OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG | INF | FAKULTÄT FÜR INFORMATIK

Magdeburg, den 12.09.2024

Klausur zur Vorlesung
Programmierparadigmen
(Sommer 2024)

Nachname: _____

Vorname: _____

Matrikelnummer: _____

Beachten Sie folgende Hinweise:

- Die Dauer der Klausur ist **120 Minuten**. Ein Punkt entspricht damit in etwa 90 Sekunden der Klausurzeit.
- Legen Sie Ihren Studierendenausweis oder Ihren Personalausweis sichtbar vor Ihren Platz. Die Prüfungsbetreuer kontrollieren diese während der Prüfung.
- Die Klausur besteht aus 18 Blättern. Prüfen Sie vor Beginn, ob alle Blätter vorhanden sind.
- Beschriften Sie **jedes** abgegebene Blatt mit Ihren Namen und Ihrer Matrikelnummer.
- Lesen Sie vor Beginn gewissenhaft die Aufgabenstellung. Ihre Lösung schreiben Sie unter die Aufgabenstellung in das vorgesehene Kästchen. Sollte dieser Platz nicht reichen, nutzen Sie die Rückseite bzw. erfragen Sie weitere Blätter bei den Betreuern. Lösungen auf anderen Blättern, als die durch die Betreuer ausgegeben, werden **nicht gewertet**. Nutzen Sie für jede Aufgabe (nicht Unteraufgaben) ein separates Blatt und machen Sie die Zuordnung zu einer Aufgabe deutlich.
- Streichen Sie falsche Lösungen **deutlich** durch.
- Benutzen Sie keinen Bleistift oder Stifte mit roter oder grüner Farbe!
- Erlaubte Hilfsmittel sind nur Stifte und ein Lineal.
- Werden zu einer Aufgabe mehrere Lösungen angegeben oder ist die Lösung nicht eindeutig oder wird bei einer geforderten Begründung zu einer Lösung diese nicht angegeben, so gilt die Aufgabe als nicht gelöst.
- Bei Fragen geben Sie ein sichtbares Handzeichen. Ein Betreuer wird dann zu Ihnen herantreten.
- Das Verlassen des Raumes ist nur nach Absprache und nur einzeln gestattet.

Viel Erfolg ! (Don't Panic ☺)

Aufgabe	1	2	3	4	5	6	$\sum$
maximale Punkte	15	21	17	11	13	3	80
erreichte Punkte							

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG | INF | FAKULTÄT FÜR INFORMATIK

Magdeburg, den 12.09.2024

Aufgabe 1 : *Prozedurales Paradigma*

a) *Was versteht man unter der Fragmentierung des Speichers? Warum stellt dies ein Problem dar und was ist die Ursache? Welche Lösung bietet sich an? [3 Punkte]*

b) *Beschreiben Sie die Funktionsweise des Garbage Collectors „Reference Counting"! Wann würden Sie die Anwendung dieses Garbage Collectors empfehlen bzw. nicht empfehlen? [3 Punkte]*

Magdeburg, den 12.09.2024

c) *Zeichnen Sie zu nachfolgendem C-Quellcode den Zustand des Stacks, unter der Annahme, dass das Programm über die main-Methode in Zeile 23 gestartet wurde und gerade die Anweisung in Zeile 19 ausgeführt hat. Gehen Sie dabei davon aus, dass seitens des Compilers keine Optimierungen stattgefunden haben. [6 Punkte]*

