JAVA 04: Funktionen, Parameter und globale und lokale Variablen.

panitz

Zusammenfassung

In diesem Kurs finden sich meist recht einfache Fragen, die sich mit Funktionen und Variablen beschäftigen. Mehrere Fragen beschäftigen sich insbesondere mit rekursiven Funktionen.

Frage: Funktionen (1)

Wenn eine Funktion keinen Wert zurückliefert (also streng genommen eine Prozedur ist), dann muss was als Rückgabetyp deklariert werden?

Korrekte Antworten

- `void`

Erläuterung

Das entsprechende Schlüsselwort ist `void`. `void`-Methoden brauchen keine `return`-Anweisung. Sie können aber trotzdem auch durch eine `return`-Anweisung verlassen werden.

Frage: Funktionen (2)

Der Kopf (header) einer Funktion beinhaltet:

(Mehrere Antwortmöglichkeiten).

- Funktionsnamen
- Parameterliste oder Argumentlist
- Operandenliste
- Optionenliste
- Rückgabetyp

Korrekte Antworten

- Funktionsnamen
- Parameterliste oder Argumentlist
- Rückgabetyp

Erläuterung

Zusätzlich können Methoden noch Sichtbarkeitsattribute (`protected`, `public`, `private` im Kopf haben. Auch weitere Attribut wie `static`, `final` oder `native`.

Frage: Funktionen (3)

Die Umsetzung einer mathematischen Funktion in Java beinhaltet immer:

(Mehrere Antwortmöglichkeiten).

- Mindestens einen Parameter.
- Einen Funktionsnamen.
- Mindestens ein `return`-Statement.
- Ein break-Statement.

Korrekte Antworten

- Einen Funktionsnamen.
- Mindestens ein `return`-Statement.

Erläuterung

Funktionen müssen bei Aufruf immer einen Wert liefern und brauchen daher ein `return`-Statement. Eine Funktion ohne Parameter wird auch Konstante oder 0-stellige Funktion genannt.

Frage: Funktionen (4)

In welchen Symbolen sind die Argumente von Funktionen eingeschlossen?

(Eine Antwortmöglichkeit.)

- In (runden) Klammern.
- In [eckigen] Klammern.
- In Anführungszeichen "...".
- In geschweiften Klammern.
- In <spitzen> Klammern.

Korrekte Antworten

- In (runden) Klammern.

Erläuterung

Eckige Klammern gehören zur Array-Syntax, spitze Klammern zu generischen Typen und geschweifte Klammern zur Blockbildung.

Frage: Import (1)

Erzeugt die folgende Funktion einen Syntax-Error?

```
class Main{
  public static void main(String[] args){
    System.out.println(Math.sin(Math.PI));
  }
}
```

(Eine Antwortmöglichkeit.)

- Ja, weil `main` ein reserviertes Keyword ist.
- Ja, weil `Math` nicht richtig importiert wurde.
- Nein, alles o.k.

Korrekte Antworten

- Nein, alles o.k.

Erläuterung

Die Klasse `Math` ist im Standardpaket `java.lang` und muss daher nicht importiert werden. `main` ist kein reserviertes Schlüsselwort sondern ein normaler Bezeichner.

Frage: Funktionen (6)

Gegeben sei der folgende Funktionskopf:

```
int f(int p1,int p2,int p3,int p4);
```

Welche(r) der folgenden Aufrufe sind/(ist) korrekt?

(Mehrere Antwortmöglichkeiten).

- f(1,2,3,4)
- f(1,2,3)
- f(0b1,2,0xff,4)
- f("1,2,3,4")
- f(1,2,3,4.0)

Korrekte Antworten

- f(1,2,3,4)
- f(0b1,2,0xff,4)

Erläuterung

Es müssen vier Ausdrücke für ganzzahlige Werte durch Komma separiert übergeben werden. Es gibt unterschiedliche Typen von Literalen für ganze Zahlen. `0b1` ist ein Literal im Dualsystem. `0xff` ein Literal im Hexadezimalsystem.

Frage: Funktionen (7)

Gegeben sei folgende Funktionsdefinition:

```
static void nprint(String message, int n){
  while (n > 0){
    System.out.print(message);
    n -= 1;
  }
}
```

Was wird angezeigt, wenn man `nprint("a", 4)` aufruft?

