Landeshauptstadt
München
**Referat für
Bildung und Sport**

Arbeitsheft

Bayerischer Landtag

mit OOP, XML, HTML & Swing
zu den Abgeordneten

Städtische Berufsschule für Informationstechnik

Herausgeber	Michael Niedermair Städtische Berufsschule für Informationstechnik https://www.bsinfo.eu/
Autor(en)	Michael Niedermair, Thomas Grosser
Quellen, Material, Unterstützung, ...	• Elke Niedermair
Satz	Michael Niedermair
Version	2016-01-28 (Revision: 1617) (ALPHA-Entwurf)
Lizenz	*closed version*

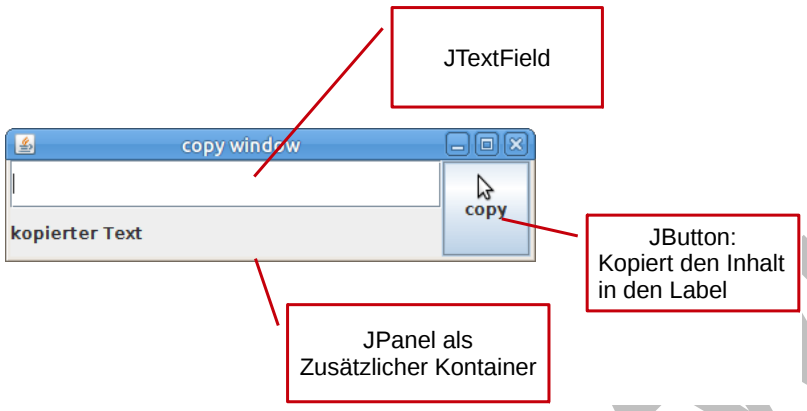
Inhaltsverzeichnis

Hinweise:
XXXX

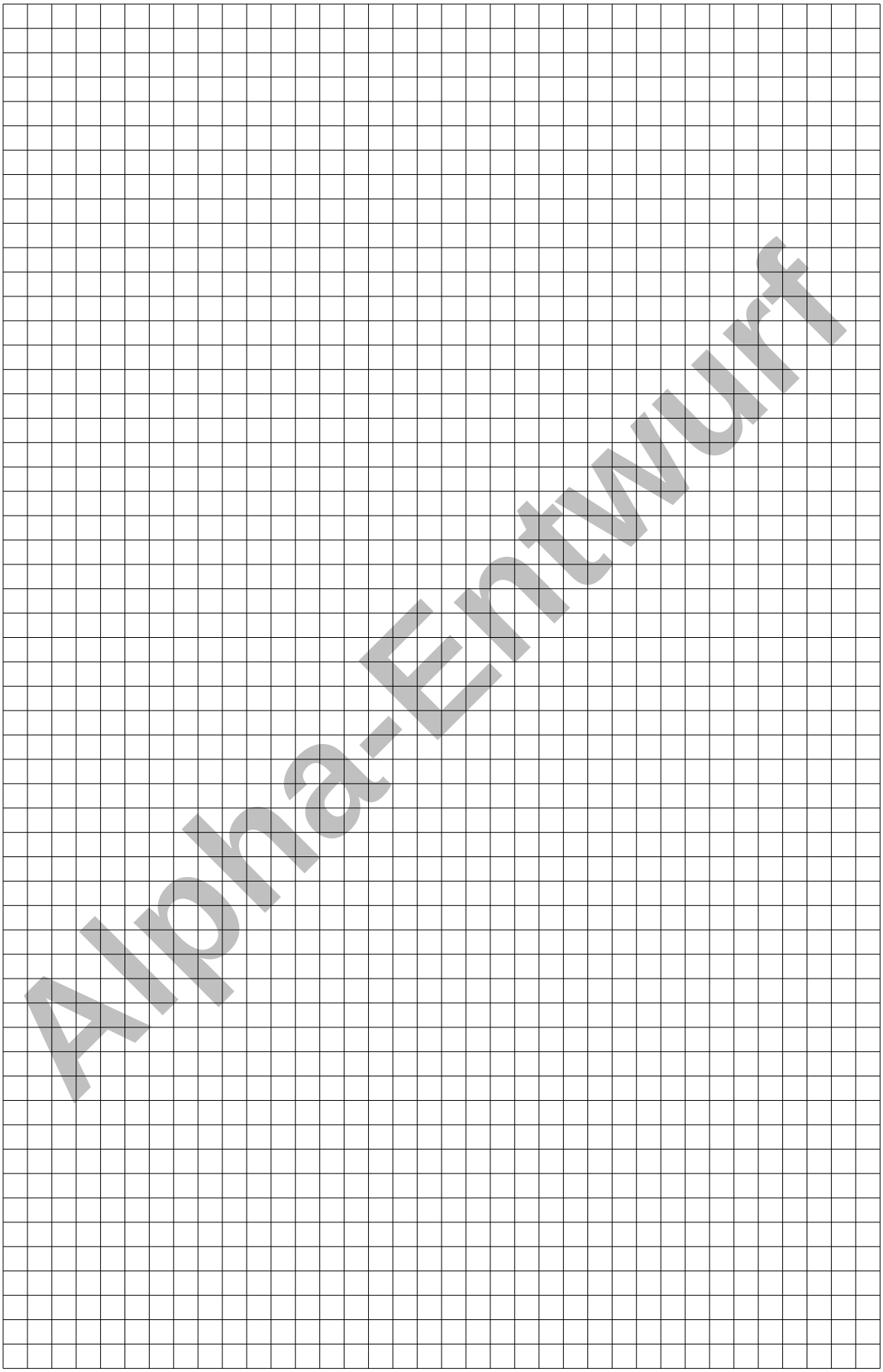
9

Aufgaben: **f_swing_gui_ew_01**

1. Probieren Sie den Code zu `EasyWindow` aus.
   Betrachten Sie den Code! Wenn Sie Fragen haben, so notieren Sie diese, fragen Ihren Nachbarn, um die Frage zu klären und erst danach den Lehrer.
2. Ermitteln Sie, welche Layoutmanager es noch gibt und wie diese die Komponenten anordnen.
3. Erstellen Sie ein Programm (Name `CopyWindow`), welches den Text aus einem Textfeld in den Label kopiert, sobald der Anwender auf den Button klickt. Danach soll der Inhalt des Textfeldes gelöscht werden und der Cursor sich wieder darin befinden.



4. Sorgen Sie dafür, dass wenn der Anwender die RETURN-Taste im Textfeld betätigt, die selbe Aktion ausgeführt wird, als wenn er auf den Button klicken würde.
   `addKeyListerner()`, ...
5. Erstellen Sie ein Programm (Name `BmiRechner`), welches den BMI berechnet und die entsprechende Kategorie anzeigt.
   siehe `https://de.wikipedia.org/wiki/Body-Mass-Index`



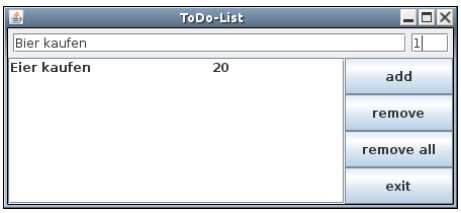
13

Aufgaben: f_swing_gui_tl_01

1. Kopieren Sie das Programm `EasyWindow` nach `ToDoList` und nehmen die entsprechenden Änderungen vor. Implementieren Sie die Funktionen: „add“, „remove“, „remove all“ und „exit“
2. Erweitern Sie die „ToDo-Liste“, so dass die Daten in einer Textdatei mit einem `Writer` geschrieben und mit einem `Reader` gelesen werden. Als Encoding verwenden Sie „UTF-8“.

   Verwenden Sie als Klassennamen „ToDoListIO“.
3. Erweitern Sie die „ToDo-Liste“, so dass die Beschreibung und zusätzlich die Menge eingegeben werden kann.

   Verwenden Sie als Klassennamen „ToDoList2IO“.



Verwenden Sie nun ein Datenhaltungsobjekt für den Eintrag:

```
   private class Entry {
      public String besch = "";
      public int anz = 0;

      @Override
      public String toString() {
         return (besch + "                                        ")
               .substring(0, 40) + " " + anz;
      }
   }
```

Ergänzen Sie alle Codestellen (auch `save()` und `load()`), so dass die Daten getrennt durch ein Tab-Zeichen „\ t“ gespeichert werden.

4. Über einen `XMLEncoder` kann ein Objekt in einer XML-Datei gespeichert und über `XMLDecoder` wieder geladen werden (mehr dazu auf Seite 94).

   Wandel Sie dazu die Klasse `Entry` in ein JavaBean um.

   Erstellen Sie dazu eine innere Klasse `MyListModel`

```
public static class MyListModel<T> extends AbstractListModel<T>
                                   implements ListModel<T> {
```

und implementieren die nachfolgende Methoden:

```
public boolean addElement(T val) ...
public T getElementAt(int idx) ...
public int getSize() ...
public boolean isEmpty() ...
private void load() ...
public void removeAllElements() ...
public T removeElementAt(int idx) ...
public void save() ...
```

Lesen Sie nach, was das `T` bedeutet (mehr dazu auf Seite 128).

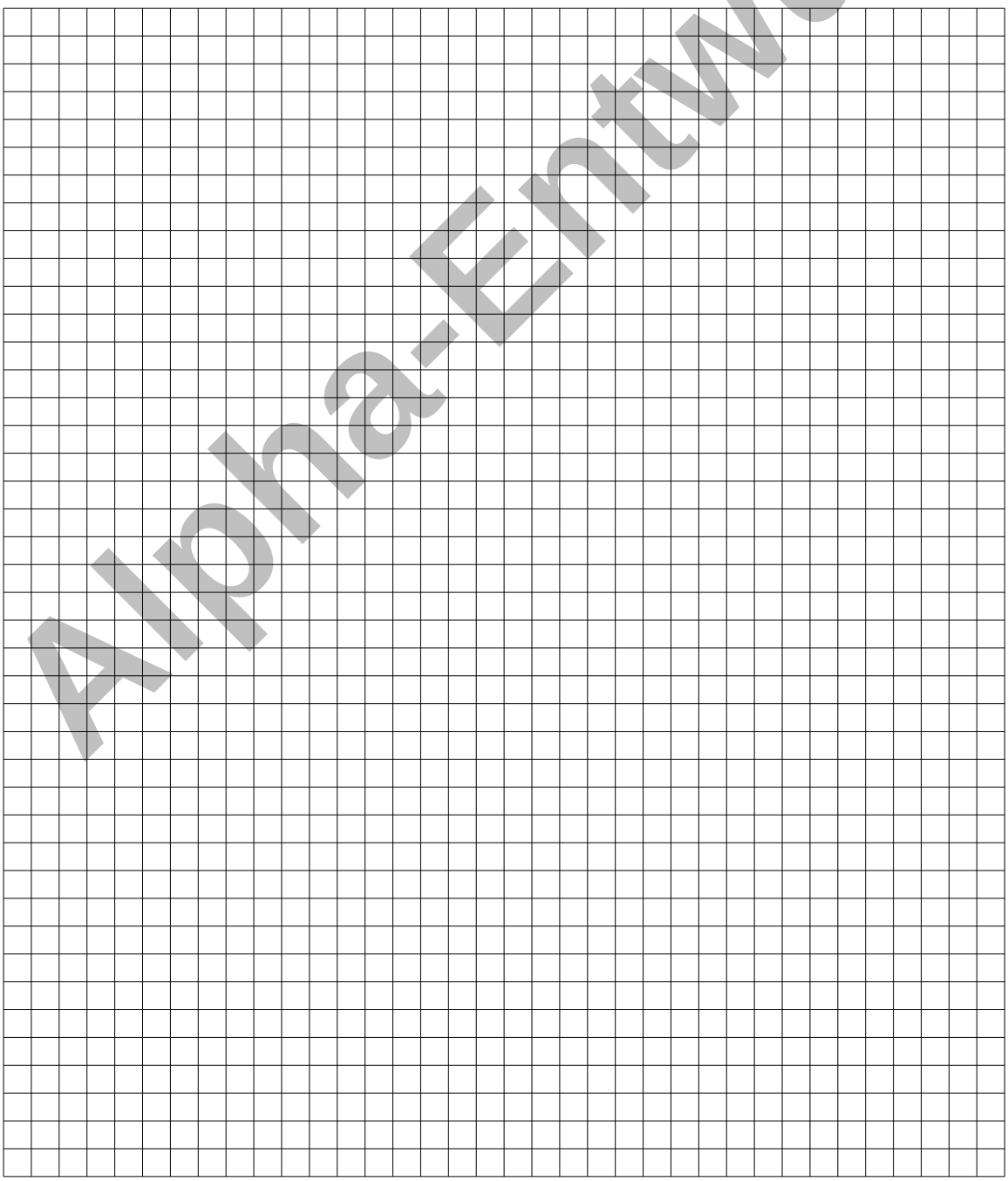
Als „Model“ verwenden Sie eine `ArrayList`.

Beispiel der so erzeugten XML-Datei:

```
<?xml version="1.0" encoding="UTF-8"?>
<java version="1.8.0_66" class="java.beans.XMLDecoder">
   <object class="java.util.ArrayList">
      <void method="add">
         <object class="de.nm.gui.swing.ToDoListXMLIO$Entry">
            <void property="anz">
               <int>10</int>
            </void>
            <void property="besch">
               <string>Bier kaufen</string>
            </void>
         </object>
```

```
      </void>
      <void method="add">
         <object class="de.nm.gui.swing.ToDoListXMLIO$Entry">
            <void property="anz">
               <int>20</int>
            </void>
            <void property="besch">
               <string>Eier besorgen</string>
            </void>
         </object>
      </void>
   </object>
</java>
```

5. Bauen Sie zwei `KeyListerner` ein:
   1. Für das Textfeld „Beschreibung“, welches dafür sorgt, dass wenn der Anwender die ENTER-Taste (oder RETURN-Taste) betätigt, automatisch ins Textfeld „Anzahl“ gesprungen wird.
   2. Für das Textfeld „Anzahl“, dass die selebe Aktion ausgeführt wird, als wenn der Anwender auf den Button „add“ klickt.



17 **Aufgaben:** **f_ltxml_oop_01**

1. Erstellen Sie aus der Datensammlung ein UML-Klassenmodell mit Attributen, Methoden und Beziehungen (siehe Abbildung 2.1).

   Jede Klasse hat dabei eine eindeutige `id` (Type `String`), die über alle Klassen eindeutig sein muss (später mehr dazu). Die Klasse abstrakte `Bezeichnung` ergänzt dies um eine Bezeichnung (`bezeichnung` Type `String`). Fassen Sie dies in abstrakten Klassen zusammen.

