

Lernziele

- Entwicklung eines Gespürs für die verschiedenen Datentypen
- Definition und Verwendung von Aufzählungstypen

Aufgabe 1) Datentypen

Öffnen Sie die Datei "Literals.java". Diese enthält eine Testklasse mit einer Methode "main" die weitere Methoden aufruft, die wir nach und nach aktivieren werden:

- `exploreRanges()`
- `exploreCompat()`
- `explorePitfalls()`
- `exploreArrays()`

Die Aktivierung der einzelnen Methoden erfolgt durch Einkommentieren, d.h. entfernen der Kommentarzeichen vor dem Aufruf.

```
functionThatWillBeCalled(); // einkommentiert
// functionThatWillNotBeCalled(); // auskommentiert
```

Im Beispielprogramm sieht das dann so aus:

```
// Main Method
public static void main(String[] args) {
    exploreRanges(); // einkommentiert
    // exploreCompat(); // auskommentiert
    // explorePitfalls(); // auskommentiert
    // exploreArrays(); // auskommentiert
}
```

Einkommentieren bzw. Auskommentieren (das temporäre Einfügen von Kommentaren) sind ein häufig genutztes Mittel zur kurzzeitigen Aktivierung/Deaktivierung von Code, z.B. zu Testzwecken oder zur Fehlersuche.

Anhand obiger Methoden sollen nun verschiedene Eigenschaften von Datentypen experimentell untersucht werden (Eigenschaften von Variablen wie Sichtbarkeit oder Lebensdauer kommen später).

- Wie war doch gleich der Zusammenhang zwischen Variable und Datentypen?
-

Teilaufgabe a) Wertebereiche

Überzeugen Sie sich zunächst durch Übersetzen und Ausführen davon, daß die Grundversion der Klasse funktionsfähig ist. Anschließend bearbeiten Sie die Methode `exploreRanges()`. In dieser Methode werden Variablen mit verschiedenen Datentypen definiert und anschließend initialisiert und über `System.out.println` (vgl. Aufgabe 2) ausgegeben.

Ändern Sie nun die Initialisierungen (Wertzuweisungen) für jede der Variablen indem Sie verschiedene Werte (Literale) ausprobieren und übersetzen Sie die Klasse erneut um den gültigen Wertebereich (vgl. Buch Seite xx) zu überprüfen. Vermerken Sie den Wertebereich in den dafür vorgesehenen Kommentaren hinter der jeweiligen Initialisierung, indem Sie die Platzhalter (???) durch die korrekten Werte ersetzen.

- Wie ist ihr Eindruck von den Wertebereichen der numerischen Datentypen?

- Was passiert, wenn Sie einen ungültigen Wert eingeben?

Teilaufgabe b)

Gehen Sie nun weiter zu der Methode `exploreCompat()`. Hier sollen am Beispiel des Additionsoperators (+) die Verträglichkeit der verschiedenen Datentypen untereinander untersucht werden. Fügen Sie dazu einige Zeilen der Form

```
System.out.println(<varA> + <varB>);
```

ein und variieren Sie dabei die Variablen A und B sowie ggf. deren Initialisierungswerte. Vermerken Sie die Verträglichkeit der Variablen in der folgenden Tabelle (es sind nicht alle Kombinationen nötig): Vermerken Sie geht bzw. geht nicht mit J/N und geben Sie ggf. den Ergebnistyp (ungefähr, d.h. Wahrheitswert, Zeichen, Ganzzahl, Fließkommazahl) an.

| | Bool | Char | String | Byte | Int | Long | Float | Double |
|--------|------|------|--------|------|-----|------|-------|--------|
| Bool | | | | | | | | |
| Char | | | | | | | | |
| String | | | | | | | | |
| Byte | | | | | | | | |
| Int | | | | | | | | |
| Long | | | | | | | | |
| Float | | | | | | | | |
| Double | | | | | | | | |

- Wie ist es mit der Verträglichkeit (Sind Typfamilien o.ä. erkennbar) ?

- Spielt die Reihenfolge der Verknüpfung dabei eine Rolle oder nicht ?

- Ist Ihnen bei der Deklaration bzw. der Initialisierung der Variablen etwas aufgefallen?

- Ist Ihnen bei der Schreibweise der Initialisierungswerte etwas aufgefallen?

Nur für Fortgeschrittene:

Schauen Sie sich nochmals die Ausgabebefehle in der Methode `exploreRanges()` an. Was wird hier verknüpft? Wundert Sie, daß das funktioniert? (Auflösung folgt später)

Teilaufgabe c)

Offensichtlich sind die numerischen Typen weitgehend verträglich zueinander. Das funktioniert so, daß "kleinere Typen" (mit weniger Genauigkeit) bei Verknüpfung mit "größeren Typen" (mit höherer Genauigkeit) automatisch "befördert", d.h. In den entsprechenden größeren Typ konvertiert werden. Man spricht hier von Promotion bzw. sog. impliziten Typumwandlungen (typecasts). Zur besseren Lesbarkeit ist auch eine explizite Typumwandlung möglich (vgl. Buch Seite xx).

