**KIapitel 5**

**Struktur eines Javaprogramms:**

* Im Zusammenhang mit der objektorientierten Programmierung haben Begriffe wie Klasse*, Objekt, Attribut, Methode, Vererbung* und *Interface* eine besondere Bedeutung.

**Klasse:**

* Klassen definieren neue Typen, die man als Programmierer komplett auf seine Bedürfnisse zuschneiden kann
* Sie sind leistungsfähiger als primitive Datentypen
* Können nicht nur einen, sondern eine Vielzahl von Werten speichern
* Können auf Botschaften reagieren und selbst aktiv werden

Beispiel:

Class Bruch {

int zaehler;

int nenner;

}

**Attribute:**

* Sind fester Bestandteil einer Klasse und werden innerhalb der Klassendefintion festgelegt
* Müssen innerhalb einer Klasse eindeutig benannt sein
* Attribute eines Objekts werden über den Objektnamen angesprochen
  + Auf Objektname folgt, durch ein Punkt getrennt, der Name des Attributs

Beispiel:

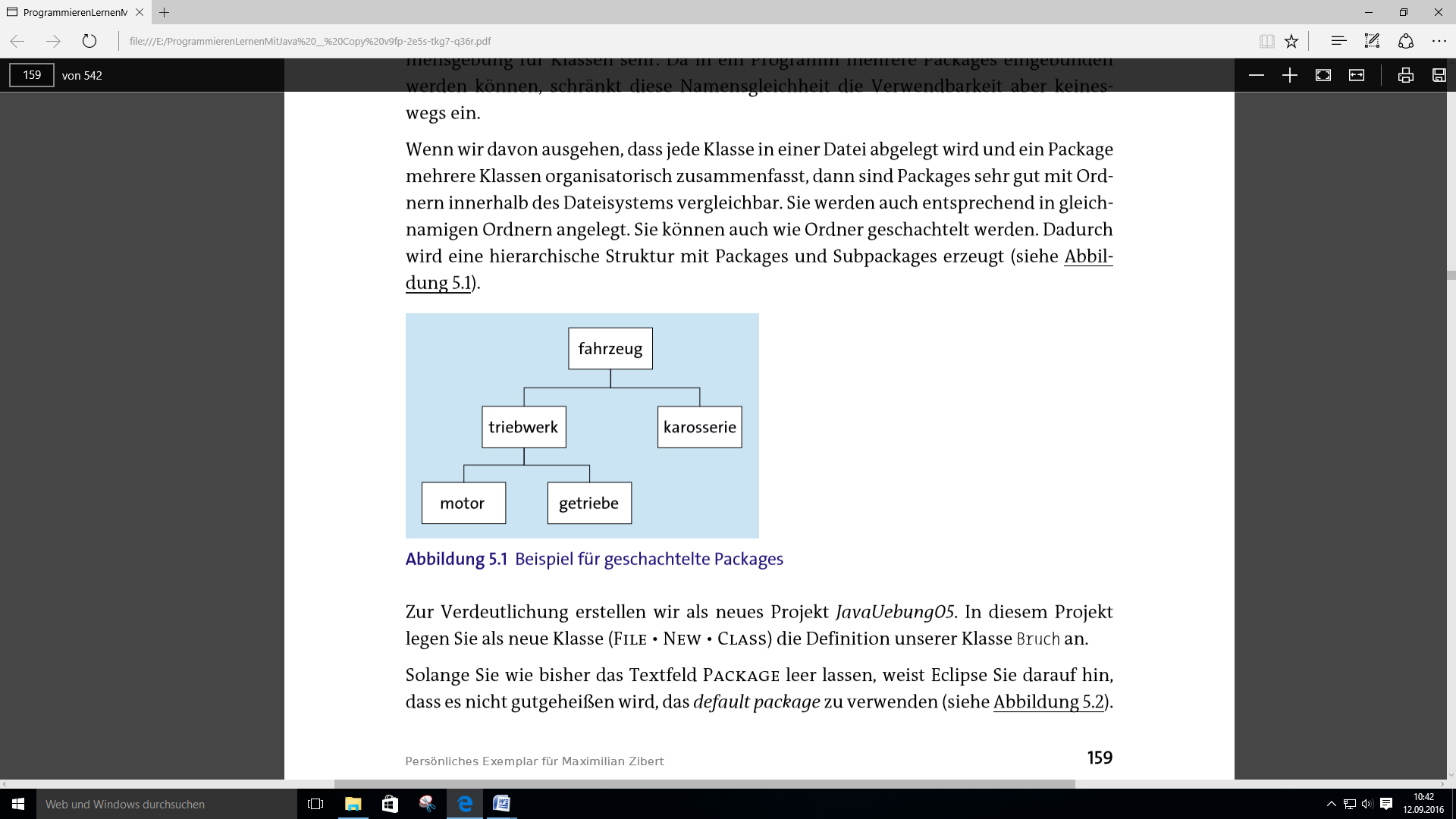
* Inkrementierung des Zählers 🡪 b.zehler++;
* Prüfen, ob der Nenner ungleich null ist 🡪 if (b.nenner != 0)