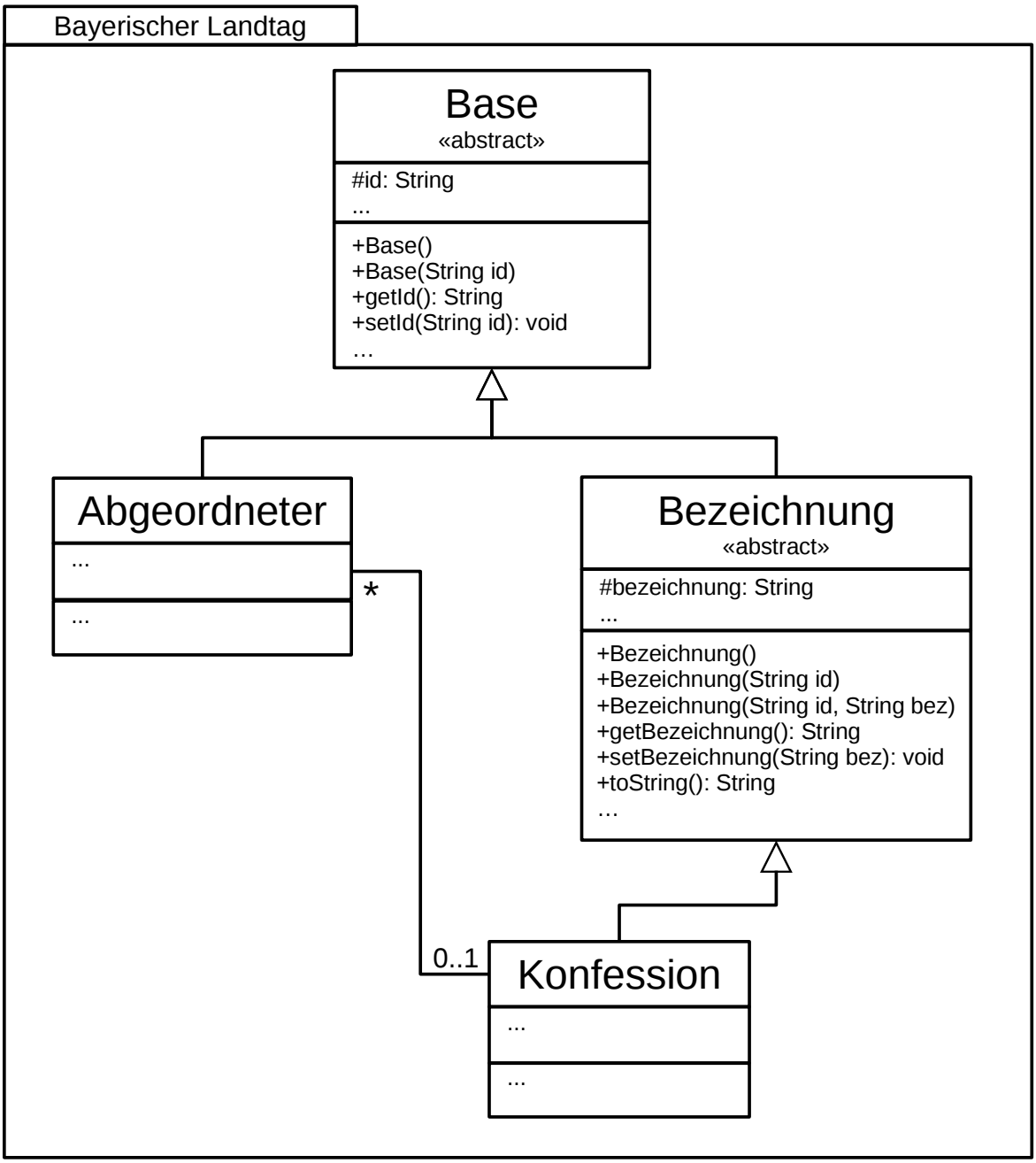
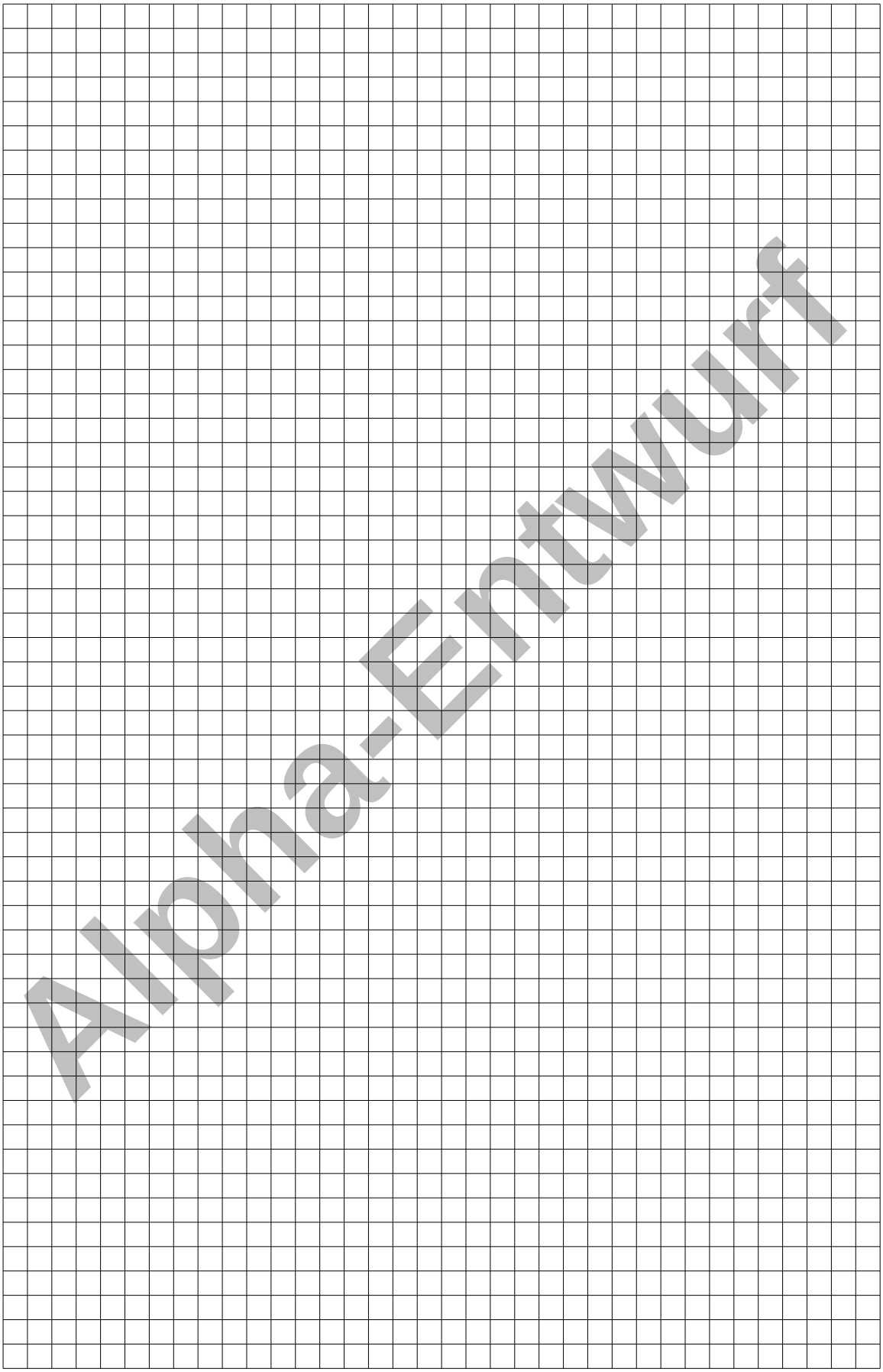


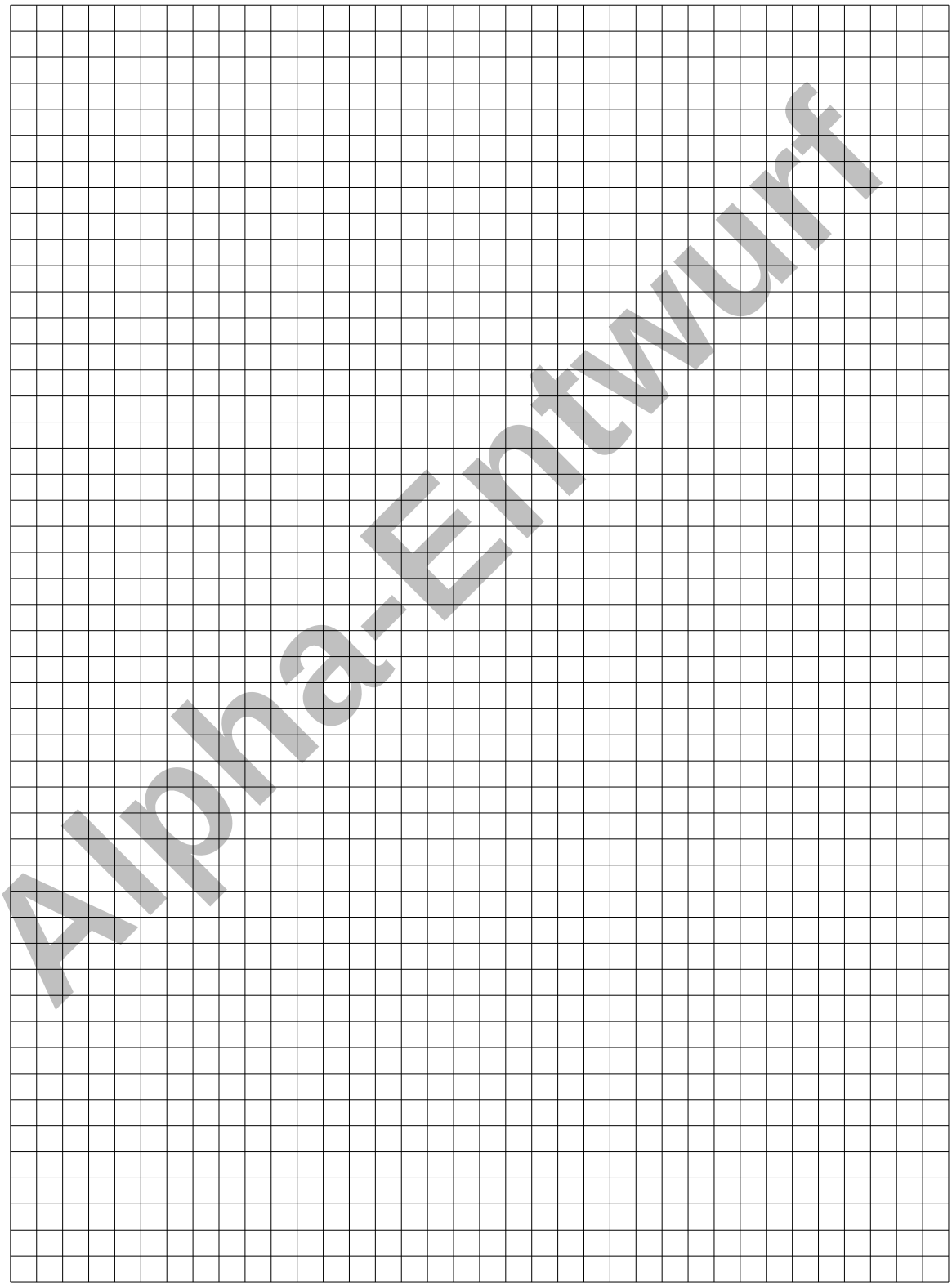
Abb. 2.1: Vorschlag Klassenmodell

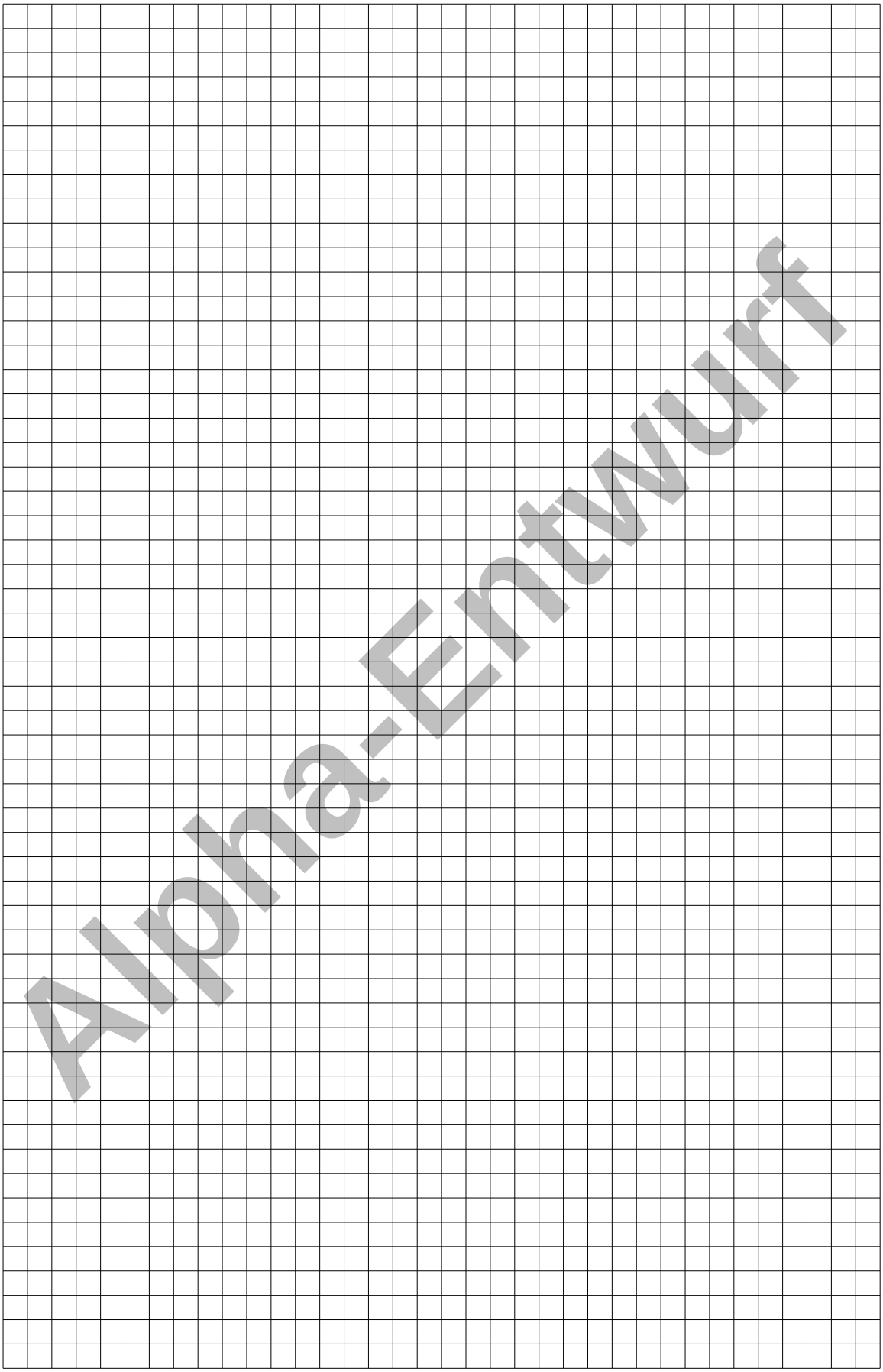


39

Aufgaben: **f_ltxml_junit_01**

1. Suchen Sie im Internet nach den Schlagworten „junit tutorial deutsch“ und erarbeiten Sie sich die Funktionalität von JUnit, beispielsweise mit Hilfe von
   `http://www.tutego.de/blog/javainsel/2010/04/junit-4-tutorial-java-tests-mit-junit/`.
   Notieren Sie sich alle `assert`-Methoden und deren Funktion.





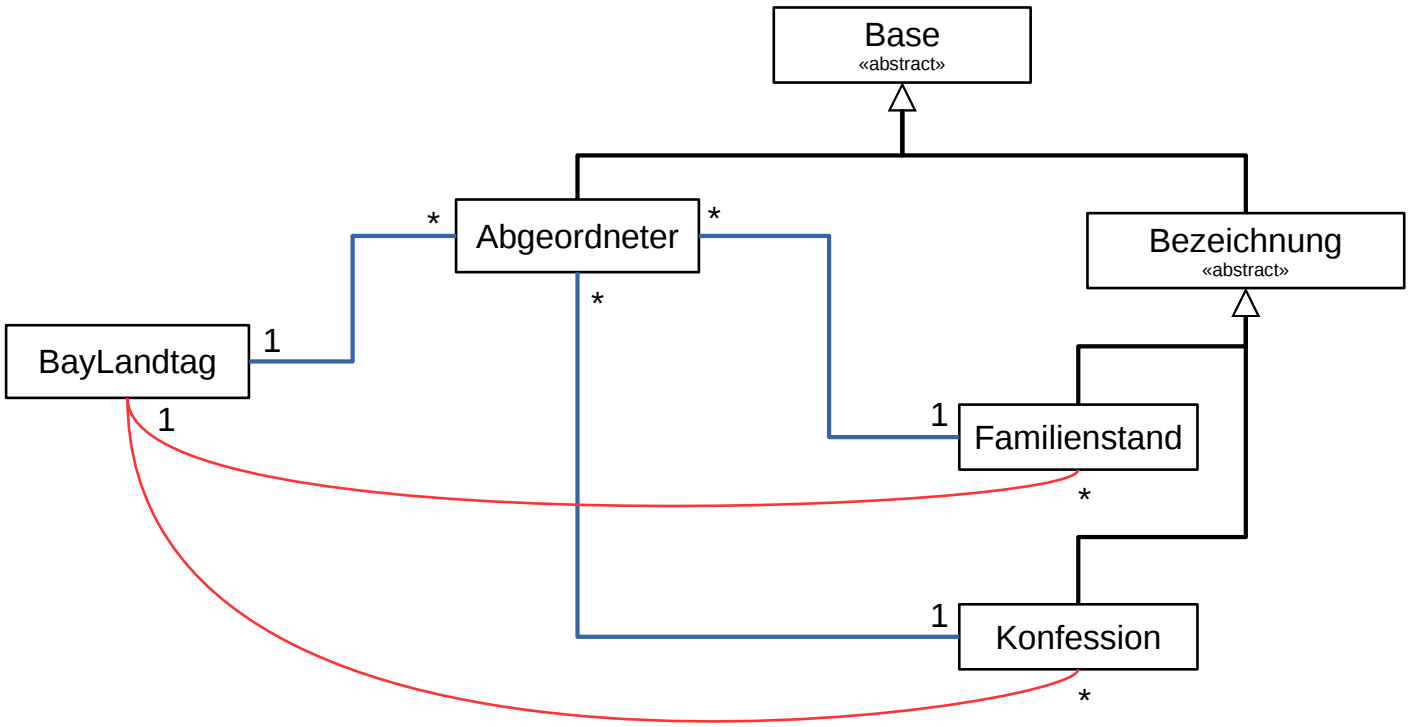
39

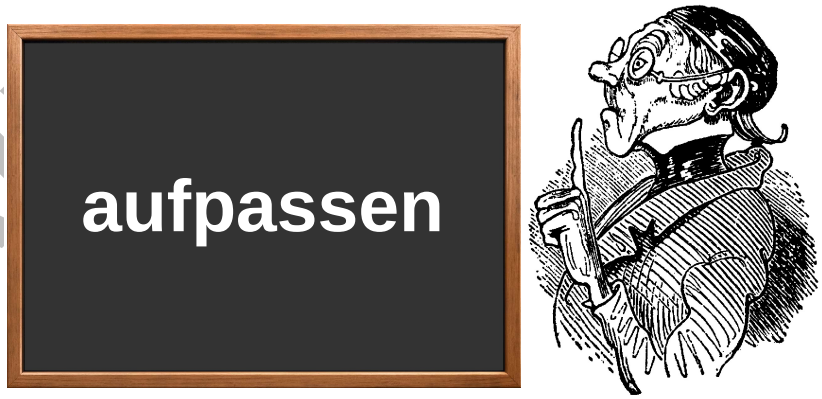
Aufgaben: **f_ltxml_test_01**

1. Sorgen Sie zuerst dafür, dass alle Compilerfehler (diese treten auf, da die Klassen und deren Methoden noch nicht existieren) beseitigt werden. Beachten Sie, dass alle erzeugten Java-Dateien im Verzeichnis `src/java/de/nm/ltxml/core` bzw. `src/java/de/nm/ltxml/core/bez` zu speichern sind. Nutzen Sie dazu die Möglichkeiten von Eclipse über Strg+1.
   Arbeiten Sie die Tests nach Ihrer Reihenfolge ab!
   `T_00_KonfessionTest`
   `T_01_FamilienstandTest`
   `T_02_AbgeordneterTest`
   `T_03_BayLandtagTest`
   `...`
   Betrachten Sie bei jedem Test den entsprechenden Javacode.
   Sollten hier Fragen auftauchen, so notieren Sie diese, Fragen zuerst Ihre Klassenmitglieder, dann Ihre Lehrer.
2. Sorgen Sie dafür, das Test `T_00_KonfessionTest` ohne Fehler läuft.
3. Sorgen Sie dafür, das Test `T_01_FamilienstandTest` ohne Fehler läuft.
4. Sorgen Sie dafür, das Test `T_02_AbgeordneterTest` ohne Fehler läuft.
5. Sorgen Sie dafür, das Test `T_03_BayLandtagTest` ohne Fehler läuft.
   Bei diesem Test wird die OOP-Struktur mit Hilfe von JAXB in eine Datei geschrieben. Um zu testen, ob die Datei „richtig“ erzeugt worden ist, wird diese wieder eingelesen. Dazu wird die Bibliothek JDOM in der Version 2 verwendet.

```
final SAXBuilder saxBuilder = new SAXBuilder();
final Document document = saxBuilder.build(output);
final Element root = document.getRootElement();

// check root
assertEquals("baylandtag", root.getName());

// check abg
final Element abge = root.getChild("abgeordnete").getChild("abg");
assertEquals("a1", abge.getAttributeValue("id"));
assertEquals("Kobold", abge.getAttributeValue("name"));
```

   JDOM verwendet für das Lesen der XML-Datei einen SAX-Parser und erzeugt daraus ein XML-Dokument. Die Klasse stellt diverse Methoden bereit, um das Dokument auszuwerten.
6. Rufen Sie die JDOM-API (Version 2) auf und ermitteln die wichtigsten Methoden der Klassen `Document`, `Element` und `Attribut`.


