JAVA 08: Kurze Programmieraufgaben in Java

panitz

Zusammenfassung

Dieser Kurs besteht aus 25 kleine Programmieraufgaben in der Programmiersprache Java. Können Sie Iteration und Rekursion? Wissen Sie, wie man Konstruktoren schreibt? Wie arbeiten Methoden mit Objekten? Wie überschreibt man die equals-Methode? Arbeiten mit Arrays. Wie schreibt man Unterklassen? Arbeiten mit generischen Typen. Implementierung von Schnittstellen. Ereignisse in Swing. Nach erfolgreichem Durcharbeiten dieser Programmieraufgaben sollten Sie in diesen Themen sicher sein.

Frage: contains auf int-Arrays

Schreiben eine Funktion, die testet ob eine bestimmte Zahl in einem Array enthalten ist. Java

```
      public final class SimpleArrayFuns {
  static boolean contains(int[] xs,int y){
    return false;
  }
}
```

Java

Lösung

Java

```
        public final class SimpleArrayFuns{
    static boolean contains(int[] xs,int y){
      for (var x:xs)
        if (x==y)
          return true;
      return false;
    }
  }
```

Java

Erläuterung

Mit einer foreach-Schleife lässt sich am einfachsten durch einen Array iterieren. Man kann jedes Element mit dem gesuchten vergleichen und im Erfolgsfall dann mit true die Methode verlassen. Wurde die Schleife ohne return-Anweisung komplett durchlaufen, so war das gesuchte Element offensichtlich nicht im Array gespeichert.

Frage: Summe der Elemente eines int-Arrays

Schreiben eine Funktion, die alle Elemente eines Array aufsummiert. Java

```
        public final class SimpleArrayFuns {
    static long sum(int[] xs){
      return 0;
    }
  }
```

Java

Lösung

Java

```
        public final class SimpleArrayFuns{
    static long sum(int[] xs){
      long result = 0L;
      for (long x:xs){
        result += x;
      }
      return result;
    }
  }
```

Java

Erläuterung

Mit einer foreach-Schleife lässt sich am einfachsten durch einen Array iterieren. Bei jedem Durchlauf wird das aktuelle Element auf die Ergebnisvariable aufaddiert.

Frage: Test auf Sortierung eines Arrays von Zahlen.

Schreiben eine Funktion, die testet, ob die Elemente eines Arrays aufsteigend sortiert sind. Java

```
        public final class SimpleArrayFuns {
    static boolean isSorted(int[] xs){
      return false;
    }
  }
```

Java

Lösung

Java

```java
public final class SimpleArrayFuns{
  static boolean isSorted(int[] xs){
    var b = Integer.MIN_VALUE;
    for (var x:xs){
      if (x<b) return false;
      b = x;
    }
    return true;
  }
}
```

Java

Erläuterung

Man merke sich in einer Variablen den zuletzt im Array gesehenen Wert. Diese sei zunächst mit der kleinsten Zahl initialisiert. Man durchlaufe den Array mit der foreach-Schleife. Wenn man auf ein Element trifft, das kleiner als das zuletzt gesehene ist, so war der Array nicht sortiert. Man setzt bei jedem Schleifendurchlauf die Variable des zuletzt gesehenen Elements für den nächsten Durchlauf neu.

Frage: containsWithProperty auf generischen Arrays

Schreiben eine Funktion, die testet, ob ein Element, für das ein Prädikat true ist, in einem Array enthalten ist. Java

```
      import java.util.function.Predicate;
public final class GenericArrayFuns {
  static <E> boolean contains(E[] xs, Predicate<E> p){
    return false;
  }
}
```

Java

Lösung

Java

```
        import java.util.function.Predicate;
public final class GenericArrayFuns{
  static <E> boolean contains(E[] xs, Predicate<E> p){
    for (var x:xs)
      if (p.test(x))
        return true;
    return false;
  }
}
```

Java

Erläuterung

Mit einer foreach-Schleife lässt sich am einfachsten durch einen Array iterieren. Man kann jedes Element mit dem gesuchten Prädikat testen und im Erfolgsfall dann mit true die Methode verlassen.