An dem Punkt zwischen Objektbezeichner und Datenelementbezeichner kann man erkennen, dass hier mit dem Attribut eines Objekts und nicht mit einer lokalen Variablen gearbeitet wird.

**Packages:**

* Packages sind sehr gut vergleichbar mit Ordnern innerhalb eines Datensystems
* Werden in gleichnamigen Ordnern angelegt
* Können auch wie Ordner geschachtelt werden
  + Hierarchische Struktur mit Packages und Subpackages

🡪Packages dienen dazu, mehrere logisch zusammengehörige Klassen zusammenzufassen und damit die Verwaltung größerer Programme zu vereinfachen.



Beispiel:

package uebung05;

class Bruch {

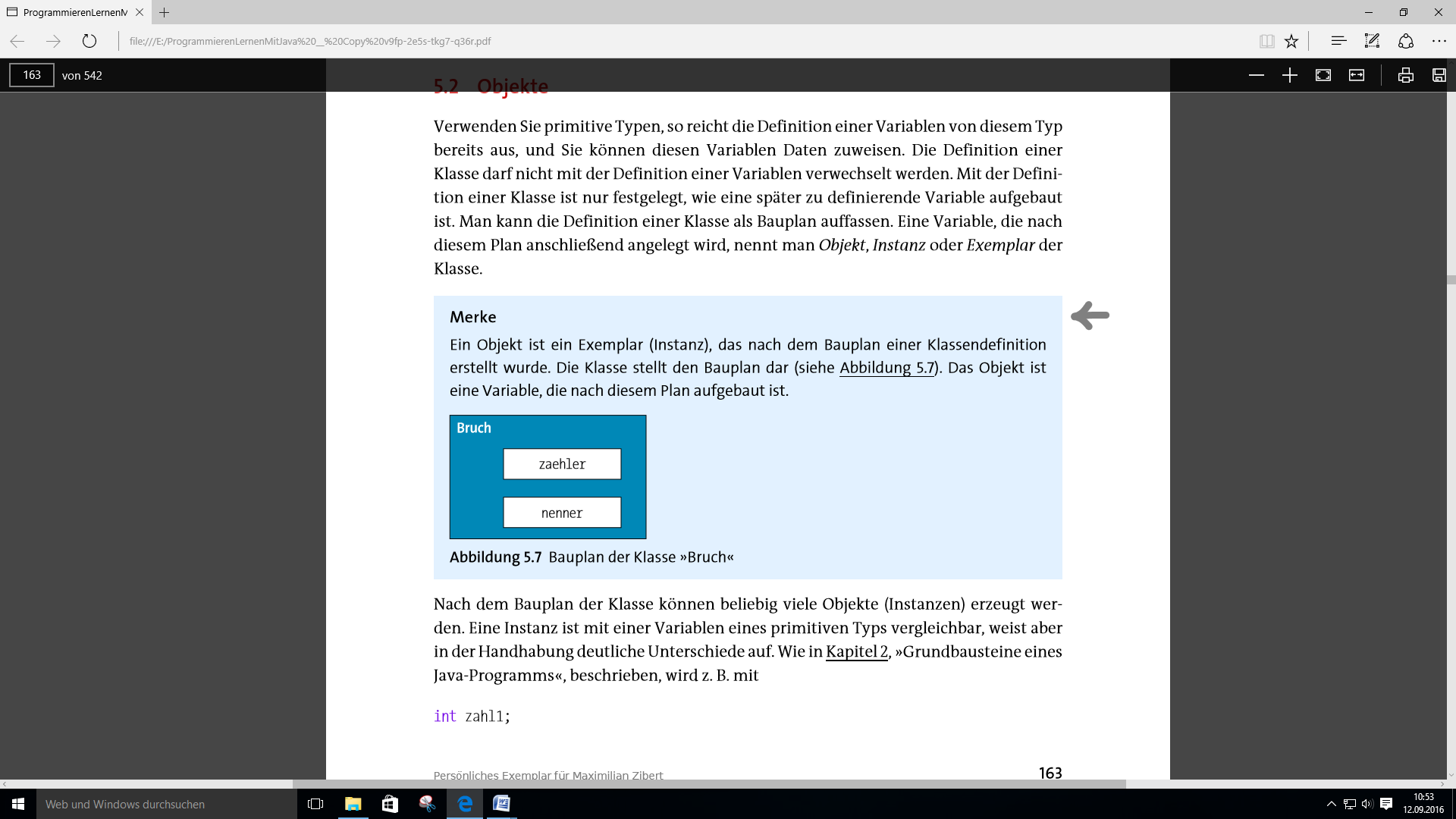
int zaehler;