Base
«abstract»
Abgeordneter
Bezeichnung
«abstract»
BayLandtag
Familienstand
Konfession
*
*
*
1
1
1
*
1
*


aufpassen

39

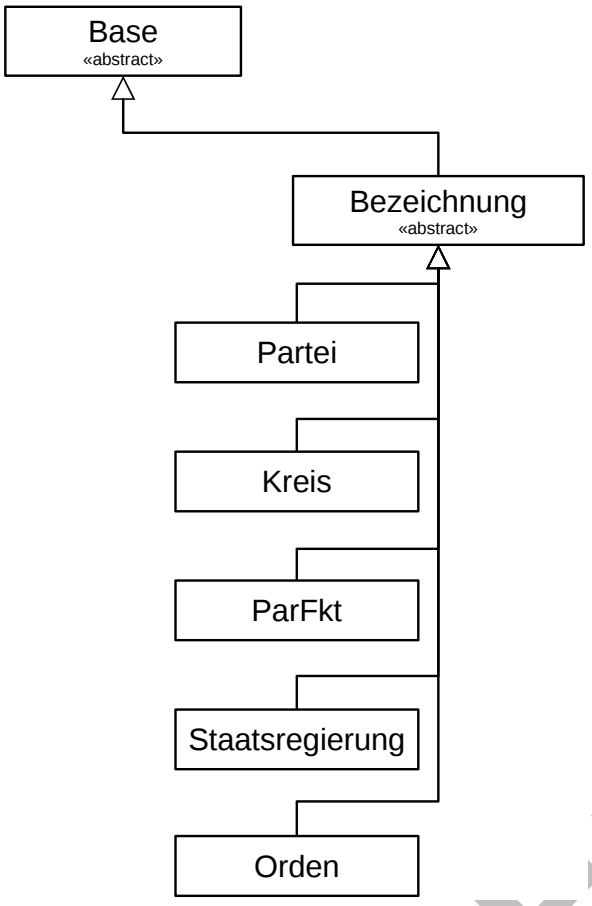
Aufgaben: **f_ltxml_test_02**

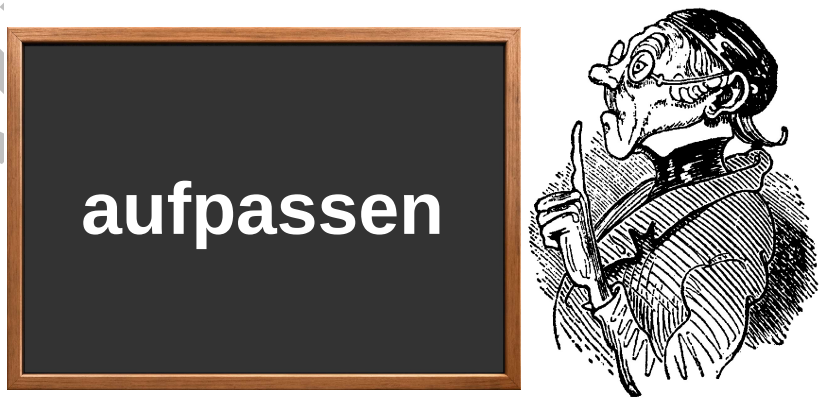
1. Sorgen Sie dafür, das Test `T_04_OrdenTest` ohne Fehler läuft.
2. Sorgen Sie dafür, das Test `T_05_StaatsregierungTest` ohne Fehler läuft.
3. Sorgen Sie dafür, das Test `T_06_ParFktTest` ohne Fehler läuft.
4. Sorgen Sie dafür, das Test `T_07_KreisTest` ohne Fehler läuft.
5. Sorgen Sie dafür, das Test `T_08_ParteiTest` ohne Fehler läuft.
6. Sorgen Sie dafür, das Test `T_09_BayLandtagTest` ohne Fehler läuft.

   Da die Abfragen der XML-Datei jetzt immer aufwendiger werden, wird für das Ermitteln der XML-Elemente XPATH verwendet.

```
final XPathFactory factory = XPathFactory.instance();
XPathExpression<Element> xp = factory.compile("/baylandtag/orden/ord",
                                  Filters.element());
final Element orde = xp.evaluateFirst(document);
assertEquals("o1", orde.getAttributeValue("id"));
assertEquals("Bayerischer Verdienstorden", orde.getAttributeValue("bezeichnung"));

final List<Element> ordlist = xp.evaluate(document);
assertEquals(2, ordlist.size());
final Element ord2 = ordlist.get(1);
assertEquals("o2", ord2.getAttributeValue("id"));
assertEquals("Ritterkreuz", ord2.getAttributeValue("bezeichnung"));
```

6. Rufen Sie die JDOM-API (Version 2) auf und suchen dort die Beispiele für XPATH. Notieren Sie sich die wichtigsten Methoden, um Elemente und Attribute zu finden und deren Inhalt zu erhalten, so dass diese verglichen werden können.
7. Testen Sie alle bisherigen Tests mit Hilfe von `T_10_00_09_Test`.



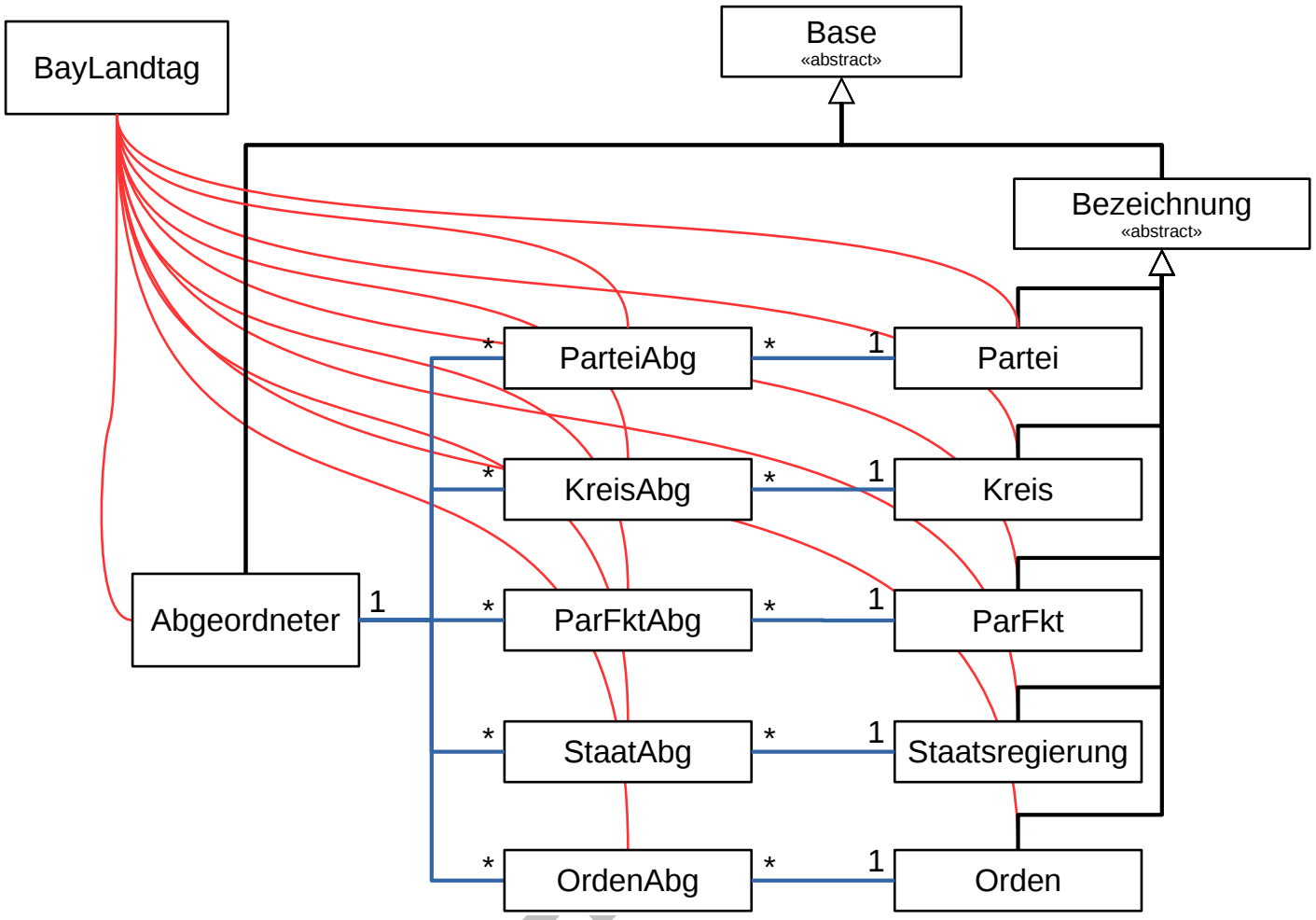


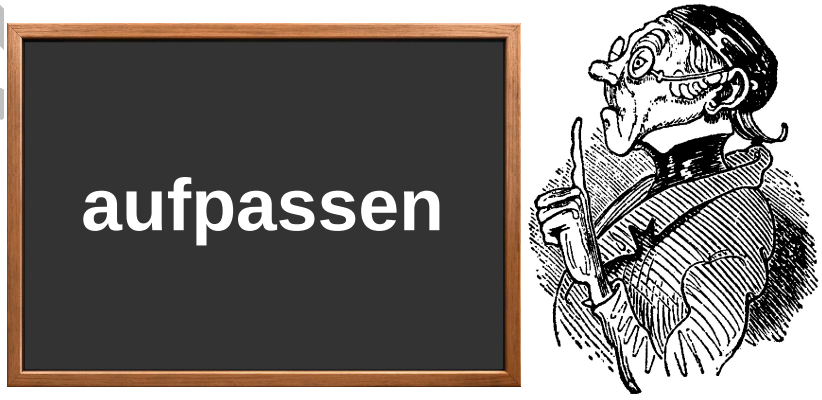
39

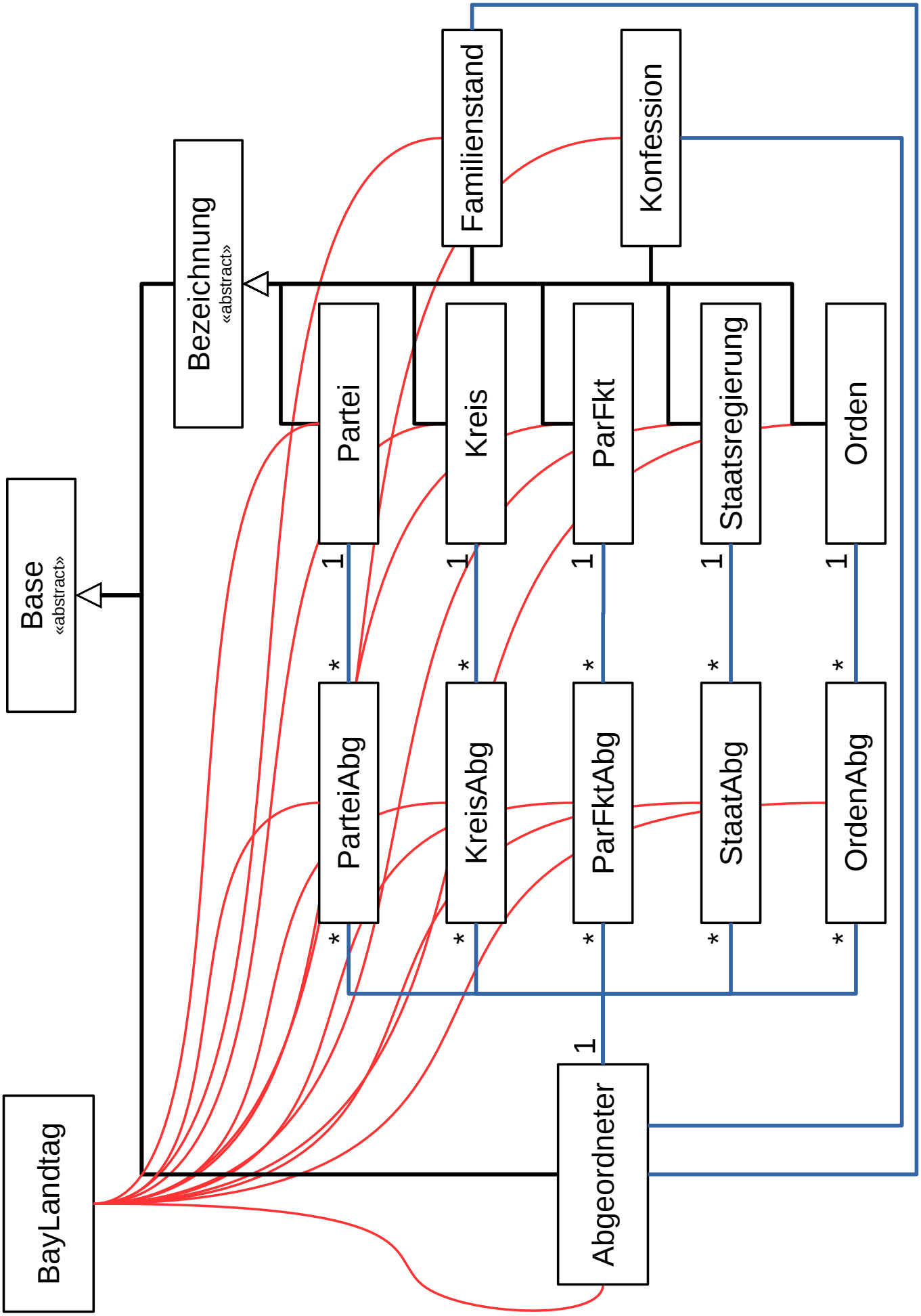
Aufgaben: **f_ltxml_test_03**

1. Nun müssen alle Beziehungen zwischen `Abgeordneter` und den zusätzlichen Daten hergestellt werden. Dazu wird jeweils eine Klasse erzeugt, die die Beziehung und zusätzliche Daten wie Datum-von, Datum-bis etc. beinhaltet.

   Sorgen Sie dafür, das Test `T_11_ParteiAbgTest` ohne Fehler läuft.
2. Sorgen Sie dafür, das Test `T_12_BayLandtagTest` ohne Fehler läuft.
3. Sorgen Sie dafür, das Test `T_13_KreisAbgTest` ohne Fehler läuft.
4. Sorgen Sie dafür, das Test `T_14_ParFktAbgTest` ohne Fehler läuft.
5. Sorgen Sie dafür, das Test `T_15_StaatAbgTest` ohne Fehler läuft.
6. Sorgen Sie dafür, das Test `T_16_OrdenAbgTest` ohne Fehler läuft.
7. Sorgen Sie dafür, das Test `T_17_BayLandtagTest` ohne Fehler läuft.
8. Testen Sie alle bisherigen Tests mit Hilfe von `T_18_00_17_Test`.