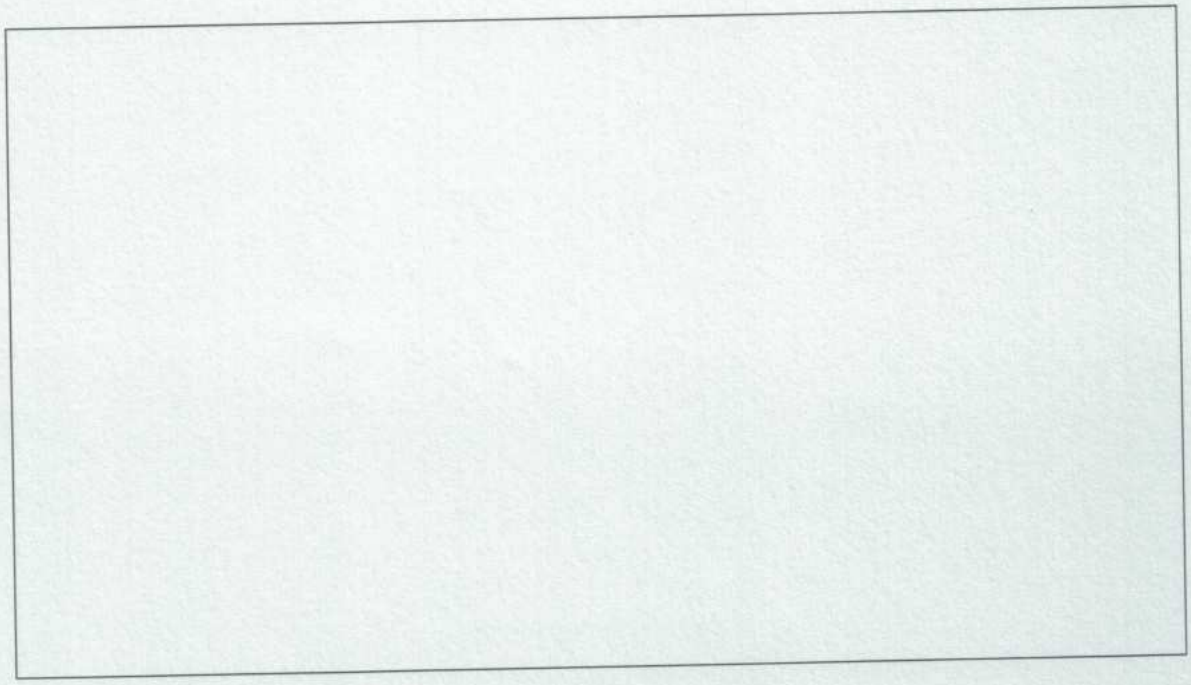
```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int isGreaterFive(int number){
5      if(number>5){
6          return 1;
7      }
8      return 0;
9  }
10
11 int isLowerSeven(int number){
12     if(number<7){
13         return 1;
14     }
15     return 0;
16 }
17
18 int checkTwo(int checkFirst, int checkSecond){
19     int resCheck = checkFirst*checkSecond;
20     return resCheck;
21 }
22
23 int main(){
24     int num = 6;
25     int result = checkTwo(isGreaterFive(num),isLowerSeven(num));
26     return 0;
27 }
```

Magdeburg, den 12.09.2024

d *) Im Jahr 2014 wurde der Heartbleed-Bug, eine Schwachstelle in der OpenSSL Verschlüsselung, bekannt. Weltweit waren etwa 17% aller SSL-verschlüsselten Webseiten davon betroffen. Nachfolgender Code zeigt eine vereinfachte Version des Heartbleed-Exploits. Erklären Sie, an welcher Stelle das Problem auftritt, welche Möglichkeiten für den Angreifer bestehen und wie das Problem beseitigt werden kann.* ***Hinweis:*** *Zur Hilfestellung wurden erläuternde Quellcodekommentare eingefügt. [3 Punkte]*

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// struct fuer eine Herzschlag-Nachricht
struct HeartbeatMessage
{
    int length; // Laenge der Nachricht
    char *data; // Nachricht als String
};
int main(){
    //-----------------------------------------------------------
    // Angreifer erstellt Heartbeat-Message
    // auf dem Angreifer-PC
    struct HeartbeatMessage msg;
    msg.length = 65536;
    char arr[1] = {'x'};
    msg.data = arr;