Das Ganze hat aber auch seine Tücken wie man an Methode `explorePitfalls()` erkennen kann. Betrachten Sie einmal in Ruhe die Implementierung dieser Methode und schauen Sie sich anschließend die Ergebnisse bei Ausführung des Programms an.

Was müsste rauskommen? Was kommt wirklich raus? Haben Sie einen Verdacht oder können Sie es sich gar erklären? Vielleicht hilft ja das Buch weiter... Meinen Sie das Ihnen das in Zukunft jemals passiert? Verlassen Sie sich drauf: Es wird (wollen wir wetten?).

Teilaufgabe d)

Zum Abschluß noch ein kurzer Blick auf die Methode `exploreArrays()` und, nunja, Arrays eben. In der Methode wird ein Array definiert, initialisiert und anschließend darauf

zugegriffen.

Was passiert, wenn auf einen ungültigen Index zugegriffen wird? Vergleichen Sie dieses Verhalten mit dem bei Verwendung eines unzulässigen Werts in der Methode `exploreRanges()`. Man spricht in diesem Zusammenhang von Übersetzungsfehlern (Compile Errors) und Laufzeitfehlern (Runtime Errors). Was ist wohl was? Welche Sorte von Fehlern ist Ihnen lieber und warum?

Erinnern Sie sich an die Eigenschaft `length` eines Arrays? Handelt es sich hier um eine Instanzvariable, Instanzmethode, Klassenvariable oder Klassenmethode? Warum?

Aufgabe 2) Definition und Verwendung von Aufzählungstypen

Teilaufgabe a) Klassische Methode (einfach)

Erzeugen Sie eine neue Datei mit dem Namen `Ampel1.java` und folgendem Inhalt

```
public class Ampel1 {  
  
    public static void main( String[] args) {  
        int farbe;  
        farbe = 1; // rot  
        System.out.println( "farbe=" + farbe);  
        farbe = 2; // gelb  
        System.out.println( "farbe=" + farbe);  
        farbe = 3; // grün  
        System.out.println( "farbe=" + farbe);  
        farbe = 4; // ???  
        System.out.println( "farbe=" + farbe);  
    }  
}
```

Für wie lesbar und typsicher halten Sie diese Implementierung?

Teilaufgabe b) Klassische Methode (verbessert)

Erzeugen Sie eine neue Datei mit dem Namen `Ampel2.java` und folgendem Inhalt

```
public class Ampel2 {  
  
    public static void main( String[] args) {  
        int RED = 1;  
        int YELLOW = 2;  
        int GREEN = 3;  
        int farbe;  
        farbe = RED;  
        System.out.println( "farbe=" + farbe);  
        farbe = YELLOW;  
        System.out.println( "farbe=" + farbe);  
        farbe = GREEN;  
        System.out.println( "farbe=" + farbe);  
        farbe = 4; // ???  
        System.out.println( "farbe=" + farbe);  
    }  
}
```

Worin besteht die wesentliche Verbesserung? Was ist noch nicht optimal?

Teilaufgabe c) Verwendung von Aufzählungstypen

Erzeugen Sie eine neue Datei mit dem Namen Ampel3.java und folgendem Inhalt

```
public enum Ampel3 {  
  
    RED, YELLOW, GREEN;  
  
    public static void main( String[] args) {  
        Ampel3 farbe;  
        farbe = Ampel3.RED;  
        System.out.println( "farbe=" + farbe);  
        farbe = Ampel3.YELLOW;  
        System.out.println( "farbe=" + farbe);  
        farbe = Ampel3.GREEN;  
        System.out.println( "farbe=" + farbe);  
        farbe = 4; // ???  
        System.out.println( "farbe=" + farbe);  
    }  
}
```

Was hat sich jetzt geändert? Was halten Sie davon?

Aufgabe 3) Zusatzaufgabe

Versuchen Sie in der Methode `exploreCompat()` durch Verknüpfung entsprechend großer Werte und Zuweisung an eine Variable eine Wertebereichsüberschreitung zu provozieren. Führen Sie dazu ggf. weitere Variablen ein und passen Sie die Initialisierungswerte entsprechend an.

Gibt es eine Fehlermeldung und wenn ja, welcher Art? Wie war das vorher in der Methode `exploreRanges()`? Haben Sie dafür eine Erklärung?

Bemerkungen:

Die Übungen sind ausdrücklich dazu gedacht, Sie zum nachdenken und experimentieren (zu Deutsch: rumspielen) anzuregen. Tun Sie's! Die Elektronen gehen nicht kaputt, wenn Sie mal einen Fehler machen. Die Aufgaben bieten genug Raum für Experimente, und Ihnen fällt sicher noch mehr ein.

Reden und diskutieren Sie miteinander, und lassen Sie ihren Partner auch mal "fahren". Der Beifahrer ist nicht stumm sondern denkt laut mit und kommentiert den "Fahrstil" des anderen (fast wie im richtigen Leben...)