(Eine Antwortmöglichkeit.)

- `aaaaa`
- `aaaa`
- `aaa`
- Der Aufruf kompiliert nicht wegen eines Typfehlers.
- Endlosschleife. Es wird `aaaaa....` gedruckt.

Korrekte Antworten

- `aaaa`

Erläuterung

Die Schleife im Methodenrumpf wird genau viermal durchlaufen. Die Schleifenbedingung wird dann falsch.

Frage: Parameterübergabe (1)

Gegeben sei die folgendes Programm und der nachfolgende Aufruf:

```
class E{
  static int k = 3;
  static void nprint(String message, int n){
    while (n > 0){
      System.out.print(message);
      n -= 1;
    }
    k = 0;
  }
  public static void main(String[] args){
    int k = 2;
    nprint("A message", k)
    System.out.println(k);
  }
}
```

Was druckt das Programm als letztes aus?

(Eine Antwortmöglichkeit.)

- 0
- 1
- 2
- 3

Korrekte Antworten

- 2

Erläuterung

Es gibt zwei Variablen `k`. Eine globale der Klasse und eine lokale in dr Methode `main`. Die lokale Variable `k` überdeckt die Klassenvariable `k`.

Frage: Parameterübergabe (3)

Was realisiert Java die der Übergabe einer Variable eines primitiven Typs als Funktionsargument?

(Eine Antwortmöglichkeit.)

- Einen undefinierten Funktionsaufruf.
- Call by Value.
- Call by Reference.
- Call by Name.

Korrekte Antworten

- Call by Value.

Erläuterung

Das bedeutet, dass man nicht eine Variable übergeben kann, und die Methode die Variable dann verändert. In folgendem Programm behält Variable x den Wert 41 und wird nicht erhöht.

```
class Value{
  static void inc(int x){ x = x +1;}
  public static void main(String[] _){
    var x = 41;
    inc(x);
    System.out.println(x);
  }
}
```

Frage: Namensräume (1)

Was wird bei der Ausführung des folgenden Programms ausgedruckt?

```
class T{
  static int x = 1;
  static void f1(){
    int y = x + 4;
    System.out.println(y);
  }
  public static void main(String[] args){
    f1();
    System.out.println(x);
  } }
```

(Eine Antwortmöglichkeit.)

- 1
- [0.01em]verb[1]

- Das Programm kompiliert nicht, weil `f1` nicht unqualifiziert in `main` aufgerufen werden kann.
- 1
  1
- [0.01em]verbM

Korrekte Antworten

- `[0.01em]verb`[1]

Erläuterung

Die beiden Variablen `x` und `y` sind komplett unabhängig. Eine Änderung der Variable `y` hat für die Variable `x` keinen Effekt, obwohl `y` ursprünglich mit dem Wert von `x` initialisiert wurde.

Frage: Namensräume (2)

Was wird bei der Ausführung des folgenden Programms ausgedruckt?

```
class T{
  static int x = 1;
  static void f1(){
    int x =  ;
    System.out.println(x);
  }
  public static void main(String[] args){
    f1();
    System.out.println(x);
  }
}
```

(Eine Antwortmöglichkeit.)

- `1`
- `[0.01em]verb`[1]
- Das Programm kompiliert nicht, weil `x` in `f1` nicht deklariert werden kann.
- `1`
  `1`
- `[0.01em]verbM`

Korrekte Antworten

- `[0.01em]verb`[1]

Erläuterung

Es gibt zwei Variablen `x`, die voneinander unabhängig sind. Die lokale Variable der Methode `f1` überdeckt die globale Variable `x` der Klasse `T`.

Frage: Namensräume (3)

Was wird bei der Ausführung des folgenden Programms ausgedruckt?

```
class T{
  static int x = 1;
  static void f1(){
    int x =  x + 3;
    System.out.println(x);
  }
  public static void main(String[] args){
    f1();
    System.out.println(x);
  }
}
```

(Eine Antwortmöglichkeit.)