int nenner;

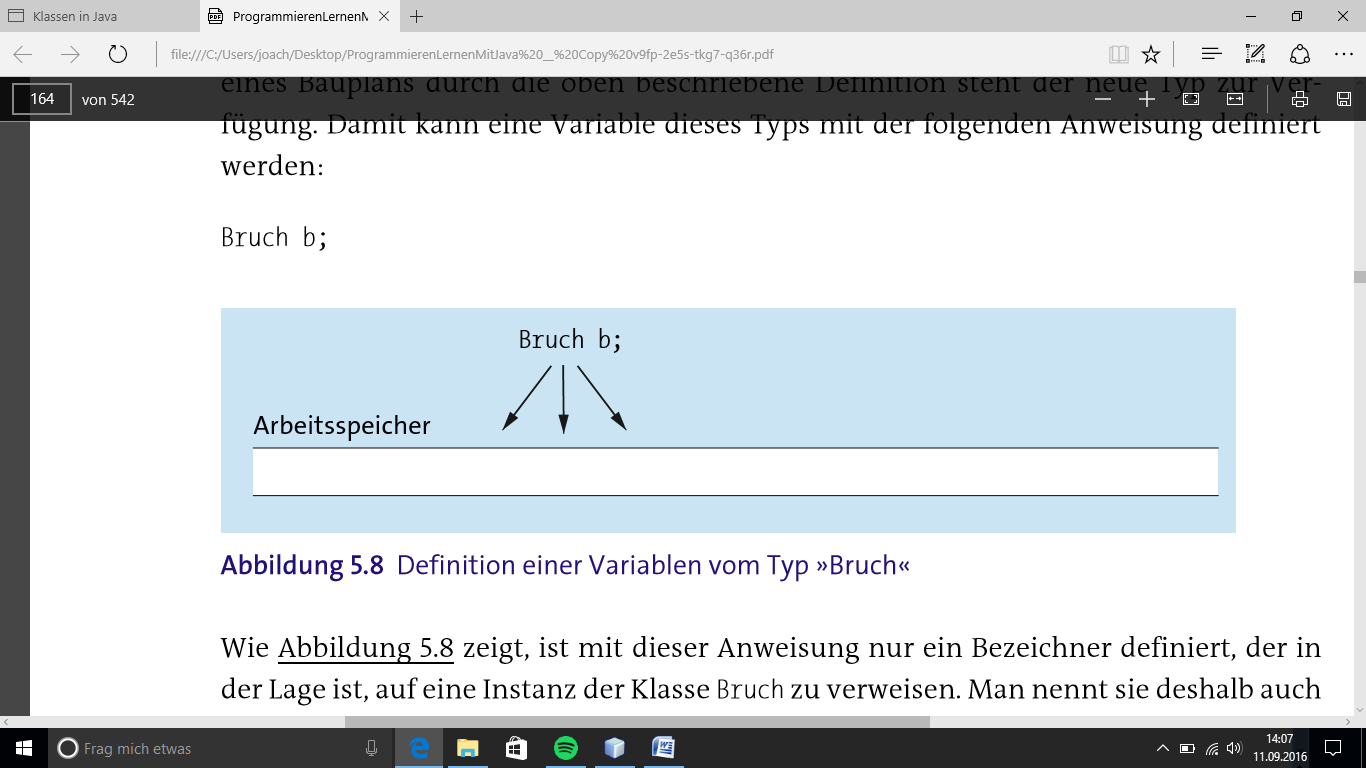
}

**Objekte:**

* Ein Objekt ist ein Exemplar, das nach dem Bauplan einer Klassendefinition erstellt wurde. Die Klasse stellt den Bauplan dar. Das Objekt ist eine Variable, die nach diesem Plan aufgebaut ist.