Frage: Sind generischen Arrays sortiert.

Schreiben eine Funktion, die testet, ein Array von Elementen, die `Comparable` implementieren aufsteigend sortiert ist.. Java

```
        public final class GenericArrayFuns {
    static <E extends Comparable<E>> boolean isSorted(E[] xs){
      return false;
    }
  }
```

Java

Lösung

Java

```
        public final class GenericArrayFuns {
    static <E extends Comparable<E>> boolean isSorted(E[] xs){
      for (int i=1;i<xs.length;i++)
        if (xs[i].compareTo(xs[i-1])<0)
          return false;
      return true;
    }
  }
```

Java

Erläuterung

Man durchlaufe den Array mit einen Index und vergleiche mit der Methode `compareTo` jeweils zwei nebeneinander liegende Objekte. Sind diese in falscher Reihenfolge, war der Array nicht sortiert. Läuft die Schleife ohne return-Anweisung durch, so sind die Elemente aufsteigend im Array gespeichert und es kann true zurück gegeben werden.

Frage: Dualzahldarstellung

Schreiben Sie für die Klasse `N` eine Methode, die die gespeicherte positive Zahl als Dualzahl in einem String darstellt. Java

```
class N{
  final long n;
  N(long n){
    assert n>0;
    this.n = n;
  }

  String toBin(){
    return "";
  }
}
```

Java

Lösung

Java

```
        class N{
    final long n;
    N(long n){
      assert n>0;
      this.n = n;
    }

    String toBin(){
      return n==1?"1":(new N(n/2).toBin()+n%2);
    }
  }
```

Java

Erläuterung

Die Beispiellösung ist rekursiv. Der Basisfall ist $n = 1$. Ansonsten ist $n > 1$ und damit auch $\frac{n}{2} > 0$, so dass rekursiv `new N(n/2).toBin()` aufgerufen werden kann. Es muss dann noch die letzte Ziffer per Modulo 2 errechnet und an das rekursive Ergebnis angehängt werden.

Frage: Oktaldarstellung

Schreiben Sie für die Klasse `N` eine Methode, die die gespeicherte positive Zahl als Oktalzahl in einem String darstellt. Java

```
      class N{
  final long n;
  N(long n){
    assert n>0;
    this.n = n;
  }

  String toOct(){
    return "";
  }
}
```

Java

Lösung

Java

```
        class N{
    final long n;
    N(long n){
      assert n>0;
      this.n = n;
    }

    String toOct(){
      return n<8?(""+n):(new N(n/8).toOct()+n%8);
    }
  }
```

Java

Erläuterung

Die Beispiellösung ist rekursiv. Der Basisfall ist $n < 8$. Ansonsten ist $n > 8$ und damit auch $\frac{n}{8} > 0$, so dass rekursiv `new N(n/8).toOct()` aufgerufen werden kann. Es muss dann noch die letzte Ziffer per Modulo 8 errechnet und an das rekursive Ergebnis angehängt werden.

Frage: Stringumkehrung

Schreiben eine Methode, die einen String mit den Zeichen in umgekehrter Reihenfolge des Stringparameters zurück gibt. Java

```
      class Reverse{
  static String reverse(String s){
    return "";
  }
}
```

Java

Lösung

Java

```
        class Reverse{

  static String reverse(String s){
    StringBuilder sb = new StringBuilder();
    for (var c:s.toCharArray())
      sb.insert(0,c);
    return sb+"";
  }

}
```

Java

Erläuterung

Man Durchlaufe die Zeichen des Eingabestring und hänge diese jeweils vorne an den Ergebnisstring. Hierzu kann man einen `StringBuilder` als Hilfsobjekt verwenden.