- 1
  3
- 3
  1
- Das Programm kompiliert nicht, weil `x` in `f1` nicht initialisiert ist.
- 1
  1
- 3
  3

Korrekte Antworten

- Das Programm kompiliert nicht, weil `x` in `f1` nicht initialisiert ist.

Erläuterung

Mit der Deklaration der lokalen Variablen `x` wird die globale Variable `x` überdeckt. Bei der Initialisierung `x + 2` wird schon auf die lokale noch nicht initialisierte Variable bezug genommen.

Frage: Betragsfunktion

Die Funktion `Math.abs()` ist für Argumente welchen Typs anwendbar?

(Mehrere Antwortmöglichkeiten).

- Integer
- String
- float
- char
- List

Korrekte Antworten

- Integer
- float
- char

Erläuterung

Die Betragsfunktion ist für die Typen: `double`, `float`, `int`, `long` überladen. Andere Zahlentypen werden implizit auf einen dieser konvertiert. `Integer` durch Unboxing zu einen `int`. `char` wird zu einen `int` umgewandelt.

Frage: Rundung 1

Was liefert `Math.round()`?

Korrekte Antworten

-

Erläuterung

In diesem Fall wird aufgerundet. Das Ergebnis ist vom Typ `long>`.

Frage: Rundung 2

Was liefert `Math.round(6.5)`?

(Eine Antwortmöglichkeit.)

- 5
- 6
- 6.0
- 7
- 7.0

Korrekte Antworten

- 7

Erläuterung

Es wird bei `.5` noch aufgerundet.

Frage: Bogenmaß und Grad

Was liefert `Math.toDegrees(Math.PI / 2)`?

(Eine Antwortmöglichkeit.)

- 0.0
- 30.0
- 45.0
- 90.0

Korrekte Antworten

- 90.0

Erläuterung

$2 * \pi$ im Bogenmaß entsprechen einen Winkel von 360°.

Frage: Grad und Bogenmaß

Was liefert `Math.toRadians(30) * 6`?

(Eine Antwortmöglichkeit.)

- 0.0
- 1.0471975511965976
- 3.141592653589793
- 5.565656

Korrekte Antworten

- 3.141592653589793

Erläuterung

Ein Winkel von 30° entspricht im Bogenmaß $\frac{\pi}{6}$.

Frage: Trigonometrische Funktion

Was liefert `Math.sin(Math.PI / 6)`?

(Eine Antwortmöglichkeit.)

- 1.0
- 1.3434343
- 3.141592653589793
- 0.49999999999999994

Korrekte Antworten

- 0.49999999999999994

Erläuterung

Trigonometrische Funktionen rechnen im Bogenmaß und nicht in Grad.

Frage: Fließkommaliteral

Welchen Typ hat das Literal `5.6`?

(Eine Antwortmöglichkeit.)

- int
- double
- boolean
- float

Korrekte Antworten

- double

Erläuterung

Zahlenliterale mit einem Punkt als Fließkommatrennzeichen sind vom Typ `double`.

Frage: Rekursion (1)

Welche der folgenden Aussagen ist wahr?

(Mehrere Antwortmöglichkeiten).

- Jede rekursive Funktion benötigt einen Rekursionsanfang und eine Abbruchbedingung.
- Jeder rekursive Aufruf einer Funktion muss das Originalproblem vereinfachen oder reduzieren, in einer Form, die zum Rekursionsende konvergieren kann.
- Ein infiniter Regress kann auftreten, wenn die Rekursion das Problem nicht in der Art reduziert, dass es zur Erfüllung der Abbruchbedingung hin konvergiert.
- Jede rekursive Funktion muss einen Rückgabewert haben.
- Rekursive Funktionen werden, verglichen mit nicht-rekursiven Funktionen, speziell aufgerufen.

Korrekte Antworten

- Jede rekursive Funktion benötigt einen Rekursionsanfang und eine Abbruchbedingung.
- Jeder rekursive Aufruf einer Funktion muss das Originalproblem vereinfachen oder reduzieren, in einer Form, die zum Rekursionsende konvergieren kann.
- Ein infiniter Regress kann auftreten, wenn die Rekursion das Problem nicht in der Art reduziert, dass es zur Erfüllung der Abbruchbedingung hin konvergiert.

