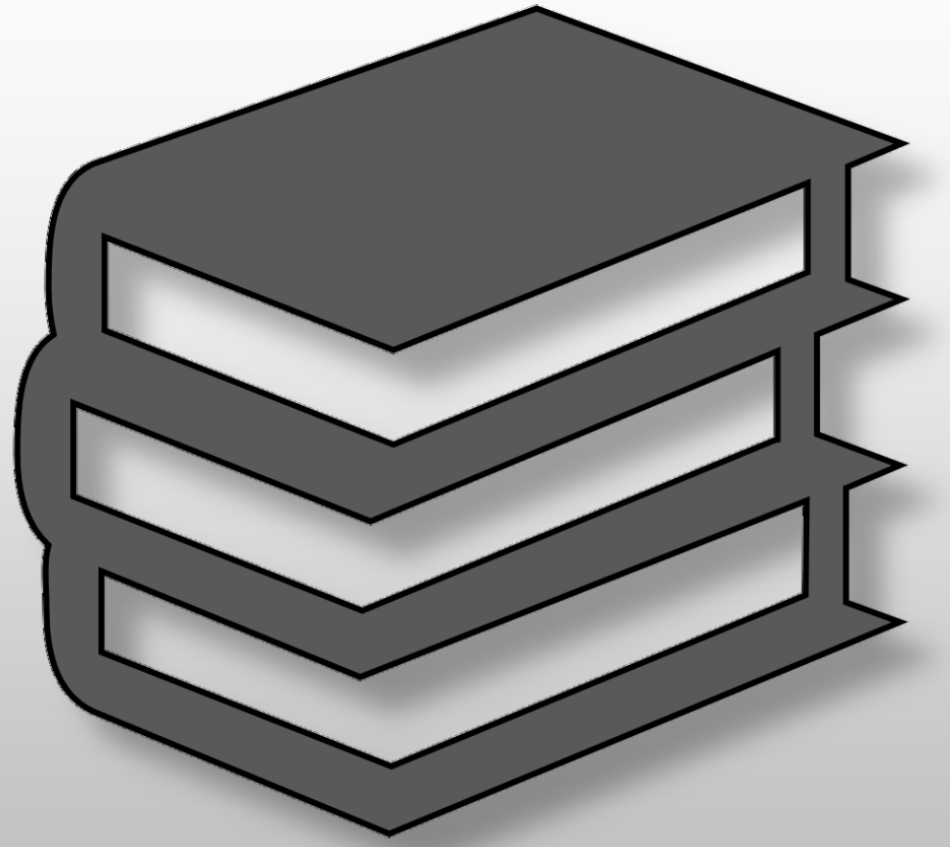
LEARN NUGGETS

WUNSCHTHEMEN SEITENS DER TEILNEHMER



INHALT

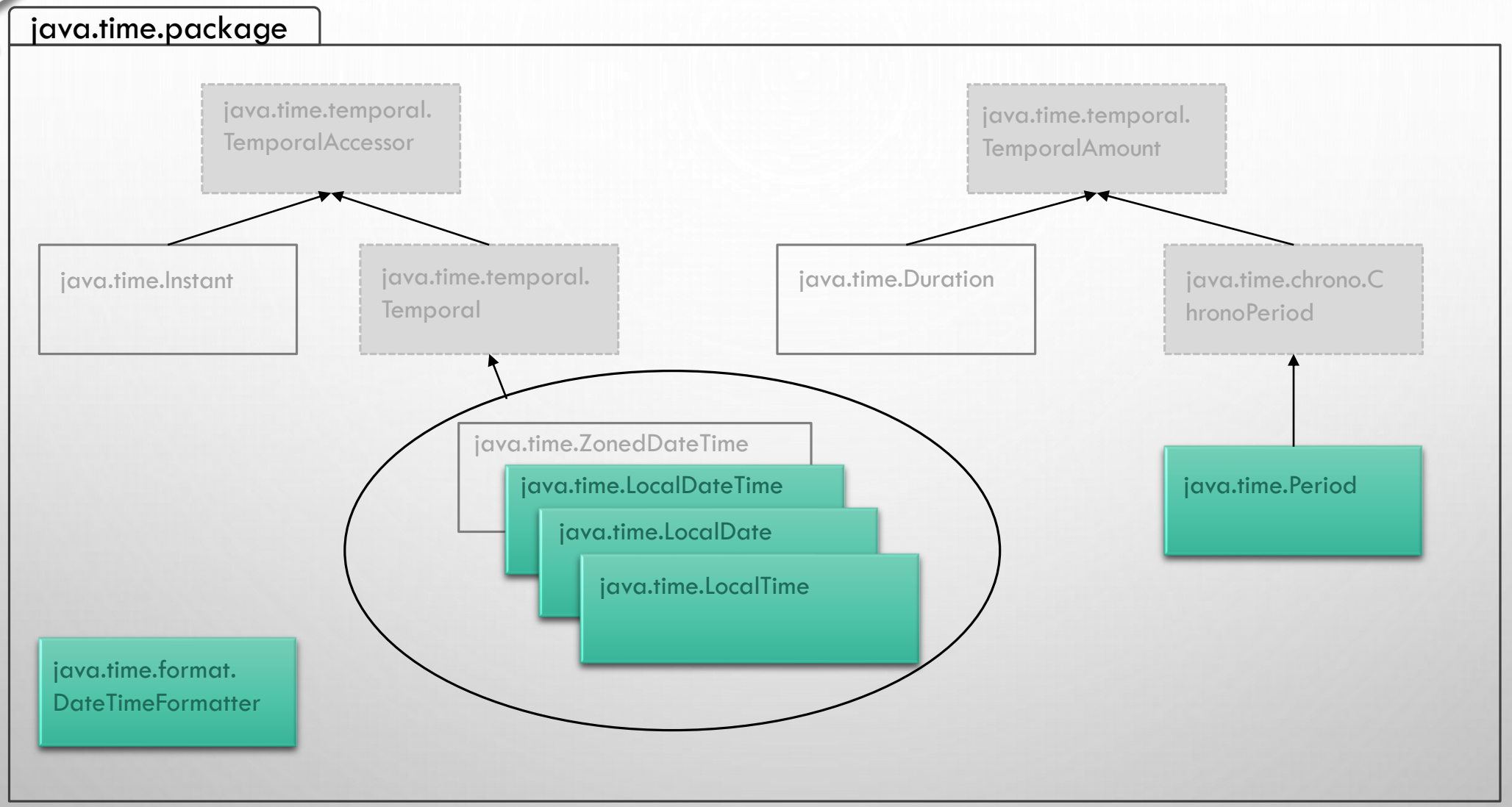
- Variable Scope

VARIABLE SCOPE

```
package oca.wiederholung;
import java.util.Scanner;

public class Zoo {
	static int anzahlTiere;
	int anzahl;
	String tier;
	Zoo(String tier){
		tier = tier;
		this.tier = tier;
		anzahlTiere = 0;
	}
	int gehege(Zoo tier, int anzahlTiere) {
		anzahl = anzahlTiere;
		anzahlTiere = 0;
		return anzahlTiere + Zoo.anzahlTiere;
	}
	public static void main(String[] args) {
		Zoo tier = new Zoo("Zebra");
		System.out.println("Wie viele " + tier.tier + " gibt es ?");
		int anzahlTiere = new Scanner(System.in).nextInt();
		while (anzahlTiere > 0) {
			if (anzahlTiere > 0 & anzahlTiere < 15) {
				System.out.println("Es gibt insgesamt: " + tier.gehege(tier,anzahlTiere)+ " Tiere im   Zoo");
				anzahlTiere = 0;
			}
			else break;
		}
	}
}
```

ARBEITEN MIT DATUM UND ZEITKLASSEN



Quelle: Associate Java 8 Programmer Certification Fundamentals OCAJP 1ZO-808; Hanumant Deshmukh;2019 Auflage 16

ARBEITEN MIT DATUM UND ZEITKLASSEN - NUTZEN

Erstellen neuer Objekte mit 3 möglichen statischen Methoden der Klassen

of Methode: Nutzt statische Methoden der jeweiligen Klasse die Überladen sind und dann ein entsprechendes Objekt erzeugen. Hier werden die Parameter für die einzelnen Werte individuell übergeben.

from Methode: Nutzt ebenfalls statische Methoden der jeweiligen Klasse, allerdings werden hier Objekte des Typs TemporalAccesor übergeben. Die Methode holt sich die für sie relevanten Werte aus dem übergebenen Objekt heraus.

now Methode Bildet ein Objekt der Klasse und nutzt dafür die aktuelle Systemzeit. Ausser der Klasse Period , da sie keine Zeiteinheiten stellt, welche die Uhrzeit darstellen.