BayLandtag
Base
«abstract»
Bezeichnung
«abstract»
ParteiAbg
Partei
KreisAbg
Kreis
Abgeordneter
ParFktAbg
ParFkt
StaatAbg
Staatsregierung
OrdenAbg
Orden
*
1


aufpassen


BayLandtag
Base
«abstract»
Bezeichnung
«abstract»
Familienstand
Konfession
Partei
Kreis
ParFkt
Staatsregierung
Orden
ParteiAbg
KreisAbg
ParFktAbg
StaatAbg
OrdenAbg
Abgeordneter
1
*

Aufgaben: **f_ltxml_test_04**

1. Im letzten Schritt wird bei der Klasse `Abgeordneter` noch ein Bild eingefügt. Das Bild wird dabei als `byte`-Array gespeichert. JAXB speichert `byte`-Arrays in Textform mit einer Base64-Kodierung.

   Base64 ist ein Verfahren, welches Bytes (8-Bit-Binärdaten) in eine Zeichenfolge, die nur aus lesbaren ASCII-Zeichen besteht, umwandelt.

   Sorgen Sie dafür, das Test `T_19_AbgeordneterTest` ohne Fehler läuft.

2. Sorgen Sie dafür, das Test `T_20_BayLandtagTest` ohne Fehler läuft.

2. Testen Sie alle bisherigen Tests mit Hilfe von `T_21_00_20_Test`.

Base64

Bei der Kodierung werden drei Bytes (3 x 8 = 24 bit) in vier Zeichen (4 x 6 = 24 bit: $2^6 = 64$, daher Base64) aufgeteilt. Dabei werden die 6-bit-Zeichen wie folgt kodiert (druckbare ASCII-Zeichen): (0 ⟶ A, 1 ⟶ B, ... , 25 ⟶ Z, 26 ⟶ a, ... , 51 ⟶ z, 51 ⟶ 0, ... , 62 ⟶ +, 63 ⟶/)[1]

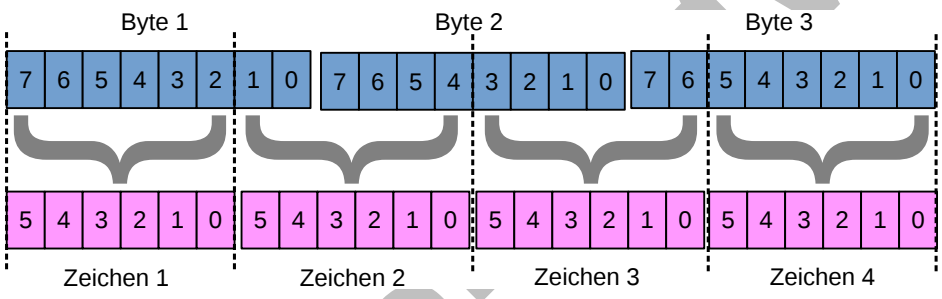


Abb. 3.1: Base64-Kodierung

Beispiel:

```
echo "Dies ist ein Text!" | base64
```

```
RGllcyBpc3QgZWluIFRleHQhCg==
```

und zurück:

```
echo "RGllcyBpc3QgZWluIFRleHQhCg==" | base64 -d
```

```
Dies ist ein Text!
```

[1] Ist die Anzahl an Bytes nicht durch drei teilbar, so wird mit Nullbits aufgefüllt. Pro aufgefüllten Byte wird ein '=' angehängt.

45

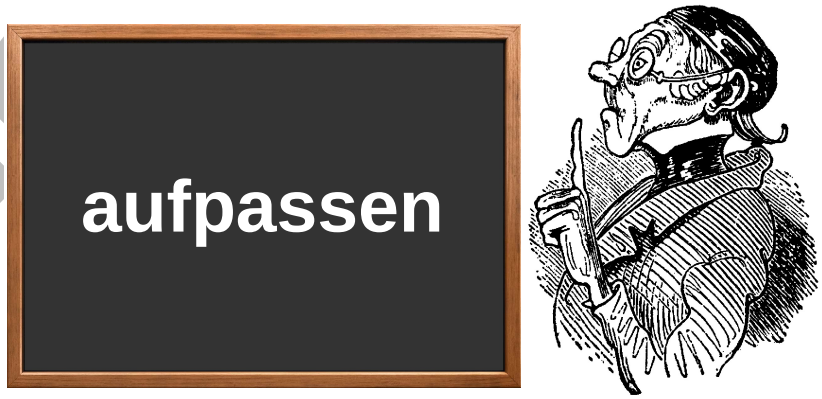
Aufgaben: f_ltxml_csv_01

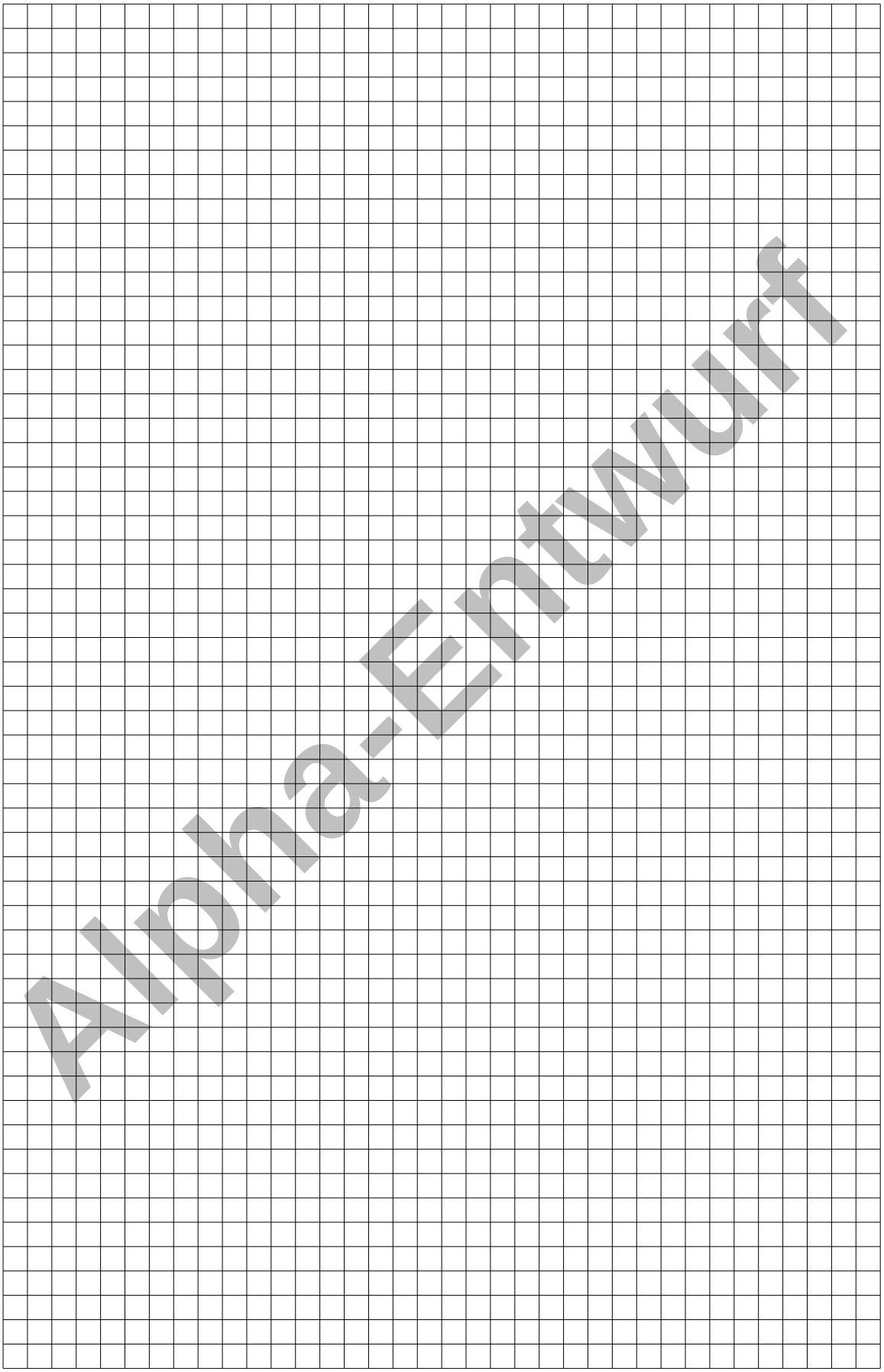
1. In den CSV-Dateien sind jeweils IDs mit Nummern angegeben. Für XML ist es aber wichtig, dass die IDs über alle Elemente eindeutig ist. Daher ist es notwendig, die IDs mit eine Prefix zu versehen, so dass diese eindeutig werden.

   Überlegen Sie sich für jeden Bereich einen klaren Prefix, beispielsweise 'a' für die Abgeordneten (ID '12' bei einem Abgeordneten wird dann zu 'a12', ...).

   Die „fertige" XML-Datei speichern Sie unter `target/landtag.xml`.

2. Parsen Sie die Datei `familienstand.csv` und „füllen" die entsprechenden Objekte. Schauen Sie in den jeweiligen Testtreibern nach, wie dort die Objekte „mit Leben" gefüllt worden sind.
3. Parsen Sie die Datei `konfession.csv`.
4. Parsen Sie die Dateien `abgeordneter.csv` und `bild.csv`.
5. Parsen Sie die Datei `kreis.csv`.
6. Parsen Sie die Datei `orden.csv`.
7. Parsen Sie die Datei `parfkt.csv`.
8. Parsen Sie die Datei `partei.csv`.
9. Parsen Sie die Datei `staatsregierung.csv`.
10. Parsen Sie die Dateien `zt_fkt_abg.csv`, `zt_kreis_abg.csv`, `zt_ord_abg.csv`, `zt_partei_abg.csv` und `zt_reg_abg.csv`.





45

Aufgaben: **f_ltxml_csv_02**

1. Nach der ersten Überprüfung wurde festgestellt, dass der Bereich der Legislaturperiode vergessen worden ist.

   Bauen Sie diesen noch ein. Beachten Sie aber, dass die Tests `T_21_00_20_Test` weiter funktionieren müssen. Als Klassenname verwenden Sie dabei `Wahlperiode`.

```
"id","von","bis"
"1","1946-12-16","1950-11-20"
"2","1950-12-11","1954-11-23"
"3","1954-12-13","1958-11-17"
"4","1958-12-04","1962-10-25"
"5","1962-12-07","1966-11-09"
"6","1966-12-02","1970-10-01"
"7","1970-12-03","1974-10-29"
"8","1974-11-12","1978-09-20"
"9","1978-10-30","1982-07-23"
"10","1982-10-22","1986-07-24"
"11","1986-10-22","1990-07-29"
"12","1990-10-24","1994-07-21"
"13","1994-10-20","1998-07-09"
"14","1998-09-28","2003-07-10"
"15","2003-10-06","2008-07-17"
"16","2008-10-20","2013-07-18"
"17","2013-09-15","2018-07-31"
```

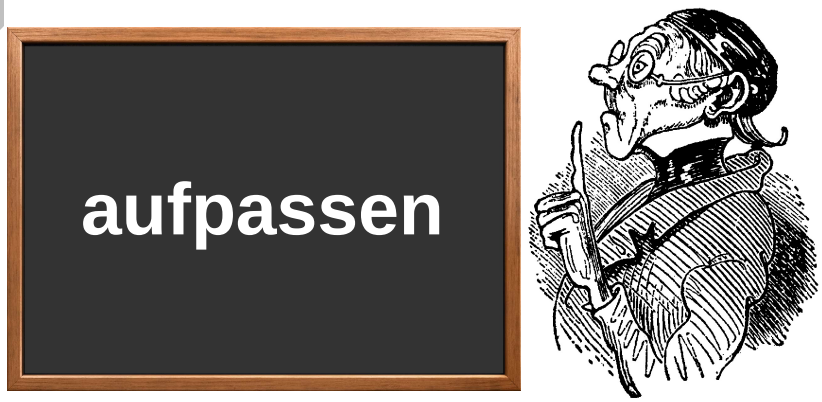
2. Die Daten bzgl. Datum habe eine Besonderheit.

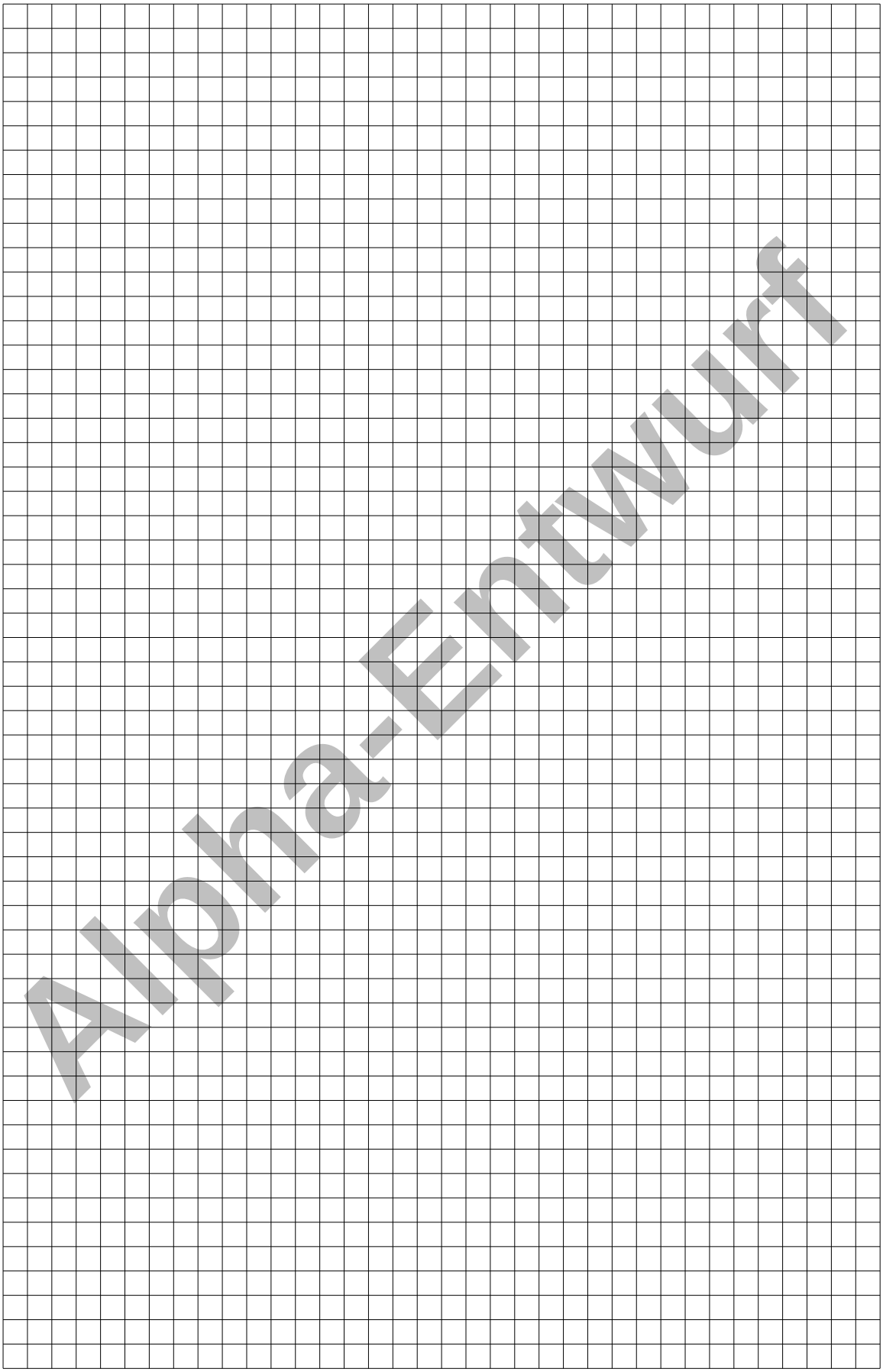
```
seit 20.10.2008   CSU
seit 20.10.2008   Stimmkreis Dachau
```

   Ist ein Abgeordneter über mehrere Legislaturperioden bis heute in einer Partei, einem Stimmkreis, ..., so wird nur das Anfangsdatum angegeben. Das Enddatum hat dabei den selben Wert und muss daher für weitere Anzeigen etc. mit dem Enddatum der aktuellen Legislaturperiode belegt werden.

```
<krbg abg="a1060" bis="2008-10-20T00:00:00+02:00" kreis="kr774"
                  von="2008-10-20T00:00:00+02:00" id="krbg1437"/>
```

   Bauen Sie daher in die Klasse `BayLandtag` eine Methode mit den Namen `calculate()` ein, die bei allen von-bis-Bereichen das Enddatum auf das Ende der Legislaturperiode setzt, wenn es dem Startdatum entspricht.





49

Aufgaben: **f_ltxml_tree_01**

1. Die Abgeordneten sollen sortiert aufgelistet werden (Name, Vorname). Daher soll die Klasse `Abgeordneter` das Interface `Comparable<Abgeordneter>` implementieren.