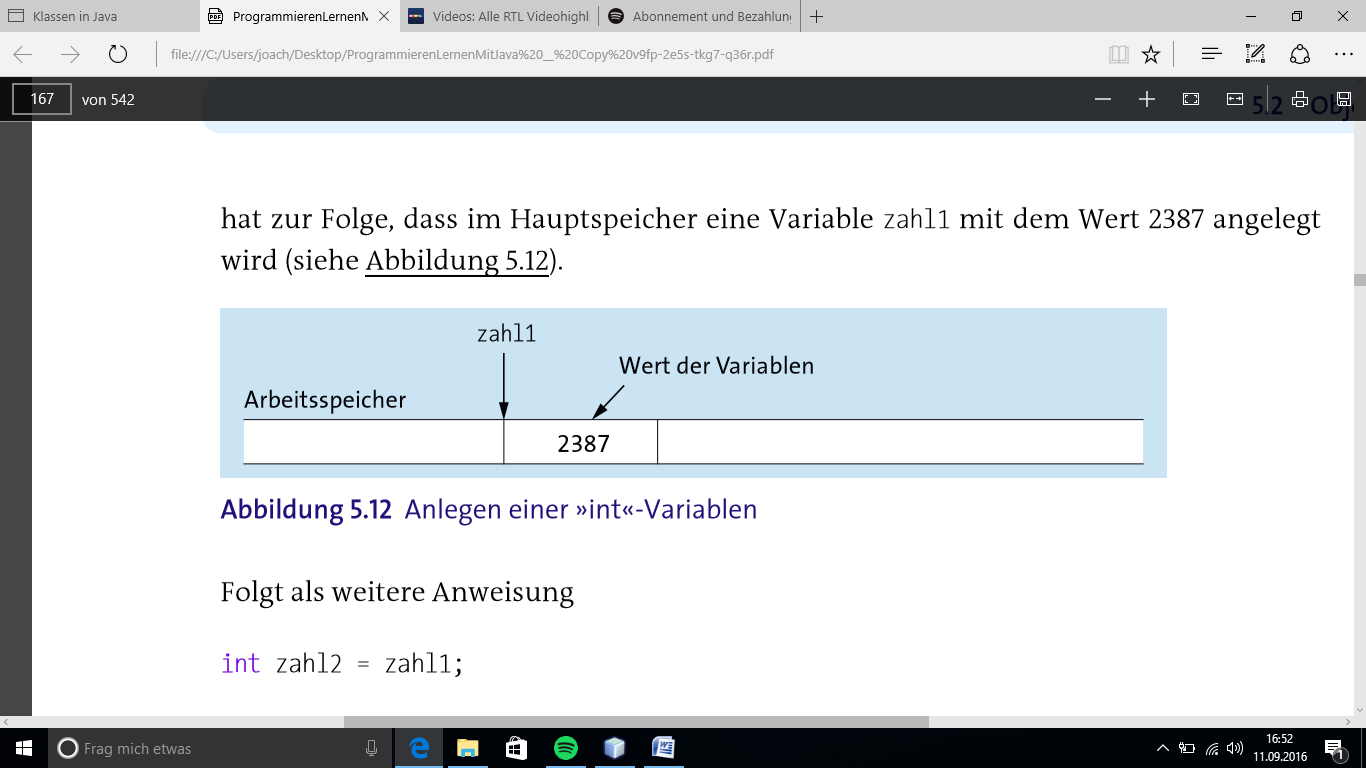
* Nach einer Erstellung des Bauplans durch die oben beschriebene Definition steht der neue Typ zur Verfügung
* Eine Variable dieses Typs kann mit folgender Anweisung definiert werden.