Frage: Palindrom

Schreiben eine Methode, die testet, ob ein String ein Palindrom ist. Java

```
    class Palindrom{
  static boolean istPalindrom(String s){
    return false;
  }
}
```

Java

Lösung

Java

```
        class Palindrom{
    static boolean istPalindrom(String s){
      for (int i=0;i<s.length()/2;i++)
        if (s.charAt(i)!=s.charAt(s.length()-i-1))
          return false;
      return true;
    }
  }
```

Java

Erläuterung

Man Durchlaufe den String mit einem Index bis zur Mitte und vergleiche jeweils i-te Zeichen mit dem i-letzten Zeichen (erste mit letzten, zweiten mit vorletzten, dritte mit drittletzten, vierte mit viertletzten…. Sind diese jeweils ungleich, kann es kein Palindrom sein.)

Frage: Quersumme

Schreiben eine Methode, die die Quersumme einer positiven Zahl berechnet. Java

```
      class Quersumme{
  static long quersumme (long n){
    assert n>=0;
    return 0;
  }
}
```

Java

Lösung

Java

```java
class Quersumme{
  static long quersumme (long n){
    assert n>=0;
    return n==0?0:(n%10+quersumme(n/10));
  }
}
```

Java

Erläuterung

Die Beispiellösung ist rekursiv. Es wird jeweils rekursiv die Methode für den Parameter `n` ohne die letzte Ziffer aufgerufen und der Wert der letzten Ziffer auf das Ergebnis aufaddiert. Die letzte Ziffer lässt sich mit modulo 10 abspalten.

Frage: Datumsordnung

Schreiben für die Klasse `Datum` eine Methode, die testet ob das `this`-Objekt im Kalender vor dem Parameter `that` liegt. Java

```
    class Datum{
  int tag, monat, jahr;
  Datum(int t, int m, int j){
    tag = t;
    monat = m;
    jahr = j;
  }

  boolean istFrüherAls(Datum that){
    return false;
  }
}
```

Java

Lösung

Java

```
        class Datum{
    int tag, monat, jahr;
    Datum(int t, int m, int j){
      tag = t;
      monat = m;
      jahr = j;
    }

    boolean istFrüherAls(Datum that){
      return jahr<that.jahr
        || jahr==that.jahr && monat < that.monat
        || jahr==that.jahr && monat == that.monat && tag < that.tag;
    }
  }
```

Java

Erläuterung

Die Beispiellösung benutzt einen großen logischen Ausdruck. Erst wird geschaut, ob schon das Jahr früher liegt, dann bei gleichem Jahr, ob der Monat früher liegt und schließlich bei gleichem Jahr und gleichem Monat, ob der Tag früher liegt.

Frage: Gleichheit für Datum

Überschreiben Sie für die Klasse `Datum` equals-Methode auf adäquate Weise.

Java

```
    class Datum{
  int tag, monat, jahr;
  Datum(int t, int m, int j){
    tag = t;
    monat = m;
    jahr = j;
  }
}
```

Java

Lösung

Java

```
class Datum{
  int tag, monat, jahr;
  Datum(int t, int m, int j){
    tag = t;
    monat = m;
    jahr = j;
  }
  @Override public boolean equals(Object o){
    if (null==o) return false;
    if (!(o instanceof Datum)) return false;
    Datum that = (Datum)o;
    return jahr==that.jahr&&monat==that.monat&&tag==that.tag;
  }
}
```

Java

Erläuterung

Es ist zu prüfen, ob der Parameter nicht `null` ist und auch vom Typ `Datum`. Anschließend ist der Typ `Datum` zuzusichern, um Tag, Monat und Jahr der beiden Objekte this und that zu vergleichen.

Frage: Fibonacci-Generator

Schreiben Sie eine Klasse, die eine Methode `long nextFib()` hat. Ein Objekt dieser Klasse soll bei jedem Aufruf der Methode `nextFib` die nächste Fibonaccizahl haben. Nacheinander soll die Methode für das Objekt also die Folge: 0, 1, 1, 2, 3, 5, 8, 13, 21,... erzeugen. Geben Sie hierzu der Klasse zwei Felder vom Typ `long`, die die nächsten beiden Zahlen, die die Methode `nextFib` zurück gibt, enthalten.