   Verwenden Sie dazu die Klasse `Collator` als Konstante in der Klasse `Abgeordneter` (siehe Seite 69).

```
/** Sortierung */
private static final Collator COLLATOR = Collator.getInstance(Locale.GERMAN);

static {
   COLLATOR.setStrength(Collator.SECONDARY);// a == A, a < Ä
}
```

2. Erstellen Sie eine Klasse mit dem Namen `de.nm.ltxml.util.Util` und ergänzen Sie die „leeren“ Methoden.

```
/** JAXB-Conntext */
private static JAXBContext context;

/** XML file */
private static final File FILE = new File("src/landtag.xml");

/** create a static context */
static {
   try {
      context = JAXBContext.newInstance(BayLandtag.class);
   } catch (final JAXBException e) {
      e.printStackTrace();
   }
}

/** read BayLandtag */
public static BayLandtag readBayLandtag() throws JAXBException {}

/** read BayLandtag */
public static BayLandtag readBayLandtag(final File file) throws JAXBException {}

/** write BayLandtag */
public static void writeBayLandtag(final BayLandtag lt) throws JAXBException {}

/** write BayLandtag */
public static void writeBayLandtag(final BayLandtag lt, final File file) throws ⤸
JAXBException {}
```

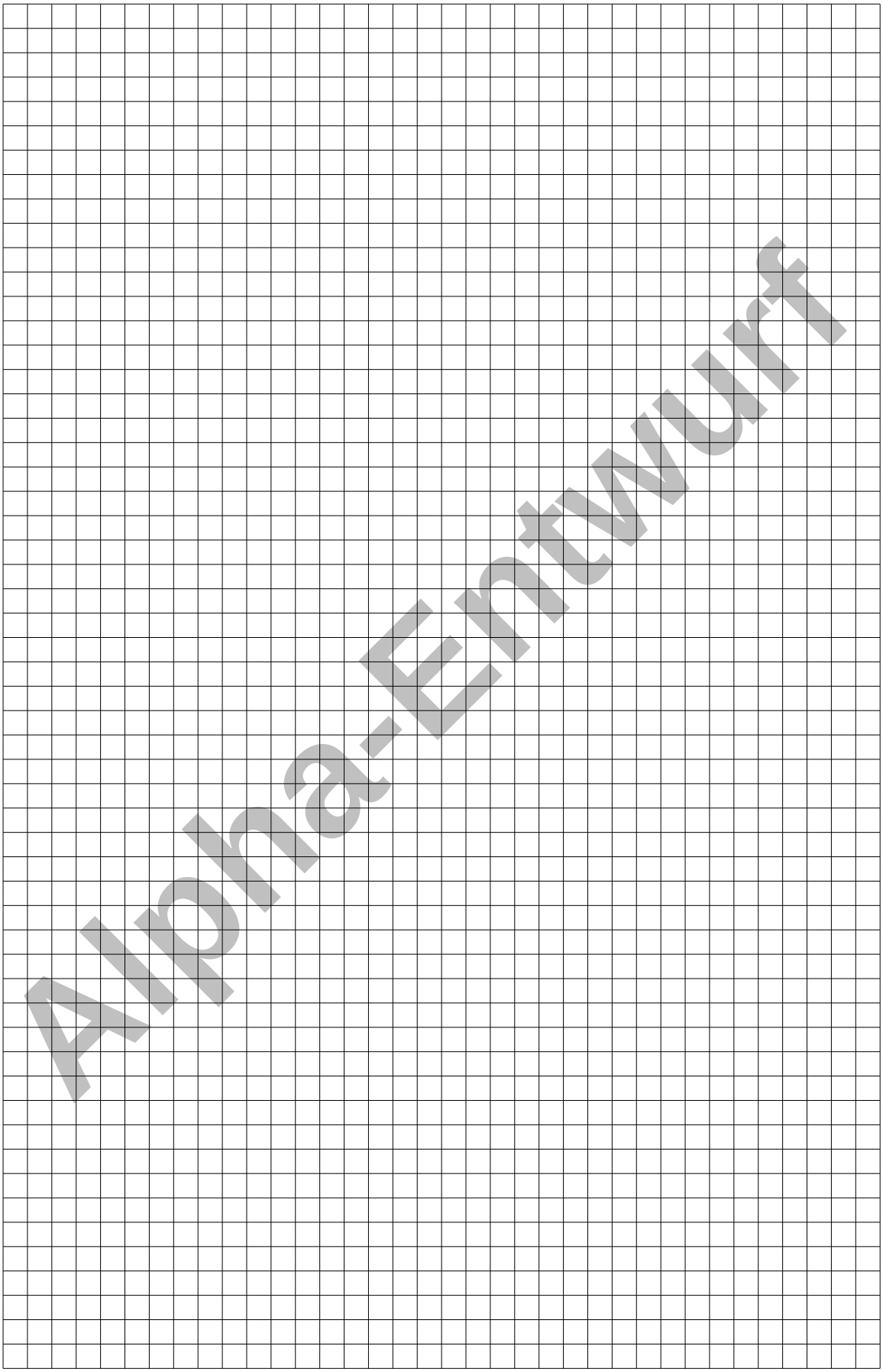
3. Erstellen Sie in der Klasse `Wahlperiode` eine Methode mit dem Namen `getDisplayString()`, die den Anzeigetext zurück liefert.

   Beispiel:
   16. Wahlperiode 2008-10-20 - 2013-07-18

4. Erstellen Sie in der Klasse `Abgeordneter` eine Methode mit dem Namen `getDisplayString()`, die den Anzeigetext zurück liefert.

   Beispiel:
   Seidenath, Bernhard

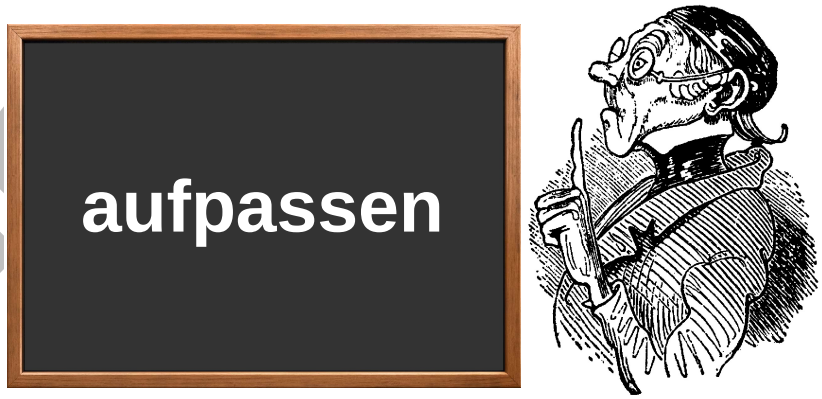
5. Erstellen Sie die Klassen `de.nm.ltxml.gui.v1.LtTree` und `de.nm.ltxml.gui.v1.GuiUtil`.

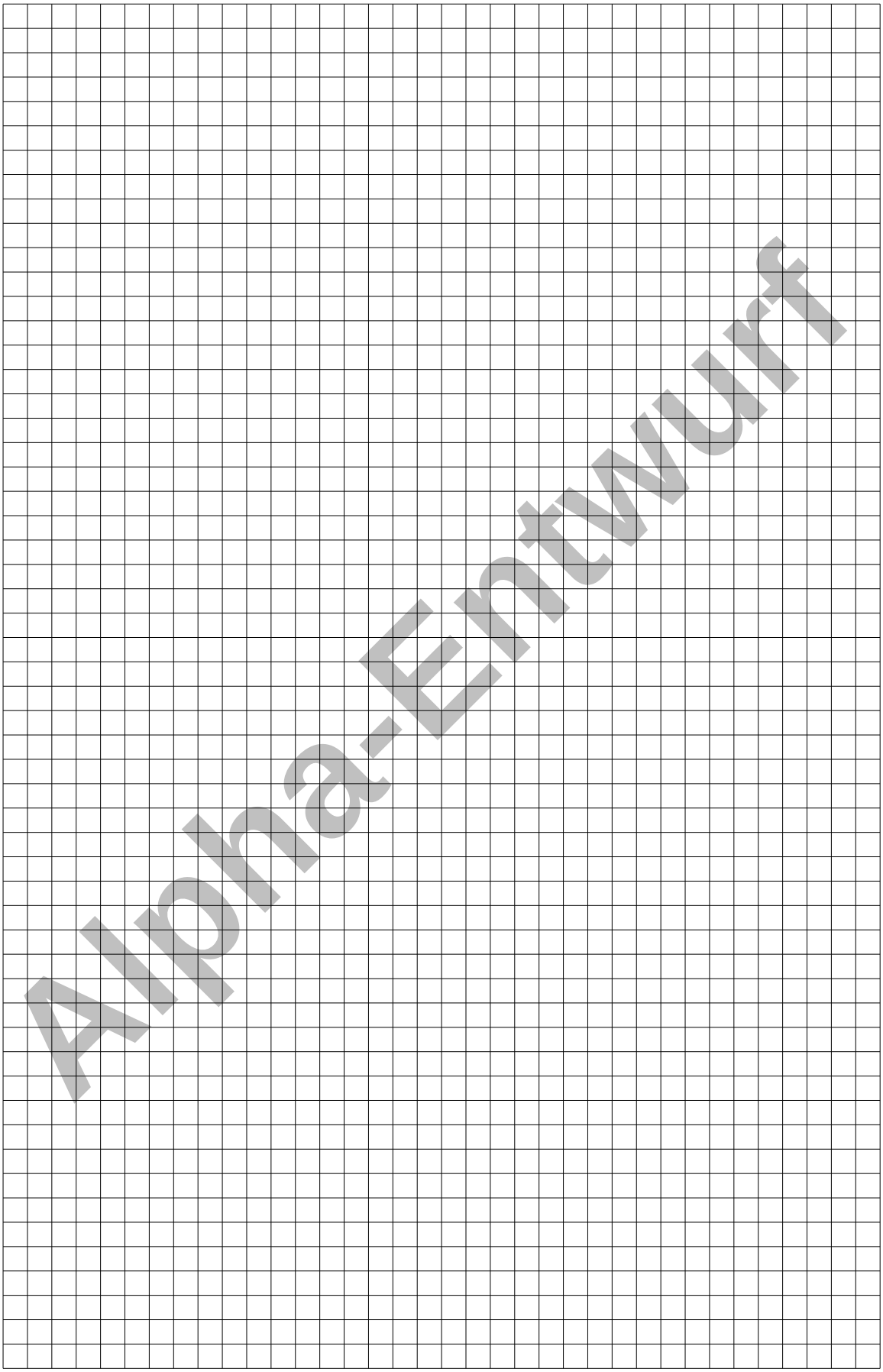


54

Aufgaben: f_ltxml_tree_02

1. Sind Sie mit der `JTabel` nicht vertraut, so lesen Sie zuerst auf Seite 174 weiter.
2. Kopieren Sie die bisherigen Klassen `LtTree` und `GuiUtil` in den neuen Namensraum `de.nm.ltxml.gui.v2`.
3. Ergänzen bzw. passen Sie den Code entsprechend an, dass die Tabelle angezeigt wird.
4. Die Anzeige sieht im Moment sehr bescheiden aus.
   1. Erweitern Sie den Code, so dass der Baum schöner aussieht, beispielsweise entsprechende Icons nach Inhalt anzeigt. Schauen Sie sich das Tutorial von Oralce dazu an.
      `https://docs.oracle.com/javase/tutorial/uiswing/components/tree.html`
   2. Erweitern Sie die Anzeige, so dass möglichst viel Informationen angezeigt werden.
   3. Hat der Abgeordnete ein Bild, so zeigen Sie diese auch an.
5. Suchen und Finden
   1. Überlegen Sie sich, nach welchen Inhalten ein Anwender suchen könnte und bauen Sie entsprechende Funktionen ein.
   2. Dabei sollen nicht nur „normale Inhalte“ über entsprechende Textfelder gesucht werden, sondern mittels Regulärer Ausdrucke auch komplexere Möglichkeiten zur Verfügung stehen (siehe Seite 194).

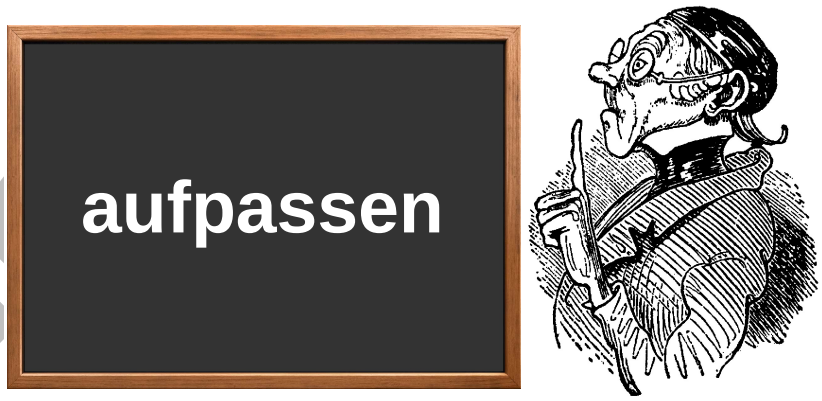


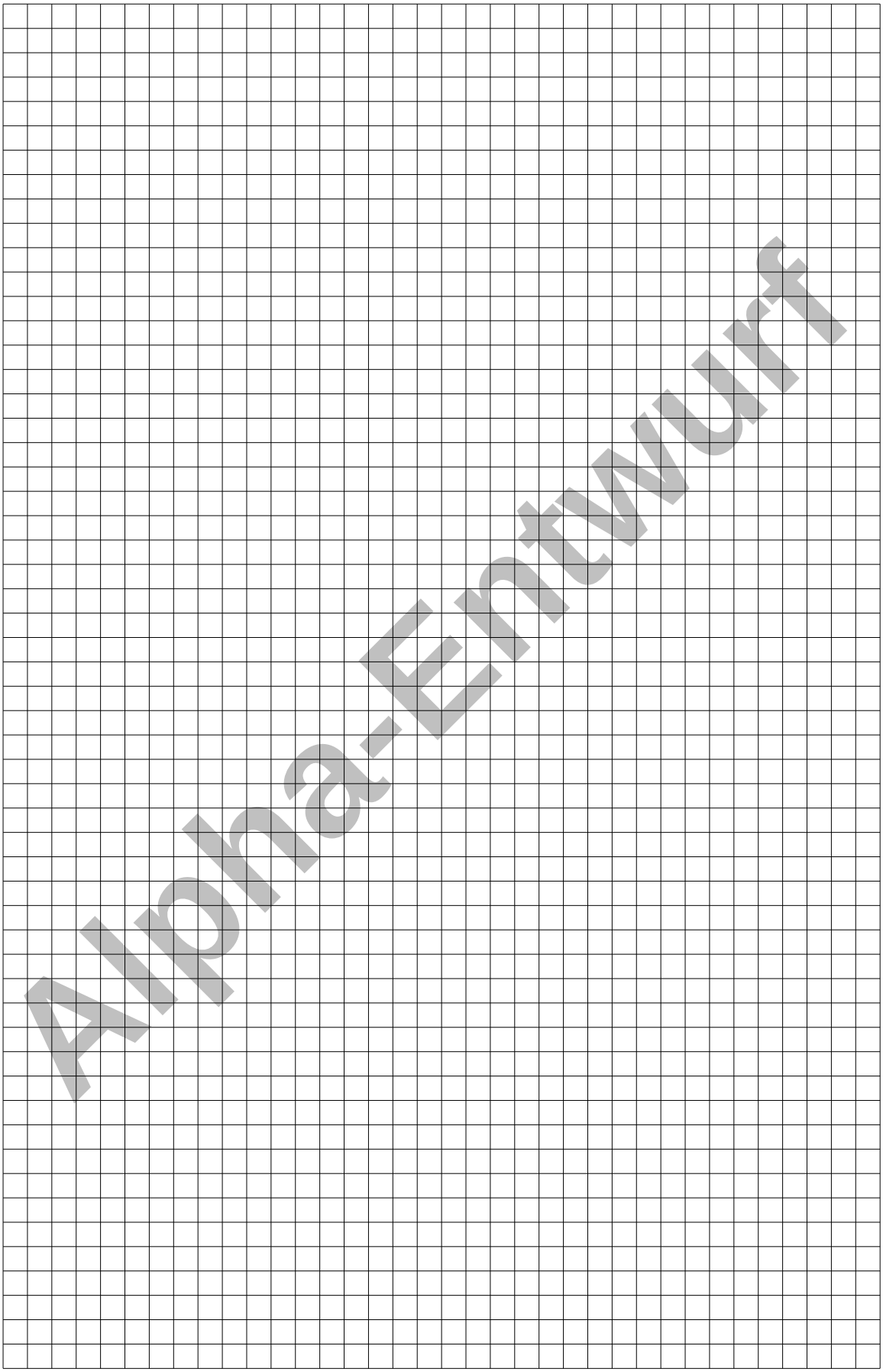


58

Aufgaben (für Gute): f_ltxml_tree_03

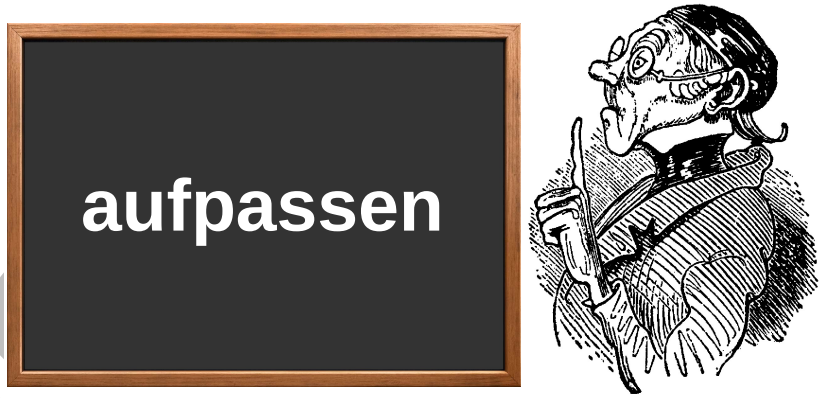
1. Erweitern Sie die Anzeige, so dass wie bei Version 2 möglichst alle Informationen angezeigt werden.
2. Im Moment ist die Baumhierarchie „Wahlperiode“ – „Anfangsbuchstabe“ – „Abgeordneter“.
   Bauen Sie verschiedene Baumhierarchien ein. z. B.
   - „Partei“ – „Wahlperiode“ – „Anfangsbuchstabe“ – „Abgeordneter“
   - „Stimmkreis“ – …
   - „Orden“ – …
   - „Parlamentarische Funktion“ – …
   - „Staatsregierung“ – …
3. Für Profis:
   Bauen Sie mit Hilfe von XSLT einen HTML-Export ein.

