

Hauptklausur zur Informatik A WS 2002/3

Aufgabe B5

Gegeben ist folgende (unfertige) Klasse Matrix, die quadratische Matrizen darstellen soll. Ein geeigneter Konstruktor, der hier nicht angegeben ist, möge die Werte der Matrixfelder im Attribut a ablegen.

a) Programmieren Sie die Methode max(), die den Wert des kleinsten Elementes in der Matrix ermittelt.
b) Programmieren Sie die Methode diagonale(), die die Diagonale der Matrix liefert.

```
class Matrix
{ int[][] a;

  public int min() {
  // bestimmen Sie das kleinste Element der Matrix.









  }

  public int[] diagonale(){
  //hier ihr Code für die Diagonale









  }
}
```

Hauptklausur zur Informatik A WS 2002/3

Aufgabe B6

Polynome $f(x) = a_0 x^n + ... + a_{n-1} x + a_n$ wertet man am besten nach dem Horner-Schema aus:

$f(x) = (...(a_0 x + a_1)x + ... + a_{n-1})x + a_n$.

Vervollständigen Sie nachfolgendes Programmstück durch die Methode pWert, die nach dem Hornerschema das Polynom mit den Koeffizienten

```
double[] a = new double[n+1]
```

an der Stelle x auswertet.

```
class Polynom {
  private double[] a;

  public Polynom(double[] a) {
    this.a = a;
  }

  double pWert(double x) {









  }
}
```

Hauptklausur zur Informatik A WS 2002/3

Aufgabe B1

Folgendes Programm berechnet rekursiv die Fibonacci-Zahlen:

```
static long fib(int n) {
  if (n<=1) return 1;
  return fib(n-1)+fib(n-2),
}

...
public static main(String[] args) {
  long r=fib(3);
}
...
```

Skizzieren Sie den Stack bei der Berechnung von fib(3), wenn zum ersten Mal fib(1) aufgerufen wird!

Hauptklausur zur Informatik A WS 2002/3

Aufgabe B2.a

Gegeben sei folgende Klasse **Bruch**, die Brüche „normiert“ darstellt: In der internen Speicherung sind die Brüche vollständig gekürzt, der Nenner ist stets positiv, die Null wird eindeutig als 0/1 dargestellt. Die Klasse Bruch implementiert das Interface Comparable.

a) Implementieren Sie die Methode **public int compareTo(Object b)**.

```
class Bruch implements Comparable{

  private int zaehler, nenner; // normierte Darstellung hier!

  private int ggT(int a, int b) {
    if (b == 0) return a;
    else return ggT(b, a % b);
  }

  public Bruch (int zaehler, int nenner)
    throws ArithmeticException
  {
    if (nenner==0)
      throw new ArithmeticException("Nenner ist 0");
    if (nenner<0) {
      nenner = -nenner; zaehler = -zaehler;
    }

    if (zaehler==0) {
      this.zaehler=0; this.nenner=1;
    } else {
      int k = ggT(Math.abs(zaehler), nenner);
      if (k==1) {
        this.zaehler=zaehler;
        this.nenner=nenner;
      } else {
        this.zaehler=zaehler/k;
        this.nenner=nenner/k;
      }
    }
  }

  int getZaehler() { return zaehler; }

  int getNenner() { return nenner; }
```

Aufgabe B2.a (Fortsetzung)

```
  public int compareTo(Object b) {
    // -1 falls aktuelles Objekt kleiner b
    // 0 für gleich
    // 1 sonst
    // hierhin mit Ihrem Code.









  }
}
```

Aufgabe B2.b

In der Klasse **Bruchrechnung** sollen die Operationen Subtraktion und Multiplikation zweier Brüche implementiert werden. die Methode **mult** haben wir schon ausprogrammiert.

b) Programmieren Sie die Methode **add** aus,

```
class Bruchrechnung {

  public static Bruch sub(Bruch a, Bruch b)
    throws ArithmeticException
  { // Platz für Ihre Lösung:
```

```
  }

  public static Bruch mult(Bruch a, Bruch b)
    throws ArithmeticException
  { return new Bruch(a.getZaehler()*b.getZaehler(),
                     a.getNenner()*b.getNenner());

  // Ein Beispiel, wie man diese Methoden anwenden kann:
  public static void main(String[] args)throws Exception {
  Bruch a=new Bruch(1,1);
  Bruch b=new Bruch(1,4);
  Bruch c=new Bruch(4,3);
  if (mult(sub(a,b),c).compareTo(new Bruch(1,1))==0)
    System.out.println("Richtig.");
 }
}
```