Erläuterung

Es gibt Programmiersprachen, in denen man auch rekursive Funktionen schreiben kann, die keine Abbruchbedingung haben. In Java sind solche Funktionen aber nutzlos, da sie immer zu einen Laufzeitfehler führen.

Frage: Rekursion (2)

Welche der folgenden Aussagen ist wahr?

(Eine Antwortmöglichkeit.)

- Die Fibonacci-Folge beginnt mit 0 und 1. Jedes folgende Element ist die Summe der zwei vorangehenden.
- Die Fibonacci-Folge beginnt mit 1 und 1 und jedes folgende Element ist die Summe der zwei vorangehenden.
- Die Fibonacci-Folge beginnt mit 1 und 2 und jedes folgende Element ist die Summe der zwei vorangehenden.
- Die Fibonacci-Folge beginnt mit 2 und 3 und jedes folgende Element ist die Summe der zwei vorangehenden.

Korrekte Antworten

- Die Fibonacci-Folge beginnt mit 0 und 1. Jedes folgende Element ist die Summe der zwei vorangehenden.

Erläuterung

Per Definition beginnt die Fibonaccifolge mit den Zahlen 0 und 1.

Frage: Rekursion - Iteration (1)

Analysieren Sie folgende Java-Funktion und markieren Sie die richtigen Aussagen.

```
static int f(int n){
  if (n == 1) return 1;
  return n + f(n - 1);
}
```

(Mehrere Antwortmöglichkeiten).

- f ist eine Funktion, die sich rekursiv aufruft.
- f ist eine Funktion, die die Lösungsmethode der Iteration nutzt.
- f ist eine Funktion, die die Lösungsmethode der Rekursion nutzt.
- Iterationen brauchen immer eine Abbruchbedingung.

Korrekte Antworten

- `f` ist eine Funktion, die sich rekursiv aufruft.
- `f` ist eine Funktion, die die Lösungsmethode der Rekursion nutzt.

Erläuterung

Iterationen werden durch Schleifen realisiert.

Frage: Rekursion - Iteration (2)

Welche der folgenden Aussagen ist wahr?

(Mehrere Antwortmöglichkeiten).

- Rekursive Problemlösungen laufen im Mittel langsamer als entsprechende nicht-rekursive Lösungen.
- Rekursive Funktionen benötigen fast immer weniger Speicherplatz als nichtrekurisve Funktionen.
- Primitiv-rekursive Funktionen können immer durch iterative Lösungen ersetzt werden.
- In einigen Fällen erlaubt die Rekursion eine Straightforward-Lösung, die anders nur schwer zu formulieren ist.

Korrekte Antworten

- Rekursive Problemlösungen laufen im Mittel langsamer als entsprechende nicht-rekursive Lösungen.
- Primitiv-rekursive Funktionen können immer durch iterative Lösungen ersetzt werden.
- In einigen Fällen erlaubt die Rekursion eine Straightforward-Lösung, die anders nur schwer zu formulieren ist.

Erläuterung

Besonders Baumalgorithmen lassen sich einfacher rekursiv als iterativ realisieren.

Frage: Rekursion in Java (2)

Was ist Abbruchbedingung für folgende rekursive Funktionsdefinition?

```
static void f(int n){
  if (n > 0){
    System.out.println(n % 10);
    f(n / 10);
  }
}
```

(Eine Antwortmöglichkeit.)

- n>0
- n<=0
- Es gibt keine Abbruchbedingung.
- n<0

Korrekte Antworten

- `n<=0`

Erläuterung

Als Abbruchbedingung wird ein Ausdruck bezeichnet, für den die rekursive Funktion sich nicht ein weiteres Mal aufruft.

Frage: Rekursion in Java (3)

Analysieren Sie die folgende rekursive Funktion.

```
static int f(int n){
  return n * f(n - 1);
}
```

Welche der folgenden Aussagen ist wahr?

(Mehrere Antwortmöglichkeiten).

- Der Aufruf von `f(0)` lierfert 0 zurück.
- Der Aufruf von f(1) lierfert 1 zurück.
- Der Aufruf von f(2) lierfert 2 zurück.
- Der Aufruf von f(3) lierfert 6 zurück.
- Die Funktion läuft endlos und erzeugt schließlich eine StackOverflowException.