Java

```
    class Fib{
  //die nächsten beiden Fibonaccizahlen
  private long n0 = 0;
  private long n1 = 1;

  long nextFib( ) {
  }
  public static void main(String[] args){
    Fib fib = new Fib();
    for (int i=0;i <= 10; i = i+1){
      System.out.println(fib.nextFib());
    }
  }
  }
^^I
```

Java

Lösung

Java

```
        class Fib{
    private long n0 = 0;
    private long n1 = 1;

    long nextFib(){
      long result = n0;
      n0 = n1;
      n1 = n0 + result;
      return result;
    }
    public static void main(String[] args){
      Fib fib = new Fib();
      for (int i=0;i <= 10; i = i+1){
        System.out.println(fib.nextFib());
      }
    }
}
^^I
```

Java

Erläuterung

Ein Objekt merkt sich in zwei Feldern, die nächsten beiden Fibonaccizahlen. Bei der Methode `nextFib` wird jeweils der erste der beiden Zahlen (`n0`) für das Ergebnis genommen. Die zweite (`n1`) wird die neue erste und für die neue zweite werden beide addiert (`n0+n1`).

Frage: Schaltjahr

Schreiben Sie eine Methode, die testet, ob der Parameter eine Jahreszahl ist, die ein Schaltjahr ist.

Java

```
    class Schaltjahr{
  static boolean schaltjahr(int jahr){
    return false;
  }
}
```

Java

Lösung

Java

```
        class Schaltjahr{
   static boolean schaltjahr(int jahr){
     return jahr%400==0 || jahr%4==0 && jahr%100!=0;
   }
}
```

Java

Erläuterung

Die Beispiellösung nutzt einen einzigen bool'schen Ausdruck. Jahre, die durch 400 teilbar sind, sind Schaltjahre. Ansonsten in allen durch vier teilbaren Jahren aber nicht in Jahren, die durch 100 teilbar sind. 1900 war kein Schaltjahr aber 2000 war ein Schaltjahr.

Frage: Kreuz

Schreiben Sie eine Methode, die einen String mit n-Zeilen und n-Spalten erzeugt, dessen diagonales Kreuz mit dem Zeichen 'X' besetzt ist. Zum Beispiel für $n = 5$:

```
X   X
 X X
  X
 X X
X   X
```

Java

```java
        class Kreuz{
    static String kreuz(int n){
      return "";
    }
  }
```

Java

Lösung

Java

```
        class Kreuz{
  static String kreuz(int n){
    var r = "";
    for (int x=1;x<=n;x++){
      for (int y=1;y<=n;y++)
         r += x==y||x+y==n+1?'X':' ';
      r += '\n';
    }
    return r;
  }
}
```

Java

Erläuterung

Es soll ein 2-dimensionales Bild erzeugt werden. Das lässt sich gut lösen, durch eine Schleife in einer weiteren Schleife. Die äußere Schleife iteriert nacheinander über die verschiedenen Zeilen, die innere setzt dann jeweils das Zeichen in der entsprechenden Spalte einer Zeile. Nach der inneren Schleife ist jeweils das Zeilenende einzufügen.

Frage: Test auf Element in Array-basierter Liste

Schreiben Sie die Methode, die testet, ob ein bestimmtes Element in einer Array-basierten Liste ist.
Java

```java
    public class AL<E>{
  private E[] store = (E[])new Object[5];
  int size=0;
  void add(E e) {
    if (store.length <= size){
      E[] newStore = (E[]) new Object[size + 5];
      for (int i = 0; i < store.length; i++) {
        newStore[i] = store[i];
      }
      store = newStore;
    }
    store[size++] = e;
  }

  boolean contains(E e){
    return false;
  }
}
```

Java

Lösung

Java

```
        public class AL<E>{
    private E[] store = (E[])new Object[5];
    int size=0;
    void add(E e) {
      if (store.length <= size){
        E[] newStore = (E[]) new Object[size + 5];
        for (int i = 0; i < store.length; i++) {
          newStore[i] = store[i];
        }
        store = newStore;
      }
      store[size++] = e;
    }

    boolean contains(E e){
      for (int i = 0; i<size; i++)
        if (store[i].equals(e))  return true;
      return false;
    }
  }
```

Java

Erläuterung

Man beachte, dass man nur im `store` über die bereits gesetzten Elemente iterieren darf, also bis zum Index `i<size`. Zum Vergleich nutze man die `equals`-Methode und nicht die Identität über den Operator `==`.