**Wertzuweisungen bei Objekten:**

* Objekte sind Referenztypen.
* Eine Wertzuweisung hat andere Folgen als bei den primitiven Typen

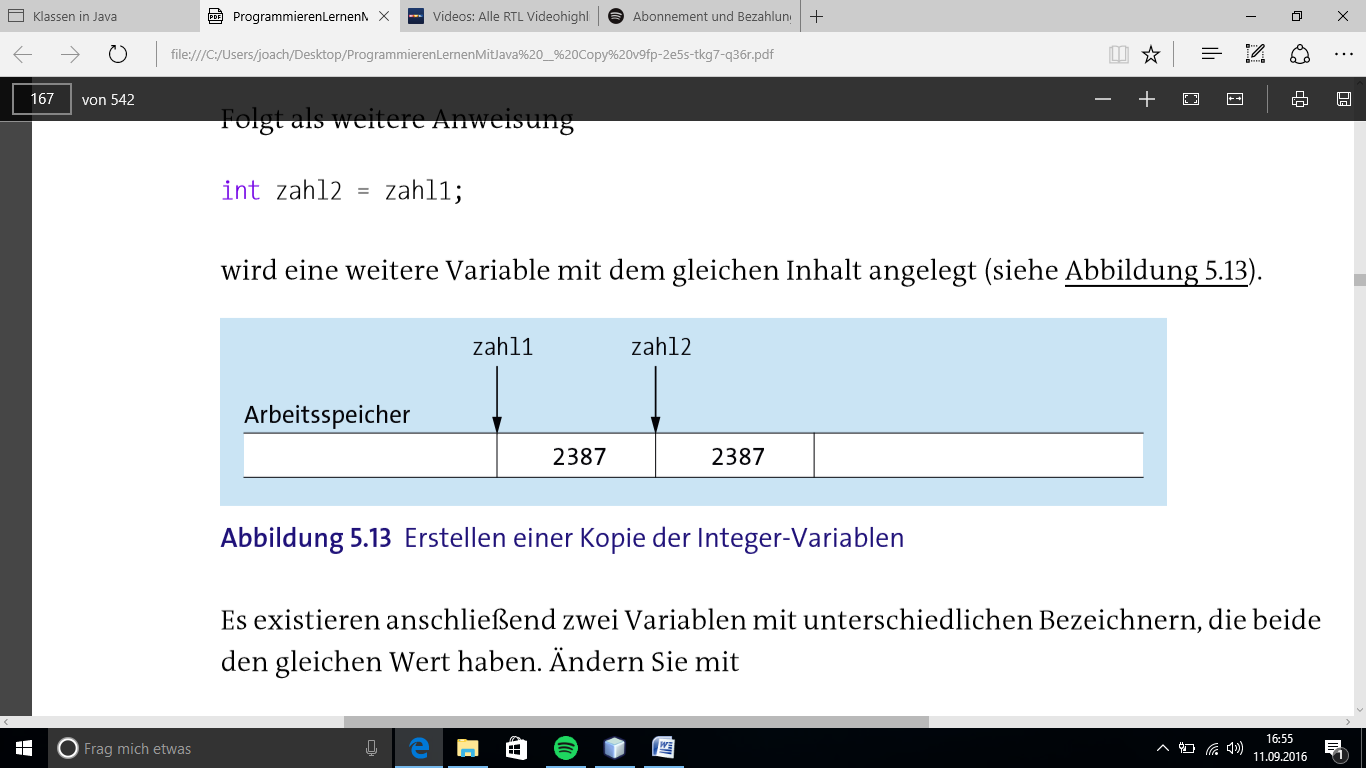
int zahl1 = 2387 🡪 im Hauptspeicher eine Variable zahl 1 mit dem Wert 2387