Korrekte Antworten

- Die Funktion läuft endlos und erzeugt schließlich eine StackOverflowException.

Erläuterung

Es gibt keine Abbruchbedigung. Für jeden Wert des Parameters `n` ruft die Funktion sich selbst wieder aus. Auf dem Stack muss Information der gestarteten Aufrufe gespeichert werden. Irgendwann reicht der Speicherplatz des Stack hierzu nicht mehr aus.

Frage: Rekursion in Java (4)

Wie häufig wird die Funktion `f` bei der Auswertung von `f()` im folgenden Programm aufgerufen?

```
static int f(int n){
  if (n == 0) return 1;
  return n * f(n-1);
}
```

Korrekte Antworten

-

Erläuterung

Der Parameter wird für jeden rekursiven Aufruf um 1 verringert. Die Rekursion endet, wenn der Parameter den Wert 0 hat.

Frage: Rekursion in Java (7)

Was ist der Rekursionsanfang in folgender Funktion?

```
static int  f(int n){
  if (n == 1) return 1;
  return n + f(n - 1);
}
```

(Eine Antwortmöglichkeit.)

- n ist gleich 1.
- n ist größer als 1.
- n ist kleiner als 1.
- Es gibt keinen Rekursionsanfang.

Korrekte Antworten

- n ist gleich 1.

Erläuterung

Wie bei einem Induktionsbeweis spricht man bei der Rekursion von einem Rekursionsanfang. Gemeint ist dabei der terminierende Fall, in dem keine weitere Rekursion aufgerufen wird.

Frage: Rekursion in Java (8)

Was ist der Rückgabewert für `f()`?

```
static int f(int n){
  if (n == 1) return 1;
  return n + f(n - 1);
}
```

Korrekte Antworten

- ▪

Erläuterung

Es wird die Summe von 1 bis `n` berechnet. Dieses ginge mit der Gauß-Formel natürlich einfacher.

Frage: Rekursion in Java (9)

Vervollständigen Sie den nachstehenden Code an der Stelle ??? so, dass er überprüft, ob s ein Palindrom ist.

```
static boolean isPalindrome(String s){
  if (s.length() <= 1) return true;
  if ( ??? ) return false
  return isPalindrome(s.substring(1,s.length()-1));
}
```

(Eine Antwortmöglichkeit.)

- `s.charAt(0) != s.charAt(-1)`
- `s.charAt(0) != s.charAt(s.length())`
- `s.charAt(0) != s.charAt(s.length()-1)`
- `s.charAt(1) != s.charAt(s.length())`

Korrekte Antworten

- `s.charAt(0) != s.charAt(s.length()-1)`

Erläuterung

Es wird jeweils das letzte mit dem ersten Zeichen verglichen. Diese befinden sich am Index 0 und `s.length()-1`.

Frage: Rekursion in Java (10)

Analysieren Sie den folgenden Code und markieren Sie die wahre Aussage.

```
class A {
  static void  f(String x, int length){
    System.out.print(x.charAt(length - 1) +  " ");
    f(x, length - 1);
  }
  public static void main(String[] args){
    String x = "12345";
    f(x, 5);
  }
}
```

(Eine Antwortmöglichkeit.)

- Das Programm gibt `1 2 3 4 5` aus und erzeugt dann eine StringIndexOutOfBoundsException.
- Das Programm gibt `5 4 3 2 1` aus.
- Das Programm gibt `5 4 3 2 1` aus und erzeugt dann eine StringIndexOutOfBoundsException.
- Das Programm gibt `4 3 2 1 0` aus.

Korrekte Antworten

- Das Programm gibt `5 4 3 2 1` aus und erzeugt dann eine StringIndexOutOfBoundsException.

Erläuterung

Sie Funktion hat keine Abbruchbedingung für die Rekursion. Dadurch wird irgendwann der Index `-1` erreicht. Dieser führt in der Methode `charAt` zur `StringIndexOutOfBoundsException`.