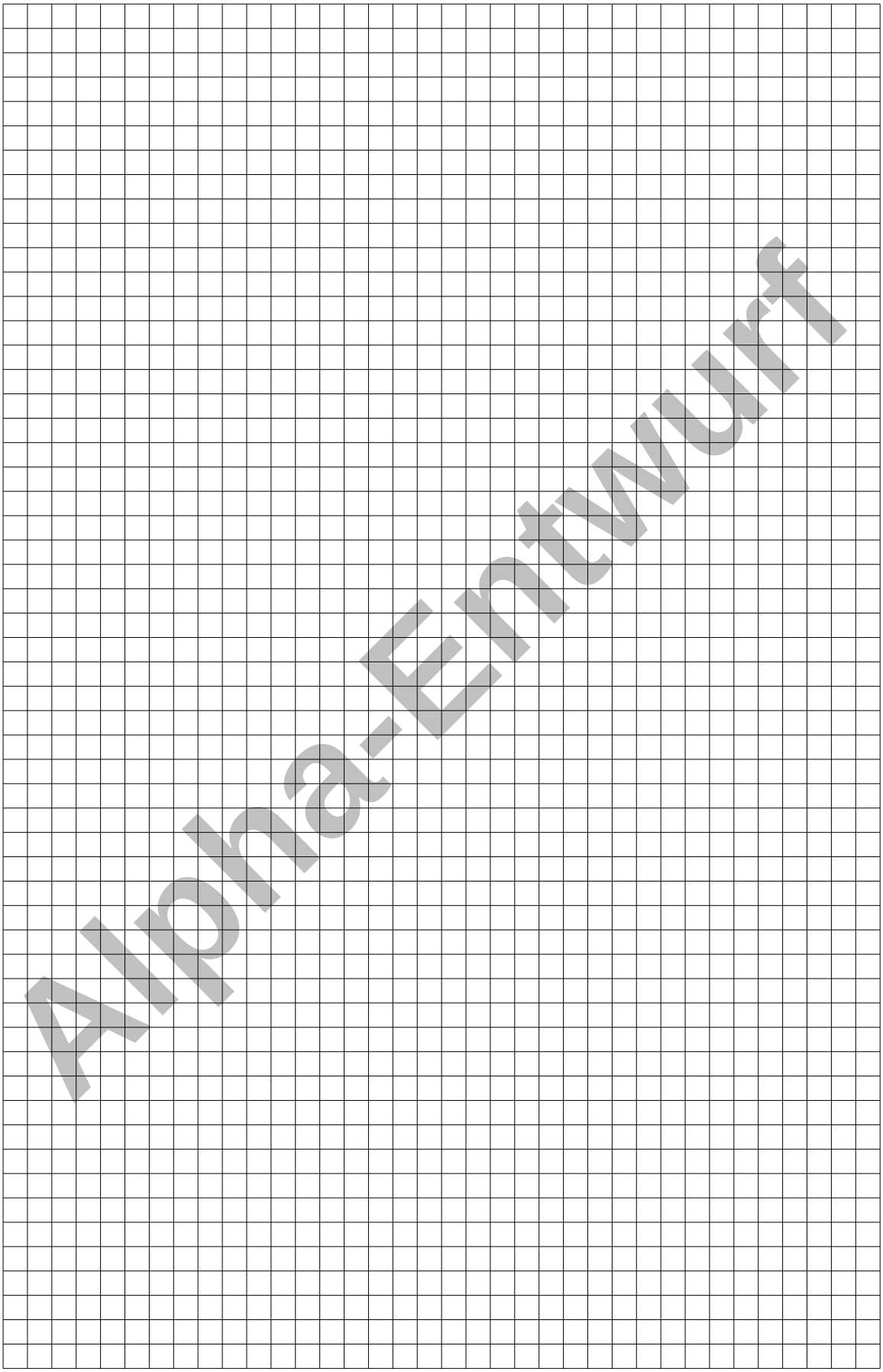




58 **Aufgaben (für Gute):** f_ltxml_tree_04

1. Die XML-Datei des Bayerischen Landtags soll an andere Übertragen werden.
   1. Bauen Sie eine Export-Funktion ein, die die XML-Datei exportiert.
   2. Dabei soll ein geeignetes Kompremierverfahren verwendet werden (siehe Seite 97).
   3. Damit der Empfänger sicher gehen kann, dass die Datei keine Übertragungsfehler hat, soll in einer weiteren Text-Datei die Prüfsumme übertragen werden. Wählen Sie ein entsprechendes Prüfverfahren aus und implementieren Sie diese Funktion(siehe Seite 105).





62

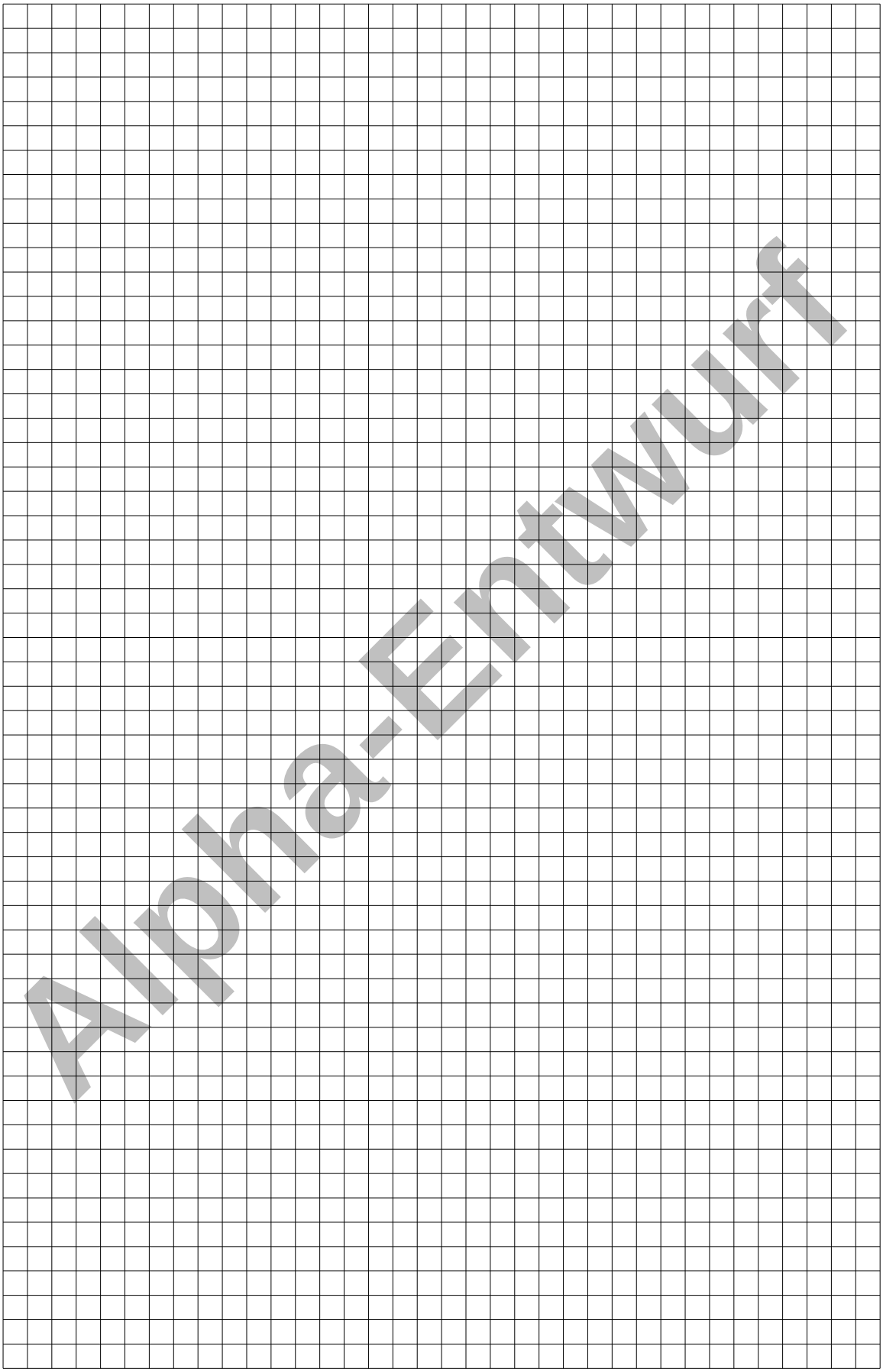
Aufgaben: **f_tools_dc_list_01**

1. Die `ArrayList` verwaltet Objekte (`String`, `Integer`, ...) in einem Array. Bei Objekten werden (siehe Abbildung 9.1.1 auf Seite 62) aber nur die Referenzen direkt hintereinander gespeichert, die einzelnen Werte stehen als Objekte irgendwo im Speicher.

   Verwendet man eine `ArrayList` beispielsweise mit `int`-Werten, so werden diese als `Integer`-Objekte gespeichert und der schnelle Zugriffsvorteil wird gemindert.

   Erstellen Sie daher eine Klasse mit dem Namen `ArrayIntList`, die sich an `ArrayList` orientiert, aber die Werte in einem echten `int`-Array speichert.

   Zur Entwicklung verwenden Sie den Testtreiber `ArrayIntListTest`.



108

Aufgaben: F_TOOLS_OOP_VERERBUNG_01

1. Wie können Sie verhindern, dass von einer Ihrer Klassen abgeleitet wird?
2. Wie greift man in einer Methode einer abgeleiteten Klasse auf geerbte `protected`-Elemente zu?
3. Wie greift man in einer Methode einer abgeleiteten Klasse auf geerbte `private`-Elemente zu?
4. Welche Zugriffsspezifizierer geben geerbte Elemente nach außen weiter?
5. Kann der Konstruktor einer abgeleiteten Klasse den `private`-Elementen seiner Basisklasse Werte zuweisen?
6. Wie lautet die Ausgabe des folgenden Programms? (Programmieren Sie es nicht nach! Überlegen Sie sich die Antwort so.)

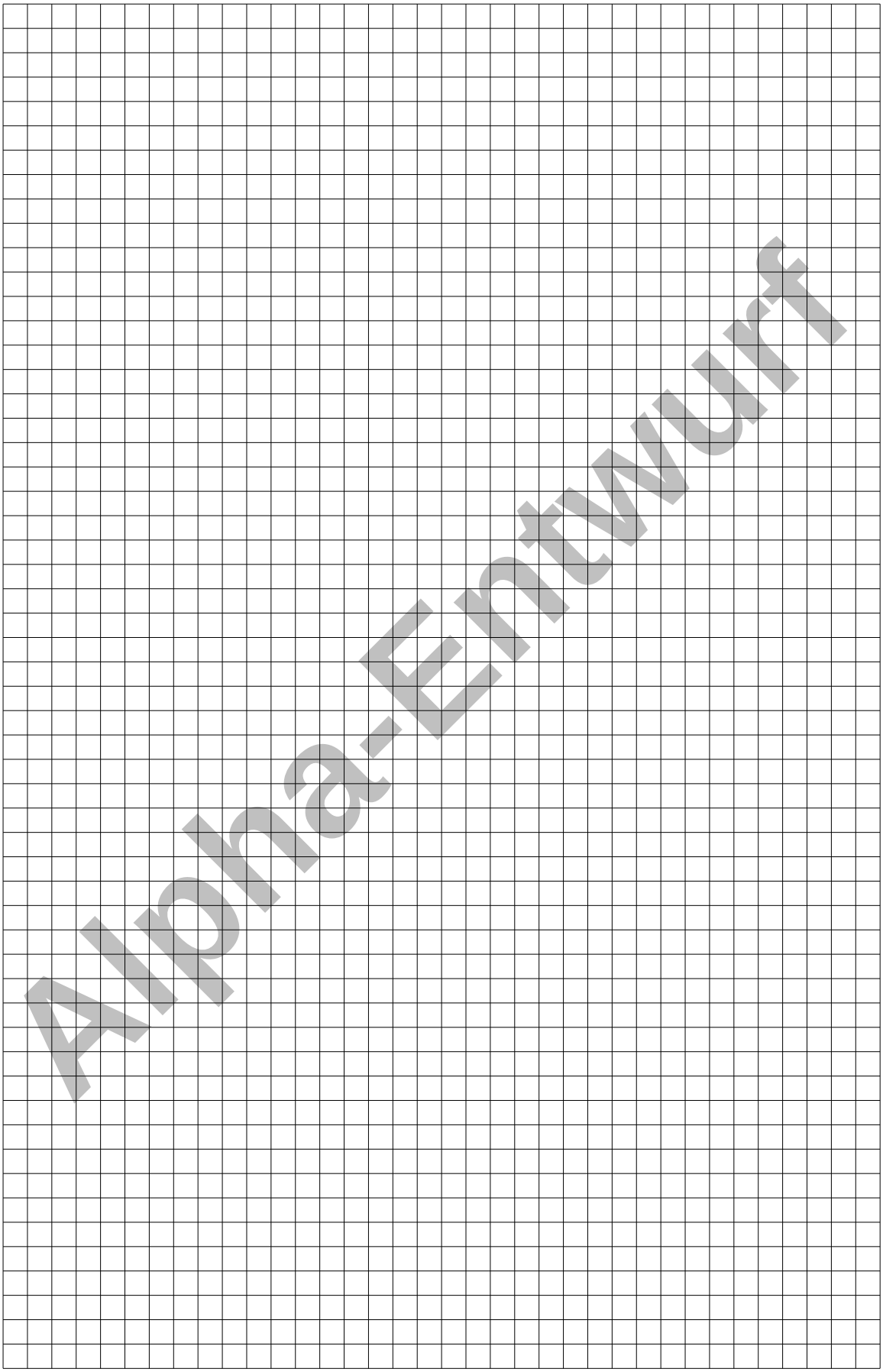
```
static class Basis {
   private int feld;

   public int getFeld() {
      return feld;
   }

   public void setFeld(int wert) {
      feld = wert;
   }
}

static class Abgeleitet extends Basis {
   public int feld;
}

public static void main(String[] args) {
   Abgeleitet obj = new Abgeleitet();
   obj.feld = 12;
   System.out.println(obj.getFeld());
}
```

7. Was ist falsch an dem folgenden Code?

```
static class Basis {
   private int basisFeld;

   Basis(int param) {
      basisFeld = param;
   }
}

static class Abgeleitet extends Basis {
   public Abgeleitet() {
   }
}
```