Frage: foreach für Array-basierter Liste

Schreiben Sie die Methode, die ein `Consumer`-Objekt auf jedes Element in einer Array-basierten Liste anwendet. Java

```
      import java.util.function.Consumer;
public class AL<E>{
  private E[] store = (E[])new Object[5];
  int size=0;
  void add(E e) {
    if (store.length <= size){
      E[] newStore = (E[]) new Object[size + 5];
      for (int i = 0; i < store.length; i++) {
        newStore[i] = store[i];
      }
      store = newStore;
    }
    store[size++] = e;
  }

  void forEach(Consumer<E> con){
  }
}
```

Java

Lösung

Java

```
        import java.util.function.Consumer;
public class AL<E>{
  private E[] store = (E[])new Object[5];
  int size=0;
  void add(E e) {
    if (store.length <= size){
      E[] newStore = (E[]) new Object[size + 5];
      for (int i = 0; i < store.length; i++) {
        newStore[i] = store[i];
      }
      store = newStore;
    }
    store[size++] = e;
  }

  void forEach(Consumer<E> con){
    for (int i=0;i<size;i++){
      con.accept(store[i]);
    }
  }

}
```

Java

Erläuterung

Das `Consumer`-Objekt hat eine Methode `accept`, die für jedes Element, das bereits in der Liste gespeichert ist, anzuwenden ist.

Frage: Zeitberechnungen

Schreiben für die Klasse `Zeit` eine Methode, die ein neues Uhrzeitobjekt erzeugt, das eine bestimme Minutenanzahl nach dem this-Objekt liegt. Java

```
      class Zeit{
  final int stunde;
  final int minute;
  Zeit(int s, int m){
    stunde = s;
    minute = m;
  }

  Zeit minutenSpäter(int min){
    return null;
  }
}
```

Java

Lösung

Java

```
        class Zeit{
  final int stunde;
  final int minute;
  Zeit(int s, int m){
    stunde = s;
    minute = m;
  }

  Zeit minutenSpäter(int min){
    return new Zeit((stunde+(minute+min)/60)%24,(minute+min)%60);
  }
}
```

Java

Erläuterung

Man beachte, dass es notwendig ist, alle 60 Minuten die Stunden zu erhöhen. Hierzu dient die Rechnung modulo 60 auf die Minuten. Ebenso ist mit den Stunden zu verfahren, die alle 24 Stunden wieder bei 0 Uhr beginnen. Hier ist also modulo 24 zu rechnen.

Frage: Letzteziffern

Schreiben Sie eine Methode, die für eine Zahl, einen String aus den letzten beiden Ziffern de Zahl erzeugt. Für Zahlen aus einer Ziffer, habe auch der Ergebnisstring nur eine Ziffer. Java

```
      class Letzteziffern{
  static String ziffern(int n){
    return "";
  }
}
```

Java

Lösung

Java

```
        class Letzteziffern{
   static String ziffern(int n){
     return ""+Math.abs(n%100);
   }
}
```

Java

Erläuterung

Mit modulo 100, lassen sich die letzten beiden Ziffern abspalten. Negative Zahlen können zuvor mit der Standardmethode `Math.abs` auf den Betragswert umgerechnet werden.

Frage: Konstruktoren

Versehen Sie die Klassen `Date` mit einem adäquaten Konstruktor, so dass Sie mit `new Date(30,11,2018)` ein Datumsobjekt erzeugen können.

Java

```
    class Date{
  int day;
  int month;
  int year;
}
```

Java

Lösung

Java

```java
class Date{
  int day;
  int month;
  int year;
  Date(int day, int month, int year){
    this.day = day;
    this.month = month;
    this.year = year;
  }
}
```

Java

Erläuterung

Der Konstruktor initialisiert alle Felder mit den als Parameter übergebenen Werten.

