Matrikelnummer:	*Punkte:*

Klausur zur Vorlesung

Grundlagen der Informatik und Numerik

Dr. Monika Meiler
Mo 10.02.2014, Beginn: 09.15 Uhr, Ende 10.45 Uhr

Bemerkungen:

- **Jedes Blatt ist mit der Matrikelnummer zu versehen.**
- **Jede Aufgabe ist auf dem vorgesehenen Blatt zu lösen. Reicht der dortige Platz nicht aus, so verwenden Sie ein mit der Matrikelnummer versehenes zusätzliches Blatt.**
- **Es sind außer Papier und Schreibzeug *keine* weiteren Hilfsmittel erlaubt (keine Taschenrechner, keine Unterlagen, . . .).**
- **Es ist leserlich und *nicht* mit Bleistift zu schreiben.**
- **Beantworten Sie Fragen pro Pfeil „➢“ mit *genau* einem Sachverhalt.**

Schalten Sie Ihr Handy aus!

Maximum: 66 Punkte
Note 1: 60 Punkte Note 2: 50 Punkte Note 3: 40 Punkte Note 4: 30 Punkte

Notizen

Matrikelnummer:	*Punkte:*

Klausuraufgabe 1 (22 Punkte)

(a) Eine EDV-Anlage arbeitet nach dem EVA-Prinzip. Erklären Sie das Prinzip jeweils in einem Satz. **2P**

- **E ...**
- **V ...**
- **A ...**

(b) Welche Beziehung besteht zwischen einem Algorithmus und einem Programm? **2P**

➢

(c) Geben Sie die Additions- und Multiplikationstabelle zum Rechnen im Positionssystem zur Basis $b = 3$ an. **4P**

***\+**	**0**	**1**	**2**
0	\		
1		\	
2			\

(d) Berechnen Sie in diesem System mit Zwischenergebnissen und machen Sie die Probe im Dezimalsystem: **8P**

$(122)_3$ Probe:

$+ (12)_3$

Übertrag _____

_____

$(122)_3 * (12)_3$ Probe:

_____

Übertrag _____

_____

(e) Wozu sind in einem Programm Datentypangaben notwendig? **2P**

➢

➢

(f) Welche (mathematische) Größen werden in den folgenden Variablen gespeichert? **4P**

Variable	**(mathematische) Größe**
`int i;`	**`i`** ist eine Variable für
`String str;`	**`str`** ist eine Variable für
`double[] v;`	**`v`** ist eine Variable für
`boolean b;`	**`b`** ist eine Variable für

Matrikelnummer:	*Punkte:*

Klausuraufgabe 2 (22 Punkte)

Berechnen Sie schrittweise mittels der Trapezregel und mittels der Simpsonregel das Flächenintegral $F = \int_a^b f(x)dx = \int_{-1}^{3} \left((1-x)^2 - 4\right)dx$:

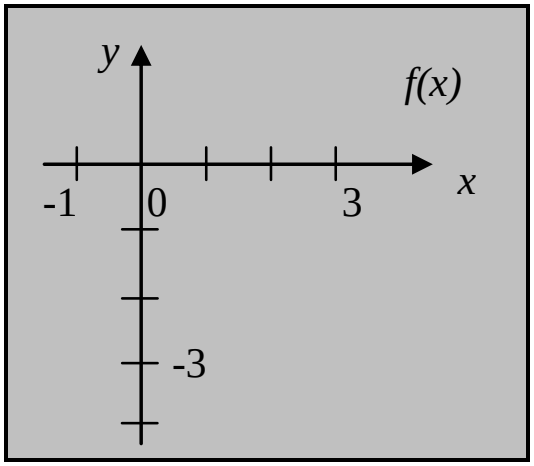
(a) Unterteilen Sie das Intervall $[-1,3]$ in 4 Teilintervalle. Berechnen Sie die Funktionswerte an den 5 Stützstellen. Verwenden Sie für die Berechnung von $f(2)$ und $f(3)$ das Hornerschema. **7P**

$f(-1) =$ $f(0) =$ $f(1) =$

$f(2) =$

$f(3) =$

(b) Skizzieren Sie die Funktion $f(x)$ und markieren Sie die zu berechnende Fläche. **3P**



(c) Berechnen Sie die Fläche F mittels der Trapezregel und mittels der Simpsonregel. **6P**