    // Angreifer sendet Nachricht msg an Server
    //-----------------------------------------------------------
    // Server erstellt Antwort mit selben Inhalt
    // auf dem Server
    struct HeartbeatMessage responseMsg;
    responseMsg.length = msg.length;
    responseMsg.data = malloc(msg.length*sizeof(char));
    // memcpy(ziel, quelle, l)
    // kopiert einen Speicherbereich der Laenge l von quelle nach ziel
    memcpy(responseMsg.data, msg.data, msg.length*sizeof(char));

    // Server sendet Antwort responseMsg an Angreifer zurueck
    //-----------------------------------------------------------

    // Angreifer liest Antwort auf dem Angreifer PC
    printf("%s\n",responseMsg.data);
}
```

Magdeburg, den 12.09.2024



/ 15

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG | INF | FAKULTÄT FÜR INFORMATIK

Magdeburg, den 12.09.2024

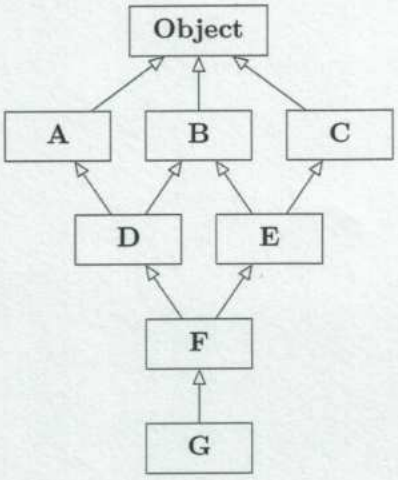
Aufgabe 2 : *Objektorientiertes Paradigma*

a) *Was versteht man in der Objektorientierung unter Überladen, Überdecken und Überschreiben. Geben Sie je ein Beispiel in JAVA an. [6 Punkte]*

b) *Was unterscheidet eine statische von einer nicht-statischen Methode in einer Java-Klasse? [2 Punkte]*

Magdeburg, den 12.09.2024

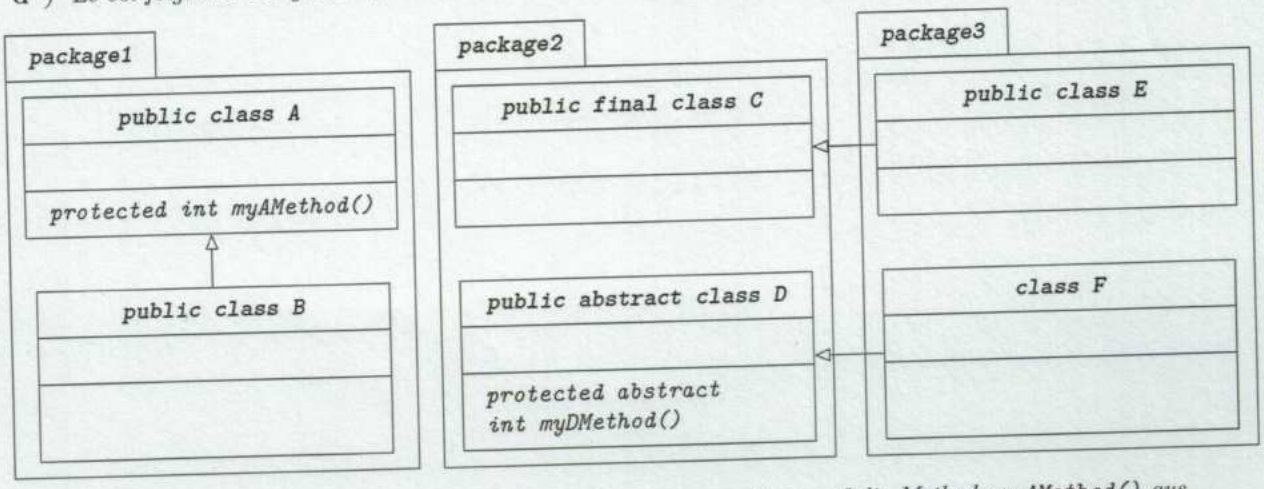
c *) Gegeben sei folgende Vererbungsstruktur. Führen Sie die Linearisierung mit der lokalen Vererbungsreihenfolge gemäß den drei Regeln aus der Vorlesung für die Klasse G durch. Geben Sie die von Ihnen ermittelte Reihenfolge der Klassen als auch die für jede Klasse jeweils angewendete Regel an. [5 Punkte]*