Frage: Rekursion in Java (11)

Damit der folgende Code überprüft, ob s ein Palindrom ist, muss anstelle von ??? welcher der folgenden Ausdrücke eingesetzt werden:

```
static boolean isPalindrome(String s){
  return isPalindromeHelper(s, 0, s.length() - 1);
}
static boolean isPalindromeHelper
        (String s, int low, int high){
  if (high <= low) return true;
  if s.charAt(low) != s.charAt(high) return false;
  return ???;
}
```

(Eine Antwortmöglichkeit.)

- `isPalindromeHelper(s)`
- `isPalindromeHelper(s, low, high)`
- `isPalindromeHelper(s, low + 1, high)`
- `isPalindromeHelper(s, low , high - 1)`
- `isPalindromeHelper(s, low + 1, high - 1)`

Korrekte Antworten

- `isPalindromeHelper(s, low + 1, high - 1)`

Erläuterung

Die Parameter `low` und `high` laufen von vorne bzw hinten durch den String bis sie sich gegenseitig in der Mitte treffen.

Frage: Rekursion in Java (13)

Analysieren Sie die folgenden zwei Klassen A und B und markieren Sie die wahre Aussage.

```
class A{
  static void f(int length){
    if (length > 1){
      System.out.println((length - 1)+ " ");
      f(length - 1);
    }  }
  public static void main(String[] args){f(5);} }
class B{
  static void f(int length){
    while (length > 1){
      System.out.println((length - 1)+ " ");
      f(length - 1);
    }  }
  public static void main(String[] args){f(5);}}
```

(Eine Antwortmöglichkeit.)

- Die beiden Programme erzeugen die gleiche Ausgabe: 5 4 3 2 1 .
- Die beiden Programme erzeugen die gleiche Ausgabe: 1 2 3 4 5 .
- Die beiden Programme erzeugen die gleiche Ausgabe: 4 3 2 1.
- Die beiden Programme erzeugen die gleiche Ausgabe: 1 2 3 4 .
- Programm A erzeugt die Ausgabe 4 3 2 1 und Programm B geht in eine Endlosschleife.

Korrekte Antworten

- Programm A erzeugt die Ausgabe `4 3 2 1` und Programm B geht in eine Endlosschleife.

Erläuterung

Variante B macht etwas sehr Unschönes: es wird Rekursion und Iteration gemischt. Für die Iteration bleibt die Schleifenbedingung stets wahr.

Frage: Rekursion in Java (14)

Analysieren Sie die folgenden zwei Funktionen und kreuzen Sie die wahre Aussage an.

```
class A{
  static int f1(int n){
    if (n == 0) return 0;
    return n + f1(n - 1);
  }
  static int f2(int n, int result){
    if (n == 0) return result;
    return f2(n - 1, n + result);
  }
  public static void main(String[] args){
    System.out.println(f1(3));
    System.out.println(f2(3, 0));
  }
}
```

(Eine Antwortmöglichkeit.)

- `f1` ist eine end-rekursive (tail recursive) Funktion.
- `f2` ist eine end-rekursive (tail recursive) Funktion.
- `f1` und `f2` sind beide end-rekursiv (tail recursive).
- Weder `f1` noch `f2` sind end-rekursiv (tail recursive).

Korrekte Antworten

- `f2` ist eine end-rekursive (tail recursive) Funktion.

Erläuterung

End-rekursive Funktion in Java haben den rekursiven Aufruf direkt nach dem Schlüsselwort `return`. Theoretisch kann ein Compiler diese Rekursionen automatisch in eine Iteration umwandeln. Viele Compiler machen dieses, z.B. der Compiler der Sprache Scala. Der javac nimmt diese Optimierung jedoch nicht vor.

Frage: Rekursion in Java (15)

Was ist die Ausgabe des folgenden Codes?

```
class A{
  static int f2(int n, int result){
    if (n == 0) return 0;
    return f2(n - 1, n + result);
  }
  public static void main(String[] args){
    System.out.println(f2(7, 0));
  }
}
```

Korrekte Antworten

- `0`

Erläuterung

Der terminierende Fall der rekursiven Methode gibt immer die Zahl `0` zurück. Damit kann die Auswertung eines Aufrufs von `f` nur mit `0` terminieren.