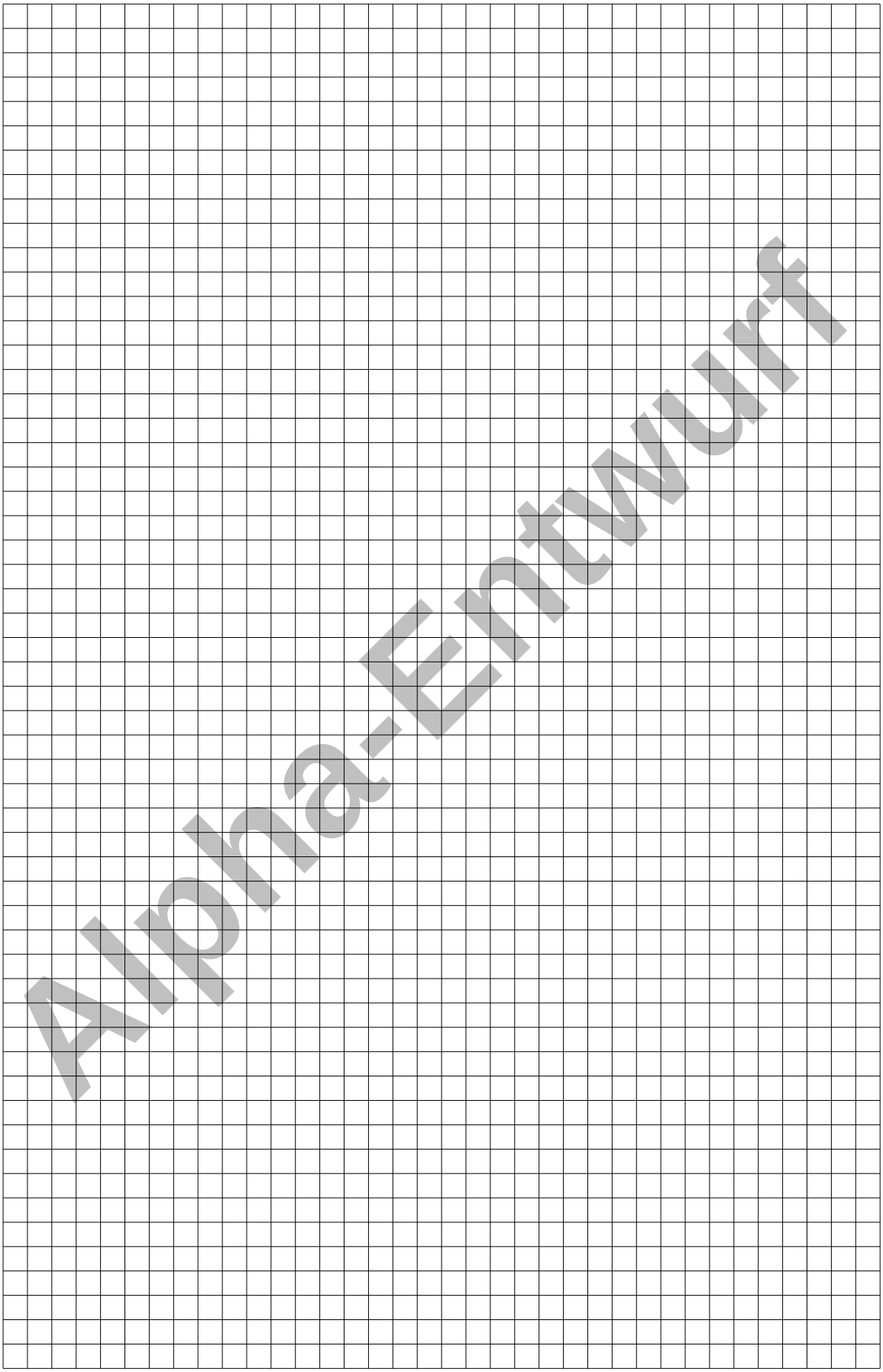
8. Schreiben Sie für die Klasse Abgeleitet einen Konstruktor, der allen eigenen und den geerbten Feldern Anfangswerte zuweist.

```
static class Basis {
   private int basisFeld;

   Basis(int param) {
      basisFeld = param;
   }
}

static class Abgeleitet extends Basis {
   private int abgFeld1;
   private int abgFeld2;

   public Abgeleitet(int param1, int param2, int param3) {


   }
}

public static void main(String[] args) {
   Abgeleitet obj = new Abgeleitet(1, 2, 3);
}
```



110 **Aufgaben:** F_TOOLS_OOP_VERERBUNG_02

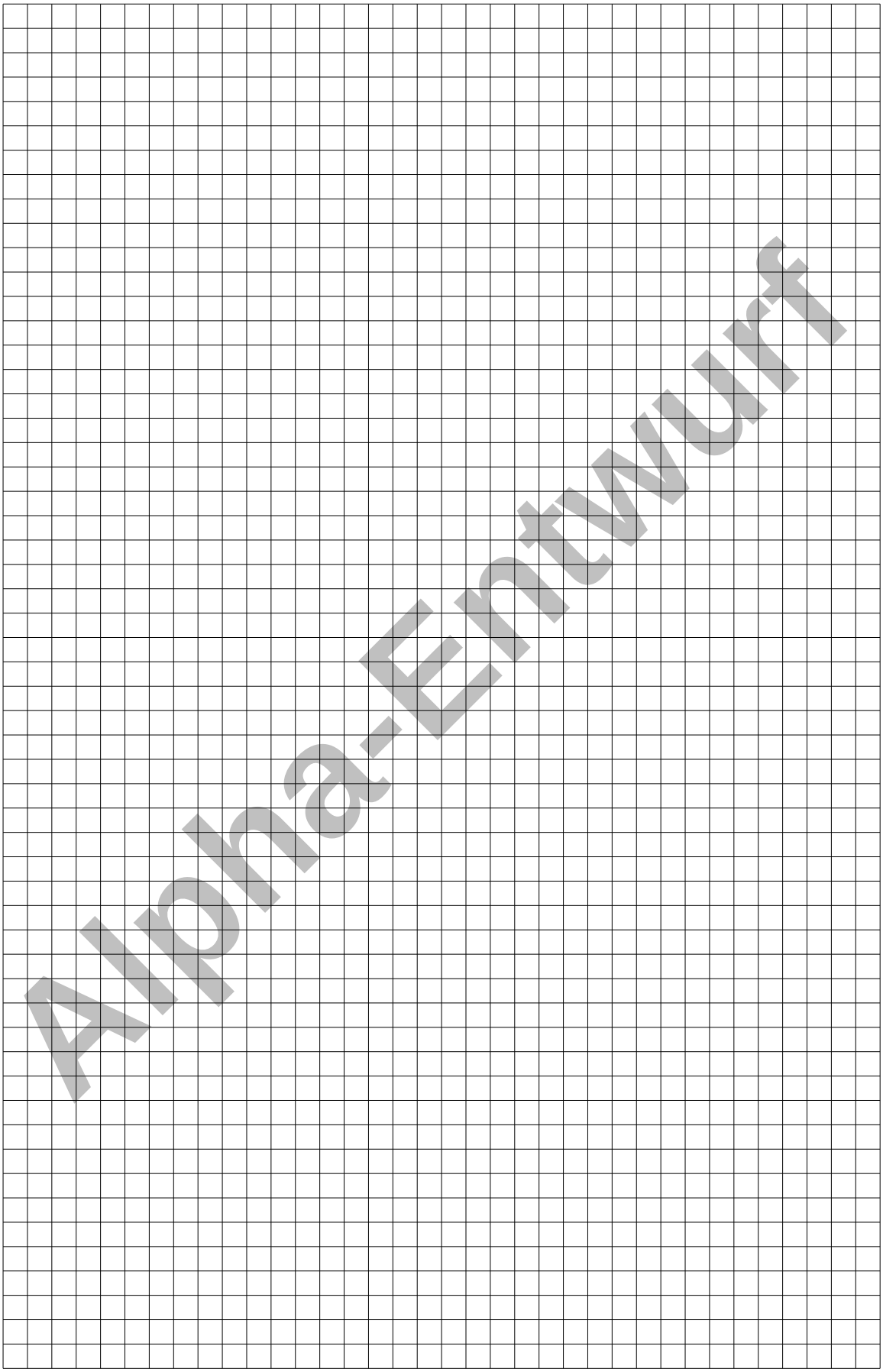
1. Wer hat Zugriff bei `public`?
   - ☐ Zugriff aus Basisklasse
   - ☐ Zugriff aus abgel. Klasse
   - ☐ Zugriff aus Paket
   - ☐ Zugriff von „woanders“
2. Wer hat Zugriff bei `default`?
   - ☐ Zugriff aus Basisklasse
   - ☐ Zugriff aus abgel. Klasse im selben „namespace“
   - ☐ Zugriff aus abgel. Klasse aus anderem „namespace“
   - ☐ Zugriff aus Paket
   - ☐ Zugriff von „woanders“
3. Wer hat Zugriff bei `protected`?
   - ☐ Zugriff aus Basisklasse
   - ☐ Zugriff aus abgel. Klasse
   - ☐ Zugriff aus Paket
   - ☐ Zugriff von „woanders“
4. Wer hat Zugriff bei `private`?
   - ☐ Zugriff aus Basisklasse
   - ☐ Zugriff aus abgel. Klasse
   - ☐ Zugriff aus Paket
   - ☐ Zugriff von „woanders“



113

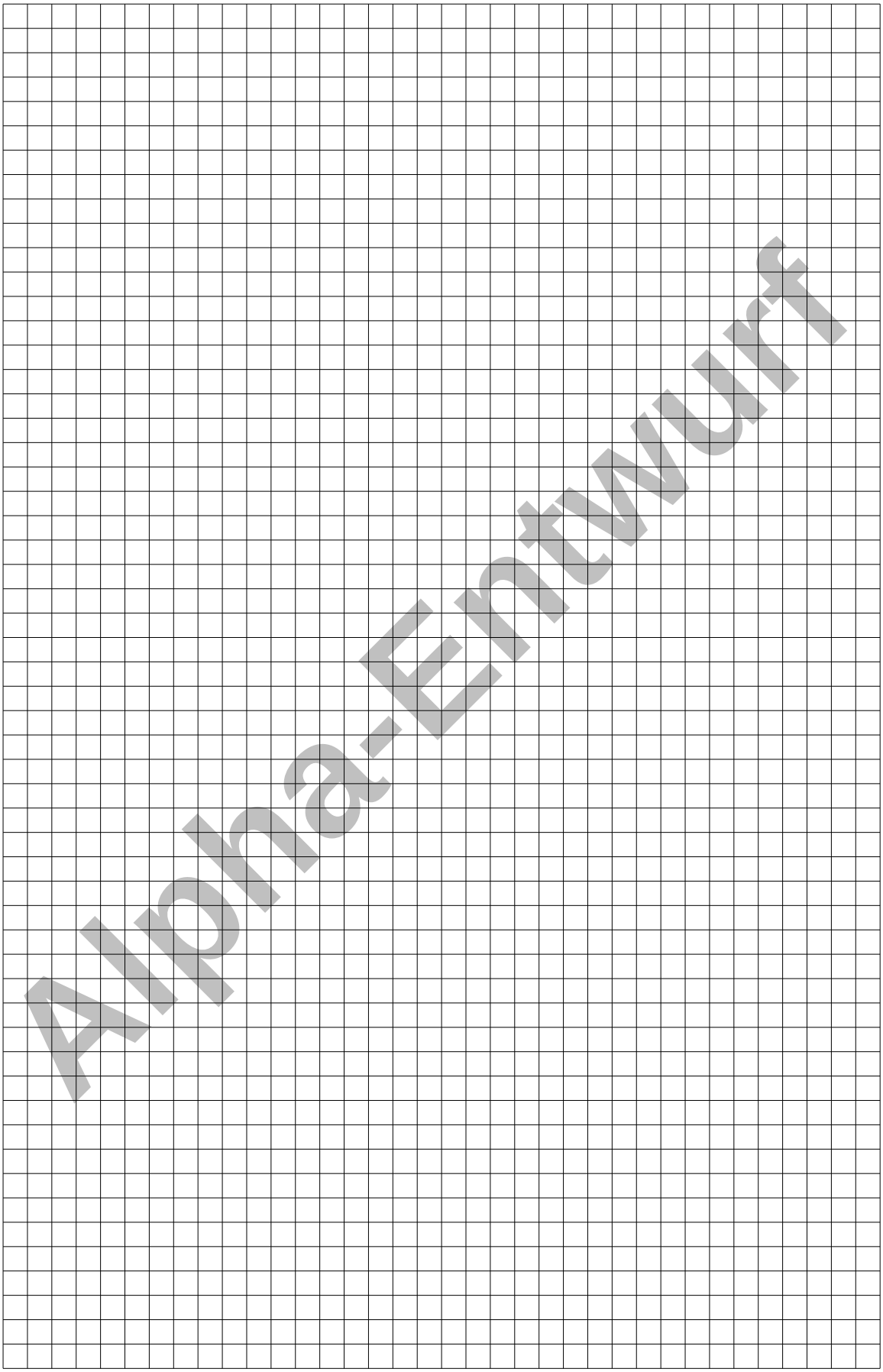
Aufgaben: F_TOOLS_OOP_VERERBUNG_03

1. Was versteht man unter „Verdecken“?
2. Was versteht man unter „Überschreibung“?
3. Was versteht man unter „Überladung“?



108-117 **Aufgaben:** **F_TOOLS_OOP_VERERBUNG_04**

1. Welche der Aussagen über Konstruktoren ist richtig?
   - ☐ Gibt es in einer Klasse keinen Konstruktor, so wird vom Compiler ein Leer-Konstruktor erzeugt.
   - ☐ Ein Konstruktor hat immer den Zugriffsbezeichner `private`.
   - ☐ Ein Konstruktor hat immer den Rückgabewert `void`.
   - ☐ Es darf pro Klasse immer nur einen Konstruktor geben.
   - ☐ Ein Basisklassenkonstruktor darf keine Parameter haben.
   - ☐ Ein Konstruktor mit Parametern hat immer den Zugriffsbezeichner `protected`.
   - ☐ Ein Konstruktor ohne Parametern hat immer den Zugriffsbezeichner `default`.
2. Welche der Aussagen über Zugriffsbezeichner ist richtig?
   - ☐ Auf Methoden, die `public` sind, kann nur die eigene Klasse zugreifen.
   - ☐ Die Zugriffsbezeichner `protected` und `default` bedeuten das selbe.
   - ☐ Auf Methoden, die `private` sind, kann nur von ihrer eigenen Klasse zugegriffen werden.
   - ☐ Auf Methoden, die `protected` sind, kann nur von ihrer eigenen Klasse zugegriffen werden.
   - ☐ Auf Methoden, die `public` sind, kann von überall zugegriffen werden.
3. Welche der Aussagen ist richtig?
   - ☐ Der Name des Packages steht noch vor der Klassendefinition.
   - ☐ Der Name des Packages steht in der Klasse ganz oben.
   - ☐ Die Pakete, die importiert werden, stehen gleich nach er Packages-Definition.
   - ☐ Der Package-Name wird vom Compiler anhand der Verzeichnisstruktur ermittelt und muss nicht angegeben werden.
4. Welche Codezeilen für Konstruktoren sind richtig?
   - ☐ `public class Base { Base void() {};}`
   - ☐ `public class Base { Base() {};}`
   - ☐ `public class Base { base() {};}`
   - ☐ `public class Base { Base(int a) {};}`
   - ☐ `public class Base { int Base() {};}`
5. Welche der Aussagen ist richtig?
   - ☐ Die Beziehung zwischen einer Superklasse und einer Subklasse nennt man „Ist ein“.
   - ☐ Die Beziehung zwischen einer Superklasse und einer Subklasse nennt man „Hat ein“.
   - ☐ Ein Subklasse erbt Methoden von der Superklasse.
   - ☐ In Java gibt es keine Vererbungsstrukturen.
   - ☐ Ist der Code korrekt im Bezug auf Vererbung von Klassen?
     ```
     public class Auto extends Fahrzeug {}
     ```



108-117 **Aufgaben:** **F_TOOLS_OOP_VERERBUNG_05**

1. Ist nachfolgender Code richtig? Geben Sie eine Begründung an!

```
static class Basis {
   public int getFeld() {
      return 5;
   }
}

static class Abgeleitet extends Basis {
   public double getFeld() {
      return 5;
   }
}
```

2. Ist nachfolgender Code richtig? Geben Sie eine Begründung an!

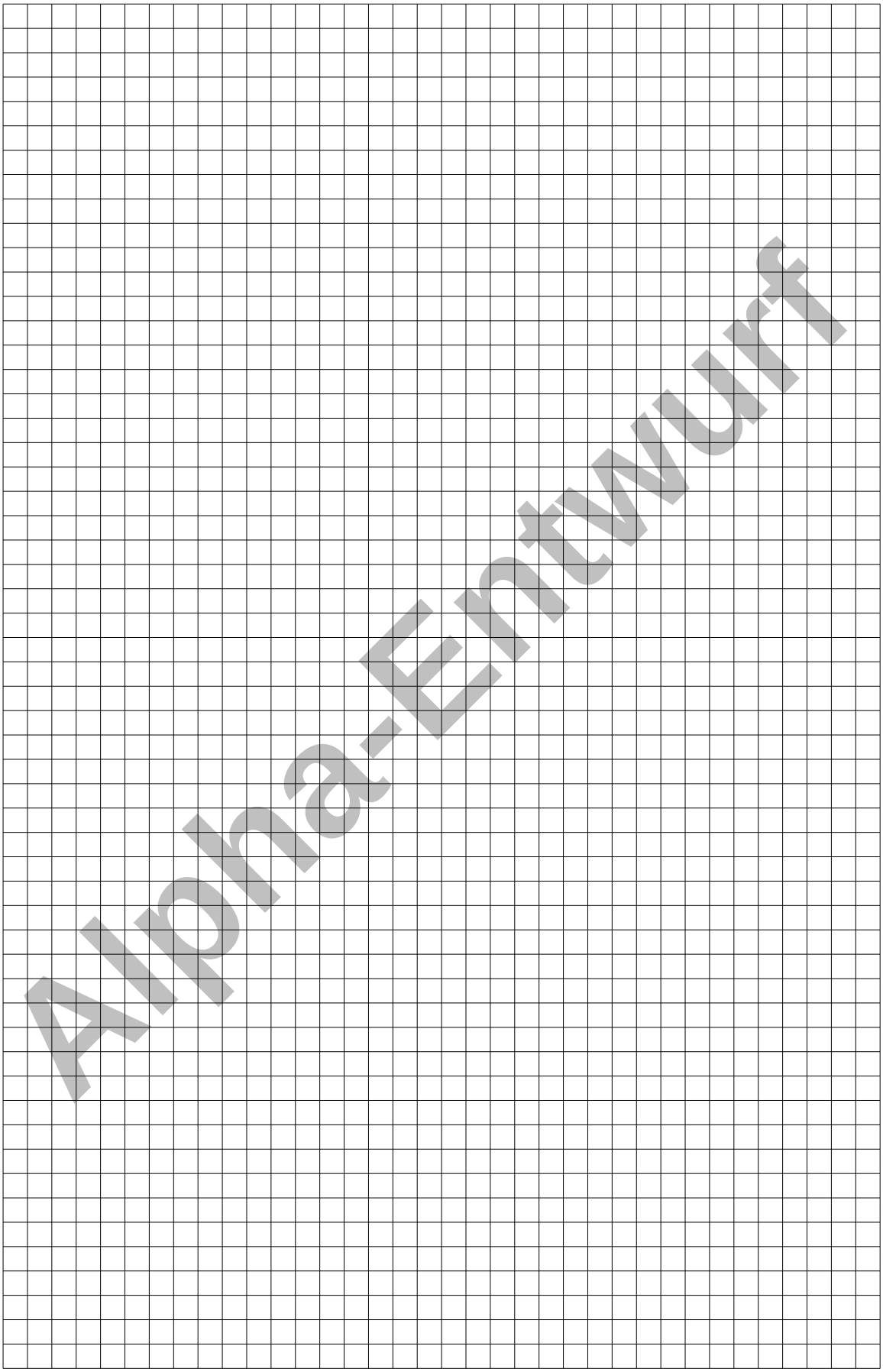
```
static class Basis {
   public int getFeld() {
      return 5;
   }
}

static class Abgeleitet extends Basis {
   protected int getFeld() {
      return 5;
   }
}
```