```
class A {...}
class B
class C
class D extends A, B{...}
class E extends C, B{...}
class F extends D, E{...}
class G extends F{...}
```

Magdeburg, den 12.09.2024

d) *Es sei folgendes Diagramm für eine Java-Klassenhierarchie gegeben:*



a) Geben Sie für **jede** *Klasse in diesem Klassendiagramm an, ob sie auf die Methode* `myAMethod()` *aus Klasse* `A` *bzw. auf die Methode* `myDMethod()` *der Klasse* `D` *zugreifen kann.*

Für jedes richtig gesetzte Kreuz gibt es einen halben Punkt, jedes falsch gesetzte Kreuz gibt einen halben Punkt Abzug. Sie bekommen für diesen Aufgabenteil mindestens 0 Punkte.

	`myAMethod`		`myDMethod`	
Klasse	*ja*	*nein*	*ja*	*nein*
A	☐	☐	☐	☐
B	☐	☐	☐	☐
C	☐	☐	☐	☐
D	☐	☐	☐	☐
E	☐	☐	☐	☐
F	☐	☐	☐	☐

b) Die Methode `myDMethod()` *in Klasse* `D` *ist als* `abstract` *gekennzeichnet. Welche Folgen hat dies für die übrigen Klassen in dem Klassendiagramm?*

c) Das Klassendiagramm enthält einen Fehler, der die Hierarchie so nicht umsetzbar macht. Welchen?

[8 Punkte]

/ 21

Magdeburg, den 12.09.2024

Aufgabe 3 *: Funktionales Paradigma*

a *) Ein häufiges Mittel im funktionalen Paradigma ist das Pattern Matching.*

1. *Wann ist ein Pattern-Matching disjunkt und wann vollständig/exhaustive?*
2. *Entscheiden Sie, ob die nachfolgenden Pattern Matching Ausdrücke jeweils disjunkt und/oder vollständig (oder nichts von beidem) sind.*

   Für jedes richtig gesetzte Kreuz gibt es einen halben Punkt, jedes falsch gesetzte Kreuz gibt einen halben Punkt Abzug. Sie bekommen für diesen Aufgabenteil mindestens 0 Punkte.

[6 Punkte]

```
1  def func1(n: Int): Int = n match {
2      case n if n>100 => n-10
3      case n => func1(func1(n+11))
4    }
5
6  def func2(arr: List[Int]): List[Int] = arr match {
7    case Nil => Nil
8    case x::y::xs => func2(xs)++List(y)++List(x)
9  }
10
11 def func3(arr: List[Int]): Int = arr match {
12     case Nil => 0
13     case x::xs if(x < func3(xs)) => x
14     case x::xs if(x >= func3(xs)) => func3(xs)
15 }
16
17 def func4(x: Int): String = x match {
18   case 1 => "one"
19   case 2 => "two"
20   case _ => "many"
21 }
22
23 def func5(x: Int,y: String): String = (x, y) match {
24   case (1, "one") => "one"
25   case (2, "two") => "four"
26   case (n, "more") => "more-more"
27 }
```

Methode	disjunkt ja	disjunkt nein	vollständig ja	vollständig nein
func1	☐	☐	☐	☐
func2	☐	☐	☐	☐
func3	☐	☐	☐	☐
func4	☐	☐	☐	☐
func5	☐	☐	☐	☐

Magdeburg, den 12.09.2024

b *) Was sind Funktionen höherer Ordnung? Wie hängen Unterversorgung und Currying mit diesen Funktionen zusammen? Nennen und erklären Sie ein praktisches Anwendungsbeispiel für eine Funktion höherer Ordnung! [4 Punkte]*

c *) Im funktionalen Paradigma kann man Konstanten mit Hilfe nullstelliger Funktionen darstellen, die jeweils einen konstanten Wert zurückgeben. Wie würden Sie das Prinzip der Variablen darauf aufbauen?* ***Hinweis:*** *Überlegen Sie sich im Vorfeld, was für das Konzept der Variablen notwendig ist. [2 Punkte]*

Magdeburg, den 12.09.2024

d) *Entscheiden Sie, ob die nachfolgenden Scala-Beispiele jeweils rekursiv, linear-rekursiv und/oder endrekursiv sind!*
Für jede komplett richtig ausgefüllte Zeile erhalten Sie einen Punkt. Eine falsch ausgefüllte Zeile gibt 0 Punkte (keinen Abzug!).
[5 Punkte]