Weitere Anweisung

int zahl2 = zahl1;

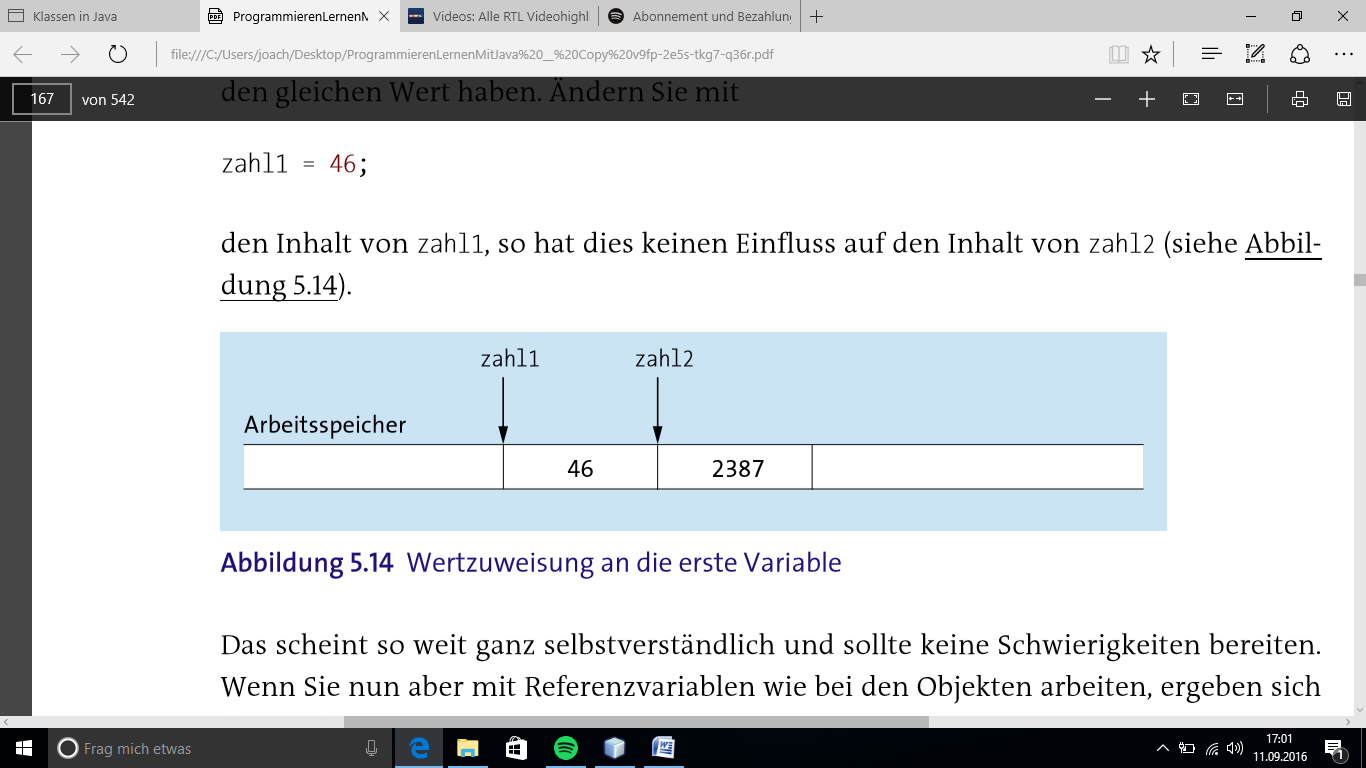
weitere Variable mit dem gleichen Inhalt:



🡪Es existieren anschließend zwei variablen mit unterschiedlichen Bezeichnern, die beide den gleichen Wert haben.

zahl1 = 46

Inhalt von zahl1 hat keinen Einfluss auf zahl2:



**Gültigkeitsbreich und Lebensdauer:**

* Unterschiede zwischen lokalen Datenträgern und Datenelementen eines Objekts
* Lokale Variablen: nur innerhalb eines Blocks, in dem sie definiert wurden
* Attribute eines Objekts: unabhängig von der Stelle, innerhalb der gesamten Klasse gültig