3. Wie nennt man folgende Beziehung?

```
class Tier {}
class Saeugetier extends Tier {}
class Hase extends Saeugetier {}
```

4. Wie nennt man folgende Beziehung?

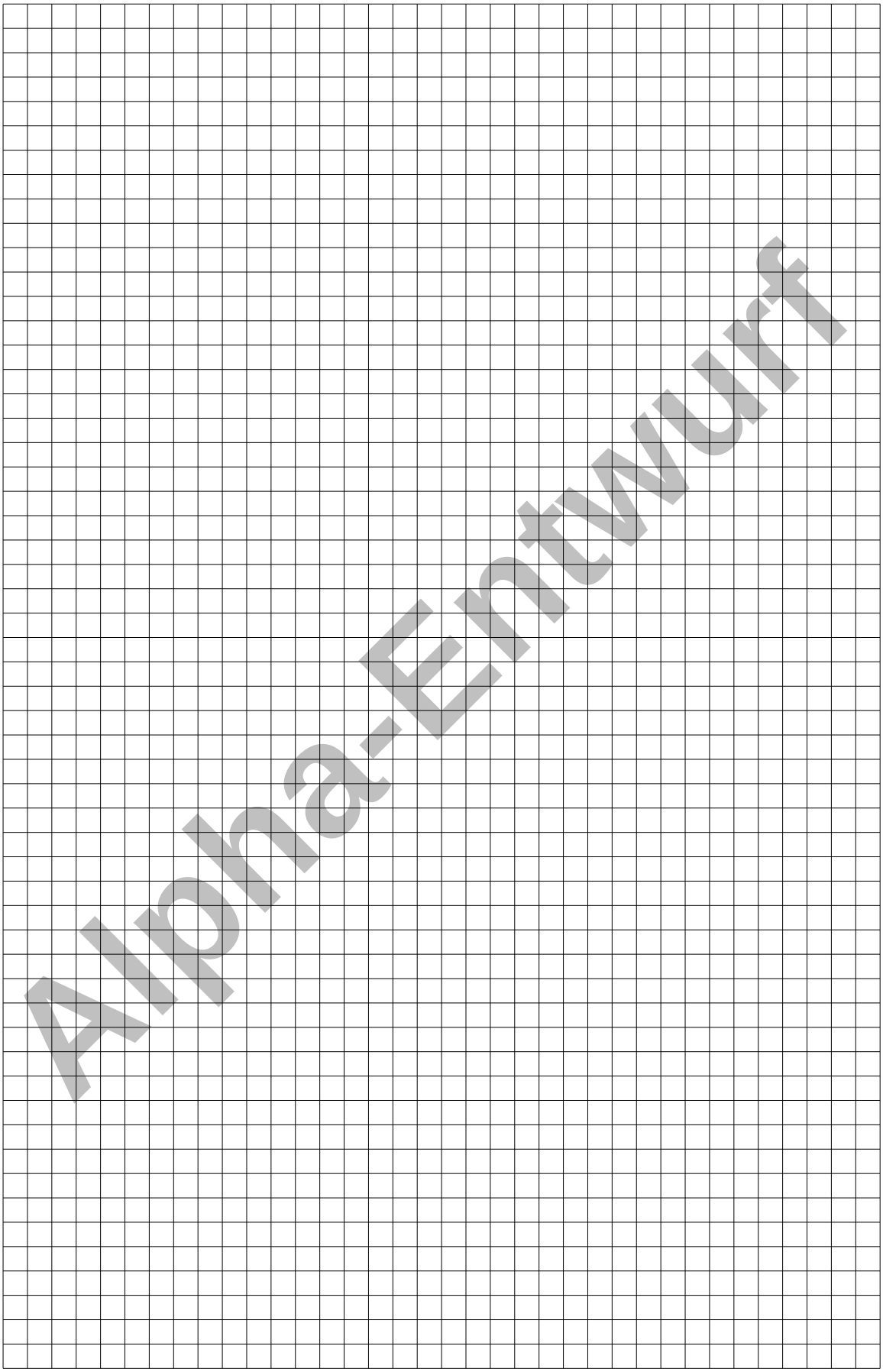
```
class Besitzer {}
class Tier {}
class Saeugetier extends Tier {}
class Hase extends Saeugetier {
   private Besitzer b;
}
```



118 **Aufgaben:** **F_TOOLS_OOP_POLY_01**

1. Welche der Bemerkungen treffen zu?
   - ☐ Methoden aus der Basisklasse müssen in der abgeleiteten Klasse implementiert werden
   - ☐ Private Methoden können überschrieben werden
   - ☐ Final Methoden können überschrieben werden
   - ☐ Static Methoden können überschrieben werden
   - ☐ Keine Methoden der basisklasse können in der abgeleiteten Klasse überschrieben werden
   - ☐ Überschreiben ist ein synonym für Überladen
   - ☐ Beim Überladen müssen die Methoden als `public` gekennzeichnet werden
   - ☐ Keine dieser Möglichkeiten
2. Welche der Bemerkungen treffen zu?
   - ☐ Eine abgeleitete Klasse erbt die Methoden aus der Basisklasse
   - ☐ Eine `protected`-Methode in der Basisklasse darf in der abgeleiteten Klasse als `public` deklariert werden
   - ☐ Final Methoden dürfen nicht überschrieben werden
   - ☐ Methoden aus der Basisklasse müssen in der abgeleiteten Klasse implementiert werden
   - ☐ Methoden aus der Basisklasse können in der abgeleiteten Klasse überschrieben werden
   - ☐ Beim Überladen von Methoden darf ich nicht öffentlicher werden (`protected` -> `public`)
   - ☐ Beim Überladen von Methoden darf ich privater werden (`public` -> `protected`)
   - ☐ Keine dieser Möglichkeiten
3. Was wird ausgegeben?

```
static class Mutter {
   void print() {
      System.out.println("Mutter");
   }
}

static class Sohn extends Mutter {
}

public static void main(final String[] args) {
   final Mutter m = new Mutter();
   final Sohn s = new Sohn();
   m.print();
   s.print();
}
```



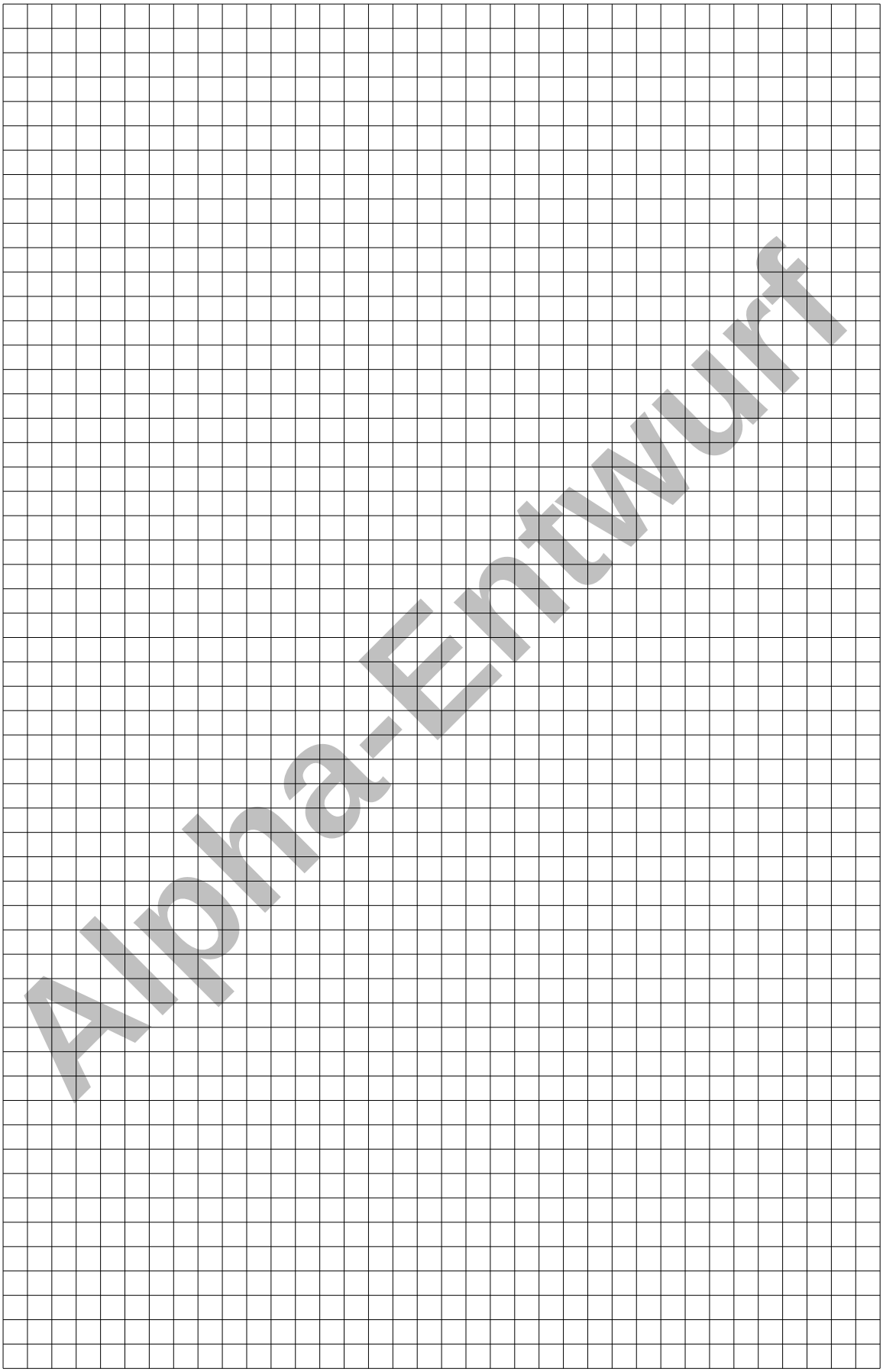
118 **Aufgaben:** **F_TOOLS_OOP_POLY_02**

1. Was wird ausgegeben?

```
static class Mutter {
   void print() {
      System.out.println("Mutter");
   }
}

static class Sohn extends Mutter {
   void print() {
      System.out.println("Sohn");
   }
}

public static void main(final String[] args) {
   final Mutter m = new Mutter();
   final Sohn s = new Sohn();
   m.print();
   s.print();
}
```

2. Was wird ausgegeben?

```
static class Mutter {
   void print() {
      System.out.println("Mutter");
   }
}

static class Sohn extends Mutter {
   void print() {
      System.out.println("Sohn");
   }
}

public static void main(final String[] args) {
   final Mutter m = new Mutter();
   final Mutter s = new Sohn();
   m.print();
   s.print();
}
```



142 **Aufgaben:** f_XML_01

1. Geben Sie die ersten Spieler (<5) in einer Tabelle aus und summieren die Monatsgehälter.

Spielerliste

S-ID	Name	Vorname	Gehalt
1	Lahm	Philpp	910000.00
2	Badstuber	Holger	420970.00
3	Ribéry	Franck	857500.00
4	Martínez	Javier	900000.00
			3088470

2. Bei der vorherigen Ausgabe ist die Summe ohne Kommastellen und Format ausgegeben. Bauen Sie hier das Element `<xsl:decimal-format>` (Name „euro“) und die Funktion `format-number(number, format, locale)` bei alle Geldbeträgen ein.

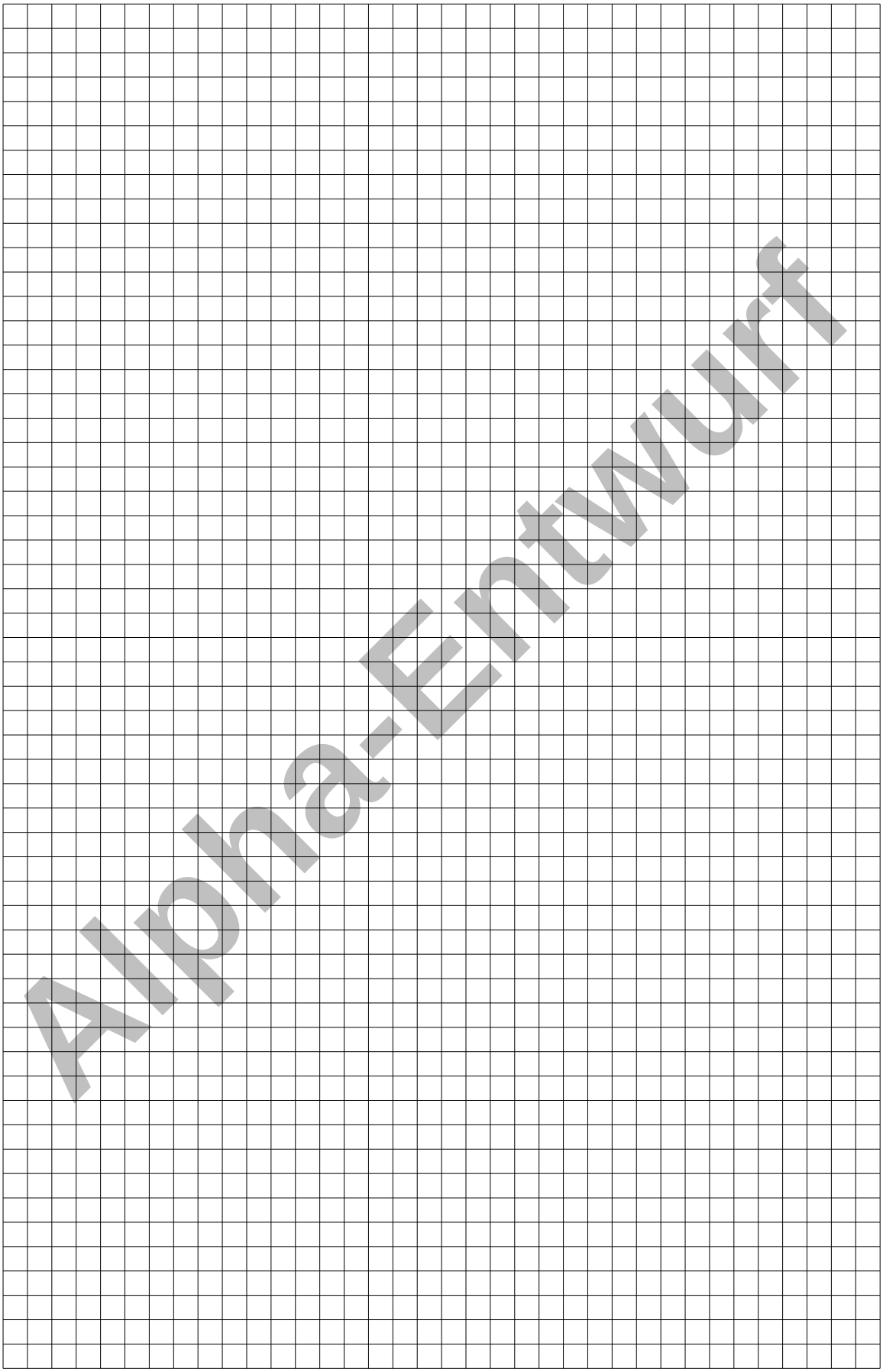
Spielerliste

S-ID	Name	Vorname	Gehalt
1	Lahm	Philpp	910.000,00
2	Badstuber	Holger	420.970,00
3	Ribéry	Franck	857.500,00
4	Martínez	Javier	900.000,00
			3.088.470,00

3. Geben Sie zu jedem Spieler die Mannschafts-ID und den Mannschaftsnamen aus. Verwenden Sie dazu `<xsl:variable>`, um sich die ID der Mannschaft zu merken.

Spielerliste

S-ID	Name	Vorname	M-ID	Mannschaft
1	Lahm	Philpp	1	FC Bayern Profis Herren
2	Badstuber	Holger	1	FC Bayern Profis Herren
3	Ribéry	Franck	1	FC Bayern Profis Herren
4	Martínez	Javier	1	FC Bayern Profis Herren



Quelle: CC BY 3.0, Guido Radig, https://commons.wikimedia.org/wiki/File:Maximilianeum_-_Frontseite_-_Panorama.jpg#/media/File:Maximilianeum_-_Frontseite_-_Panorama.jpg