```
1  def func1(n : Int): Int = n match {
2      case n if n>100 => n-10
3      case n => func1(func1(n+11))
4  }
5
6  def func2(arr: List[Int]): List[Int] = arr match {
7    case Nil => Nil
8    case x::xs => func2(xs)++List(x)
9  }
10
11 def func3(arr: List[Int]): Int = arr match {
12     case Nil => 0
13     case x::xs if(x<func3(xs)) => x
14     case x::xs if(x>=func3(xs)) => func3(xs)
15 }
16
17 def func4(arr: List[Int],arr2: List[Int], n: Int): List[Int] = arr match {
18     case Nil => arr2
19     case x::xs if x>n => func4(xs,arr2++List(x),n)
20     case x::xs if x<=n => func4(xs,arr2,n)
21   }
22
23 def func5(n:Int): Int = n match {
24     case 0 => 0
25     case n if n>0 => func5(n-1)-n
26     case n if n<0 => func5(n-1)+n
27 }
```

Methode	rekursiv	linear rekursiv	endrekursiv
func1	☐	☐	☐
func2	☐	☐	☐
func3	☐	☐	☐
func4	☐	☐	☐
func5	☐	☐	☐

/ 17

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG | INF | FAKULTÄT FÜR INFORMATIK

Magdeburg, den 12.09.2024

Aufgabe 4 *: Logisches Paradigma*

a *) Was versteht man unter dem Begriff Unifikation? Definieren Sie wann zwei Terme unifizierbar sind! [3 Punkte]*

b *) Gegeben sei folgendes Prolog-Programm. Schreiben Sie das Programm als Hornausdruck gemäß der Darstellung in der Vorlesung. Führen Sie die SLD-Resolution mit der Anfrage „*`beute(X).`*“ durch, wobei Regeln mittels Tiefensuche gefunden werden. Gehen Sie davon aus, dass bei mehreren Lösungen die erste gültig ist.*

Hinweis: *Innerhalb der Hornklauseln stellt ',' (das Komma) das logische Oder dar, während in Prolog das ',' das logische Und darstellt. Wenn verwendet, kann der* **not***-Befehl direkt logisch interpretiert werden. [8 Punkte]*

Magdeburg, den 12.09.2024

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%          %%%%% HIER DIE KLAUSELMENGE %%%%%
% tier    - Tier                               %%%%%       ANGEBEN !!!     %%%%%
% jagt    - jagt
% bUJ     - BeutetierUndJaeger
% beute   - Beutetier
% jaeger -Jaeger
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

tier(drache).
tier(maus).
tier(katze).
tier(hund).

jagt(katze, maus).
jagt(adler, maus).
jagt(drache, hund).
jagt(hund, katze).

bUJ(X):-jagt(X, _), jagt(_, X).
beute(X):-tier(X),jagt(_, X).
jaeger(X):-jagt(X, _), not(jagt(_, X)).
```