Frage: Comparable Schnittstelle

Implementieren Sie für die Klasse `Date` die Schnittstelle `Comparable`, so dass Datumsobjekte nach dem Kalender verglichen werden. Sie können die Methode `dateNumber` dabei nutzen.

Java

```
    class Date implements Comparable<Date>{
  int day;
  int month;
  int year;
  int dateNumber(){
    return year*10000+month*100+day;
  }
}
```

Java

Lösung

Java

```
class Date implements Comparable<Date>{
  int day;
  int month;
  int year;
  int dateNumber(){
    return year*10000+month*100+day;
  }
  @Override
  public int compareTo(Date that){
    return this.dateNumber()-that.dateNumber();
  }
}
```

Java

Erläuterung

Die Abbildung der Datumsobjeket auf `int`-Zahlen mit der Methode `dateNumber` erhält die Ordnung der Objekte. So kann die Ordnung auf `int` benutzt werden.

Frage: Swing Events

Ergänzen Sie die Klasse `Counter` so, dass beim Drücken des Knopfes die Variable `counter` um 1 erhöht und dann der Wert auf dem Label angezeigt wird.

Java

```
      import javax.swing.*;
class Counter extends JPanel{
  int counter = 0;
  JButton b = new JButton("drücken");
  JLabel l = new JLabel(""+counter);
  Counter(){
    add(b);
    add(l);
  }
}
```

Java

Lösung

Java

```
        import javax.swing.*;
class Counter extends JPanel{
  int counter = 0;
  JButton b = new JButton("drücken");
  JLabel l = new JLabel(""+counter);
  Counter(){
    add(b);
    add(l);
    b.addActionListener(e -> {counter++;l.setText(counter+"");});
  }
}
```

Java

Erläuterung

Da `ActionListener` eine funktionale Schnittstelle ist, kann sie einfach mit einem Lambda-Ausdruck initialisiert werden. Die Klasse `JButton` hat eine Methode `addActionListener`, um eine Methode, die bei Drücken des Knopfes ausgeführt wird, zu übergeben.

Frage: Unterklasse

Machen Sie die Klasse `Student` zu einer Unterklasse von `Person`. Sie soll ein Feld `matrNr` vom Typ `int` haben und einen Konstruktor, in dem in dieser Reihenfolge Name, Vorname und Matrikelnummer übergeben werden. Java

```
    interface PersonenArten{
  static class Person{
    String name, vorname;
    Person(String name, String vorname){
      this.name = name;
      this.vorname = vorname;
    }
  }

  static class Student{
  }
}
```

Java

Lösung

Java

```
        interface PersonenArten{
    static class Person{
      String name, vorname;
      Person(String name, String vorname){
        this.name = name;
        this.vorname = vorname;
      }
    }

    static class Student extends Person{
      int matrNr;
      Student(String name, String vorname,int matrNr){
        super(name,vorname);
        this.matrNr = matrNr;
      }
    }
  }
```

Java

Erläuterung

Die erste Anweisung eines Konstruktors muss immer der Aufruf eines Konstruktors der Oberklasse mit dem Schlüsselwort `super` sein.

Frage: Fibonacci Rekursiv

Implementieren Sie die Berechnung der n-ten Fibonaccizahl nach der rekursiven Definition. Es gelte:

- fib(0) = 0
- fib(1) = 1
- fib(2) = 1
- fib(3) = 2
- ...
- fib(n) = fib(n-2) + fib(n-1)

Java

```
      class Fib{
  static int fib(int n) {
    assert n >= 0;
    if (n <=1) return n;
  }
}
^^I
```

Java

Lösung

Java

```
        class Fib{
  static int fib(int n) {
    assert n >= 0;
    if (n <=1) return n;
    return fib(n-2) + fib(n-1);
  }
}
^^I
```

Java

Erläuterung

Die Definition lässt sich direkt abschreiben. Interessant ist bei dieser Funktion, dass die Rekursion doppelt ist. Das führt zu exponentiellen Wachstum.