- Trapezregel: $F \approx$

- Simpsonregel: $F \approx$

(d) Berechnen Sie den mathematischen Wert. **2P**

- Mathematischer Wert: $F = \int_{-1}^{3} \left((1-x)^2 - 4 \right) dx =$

(e) Vergleichen Sie die Endergebnisse mit dem mathematischen Wert und beantworten Sie für beide Verfahren die folgende Frage: Wie viele Stützstellen sind für dieses Beispiel notwendig, um eine korrekte Lösung zu erhalten? Begründen Sie Ihre Vermutung. **4P**

- Trapezregel:

- Simpsonregel:

Matrikelnummer:	*Punkte:*

Klausuraufgabe 3 (22 Punkte)

Die Menge aller rationalen Zahlen Q lassen sich als Menge aller gemeinen (gewöhnlichen) Brüche $Q = \left\{ \frac{p}{q} \middle| p, q \in Z \wedge q \neq 0 \right\}$ darstellen. Gegeben sei das folgende Klassendiagramm für solch eine Klasse **`Bruch`**, bestehend aus Zähler **`zaehler`** und Nenner **`nenner`**, einer Methode **`add`** zur Addition eines Bruches mit einem anderen und einer Methode **`kuerzen`** zum Kürzen eines Bruches. Die Klasse soll in Java implementiert werden.

Bruch	
- zaehler:	int
- nenner:	int
+ setBruch(int, int):	boolean
+ toString():	String
+ add(Bruch):	Bruch
+ getZaehler():	int
+ getNenner():	int
- kuerzen():	void

(a) Nehmen Sie ein Attribut für den Zähler **`zaehler`** und ein Attribut für den Nenner **`nenner`** in die Klasse **`Bruch`** auf. **2P**

Bruch.java

```
public class Bruch
{
/**
 * Zaehler eines Bruchs.
 */



/**
 * Nenner eines Bruchs.
 */
```

(b) Schreiben Sie eine Methode **`toString`** so, dass ein Bruch $\frac{p}{q}$ in der linearen Form p / q zurückgegeben wird. **2P**

```
/**
 * Darstellung eines Bruchs.
 * @return Bruch in linearer Schreibweise
 */
  public String toString()
  {




  }
```

(c) Gegeben sei die folgende Methode **`setBruch`** mit **`z`** als Zähler und **`n`** als Nenner eines Bruchs. Tragen Sie einen **`javadoc`**-Kommentar für diese Methode ein. **2P**

```
/**
 *

 * @param

 * @param

 * @return

 */
  public boolean setBruch( int z, int n)
  {
    if( n == 0) return false;

    zaehler = z;
    nenner = n;
    kuerzen();

    return true;
  }
```

(d) Gegeben sei die folgende Methode **`kuerzen`**. Analysieren Sie diese Methode, indem Sie alle Speicherplatzveränderungen für den Bruch $\frac{10}{6}$ protokollieren. **4 P**

```
/**
 * Kuerzen eines Bruchs.
 */
  private void kuerzen()
  {
    int m = Math.abs( zaehler);
    int n = Math.abs( nenner);

    int r;
    do
    {
      r = m % n;
      m = n;
      n = r;
    } while( n != 0);

    if( m > 1)
    {
      zaehler /= m;
      nenner /= m;
    }
  }
```

Protokoll:

zaehler	nenner	m	n	r
10	6			

Matrikelnummer:	*Punkte:*

(e) Programmieren Sie die Methode **`add`** für die Addition eines Bruchs mit einem anderen. **4P**

```
/**
 * Addition.
 * @param a Bruch als zweiter Operand
 * @return Summe des Bruchs mit dem Bruch a
 */
  public Bruch add( Bruch a)
  {









  }
```

(f) In einem Testprogramm seien vier Brüche $a,b,c,d \in Q$ wie folgt vereinbart: **4P**

```
  public static void main( String[] args)
  {
    Bruch a = new Bruch();
    Bruch b = new Bruch();
    Bruch c = new Bruch();
    Bruch d = new Bruch();

    a.setBruch( 2, 3);
    b.setBruch( 3, 18);
```

Geben Sie je eine Java-Anweisung zur Berechnung von $c = a + b$ und $d = 2 * c$ an.

```
// Beispiel c = a + b


// Beispiel d = 2 * c
```

(g) Geben Sie als Kommentare die Konsolenausgaben für *c* und *d* unter Verwendung von (b) an. **4P**

```
    System.out.println( "c = " + c); //


    System.out.println( "d = " + d); //


  }
}
```