/ 11

Magdeburg, den 12.09.2024

Aufgabe 5 *: Paralleles Paradigma*

a *) Gegeben seien folgende zwei Programme, die eine große Menge Daten mittels Bubble-Sort sortieren.*

```
1  public class BubbleParThread extends Thread {
2      int[] inputList;
3      int index, step;
4
5      public BubbleParThread(int[] inputList, int index, int step){
6          this.inputList = inputList;
7          this.index = index;
8          this.step = step;
9      }
10
11     public void run(){
12         for (int n = inputList.length; n > 1; --n) {
13             for (int i=index; i < n-step; i+=step) {
14                 if (inputList[i] > inputList[i+step]){
15                     swap(inputList, i, i + step);
16                 }
17             }
18         }
19     }
20 }
21
22 public class Bubble {
23     public static void main(String[] args){
24         int[] inputList = gatherData();
25         BubbleParThread bubbleTh = new BubbleParThread(inputList, 0, 1);
26
27         bubbleTh.start();
28
29         doStuffWith(inputList);
30     }
31 }
```

```
1  public class BubbleSeq {
2      public static void main(String[] args){
3          int[] inputList = gatherData();
4          int index = 0;
5          int step = 1;
6
7          for (int n = inputList.length; n > 1; --n) {
8              for (int i=index; i < n-step; i+=step) {
9                  if (inputList[i] > inputList[i+step]){
10                     swap(inputList, i, i + step);
11                 }
12             }
13         }
14
15         doStuffWith(inputList);
16     }
17 }
```

Magdeburg, den 12.09.2024

a) *Welches der beiden Programme läuft schneller und warum?*

b) *Erklären Sie die Auswirkungen der Nebenläufigkeit im ersten Programm bezüglich der Methode* `doStuffWith(inputList)`*!*

Gehen Sie dabei ggf. auch auf Lösungen von Problemen oder Vorteile ein.

[5 Punkte]

Magdeburg, den 12.09.2024

b *) Die CPU nutzt das Pipelining, um verschiedene Instruktionen zu verschiedenen Phasen parallel auszuführen. Bei einigen Instruktionen ist dies nicht ohne weiteres möglich. Daher nutzt sie Parallelisierungsstrategien. Wann ist das Pipelining zwischen Instruktionen allgemein nicht möglich? Welches Ziel verfolgen diese Parallelisierungsstrategien? Nennen und erklären Sie kurz zwei mögliche Techniken! Geben Sie je ein kleines Beispiel an, bei dem diese Parallelsierungstechniken angewendet werden! [4 Punkte]*

Magdeburg, den 12.09.2024

c) *Ein Programmierer möchte in seinem Quellcode die Berechnung der Methode* `doCalc` *parallelisiert ausführen. Was hindert ihn daran? Wie könnte eine Lösung dafür aussehen, sodass der Quellcode nach der Initialisierung des Arrays doch parallelisiert werden könnte? Begründen Sie in welchen Fällen Ihre Lösung korrekt funktioniert!* ***Hinweis:*** *Schauen Sie sich die Daten aus der* `getData`*-Funktion an! [4 Punkte]*

```
1  public static int[] getData(){
2      int[] data = new int[100];
3      for(int i=0;i<100;i++){
4          data[i] = i+1;
5      }
6      return data;
7  }
8
9  public static int calculate(int a, int b){
10     return a+b;
11 }
12
13 public static int[] doCalc(){
14     int[] data = getData();
15     for(int i=1;i<data.length;i++){
16         data[i]=calculate(data[i-1],data[i]);
17     }
18     return data;
19 }
```

/ 13

Magdeburg, den 12.09.2024

Aufgabe 6 *: Sonstiges*

a *) Zeichnen Sie irgendetwas (anständig bleiben!) [3 Punkte]*

/ 3