ARBEITEN MIT DATUM UND ZEITKLASSEN – ERSTELLEN VON INSTANZEN

of Methode:

static LocalDate **of** (int year, int month, int dayOfMonth)
static LocalDate **of** (int year, Month month, int dayOfMonth)

static LocalTime **of** (int hour, int minute)
static LocalTime **of** (int hour, int minute, int second)
static LocalTime **of** (int hour, int minute, int second, int nanoOfSecond)

static Period **of** (int years, int months, int days)
static Period **ofDays** (int days)
static Period **ofMonths** (int months)
static Period **ofWeeks** (int weeks)
static Period **ofYears** (int years)

static LocalDateTime **of**(int year, int month, int dayOfMonth, int hour, int minute)
static LocalDateTime **of**(int year, int month, int dayOfMonth, int hour, int minute, int second)
static LocalDateTime **of**(int year, int month, int dayOfMonth, int hour, int minute, int second, int nanoOfSecond)
static LocalDateTime **of**(int year, Month month, int dayOfMonth, int hour, int minute)
static LocalDateTime **of**(int year, Month month, int dayOfMonth, int hour, int minute, int second)
static LocalDateTime **of**(int year, Month month, int dayOfMonth, int hour, int minute, int second, int nanoOfSecond)
static LocalDateTime **of**(LocalDate date, LocalTime time)

Achtung

1. Index für Monate startet mit 1 oder dem enum Wert Month.MONATSNAME
2. Der hour Parameter nutzt die 24 Stunden Zeiteinstellung (für alle mit dieser Zeit kein Problem)
3. Bei fehlerhaften Parameterwerten für Tage oder Stunden wird eine DateTimeException geworfen z.B. der day wert für den Monat darf die übliche Anzahl an Tagen übergeben also nicht 30 wenn der Monat nur 29 Tage hat oder für hour 24 , da der Bereich nur von 0 – 23 existiert

ARBEITEN MIT DATUM UND ZEITKLASSEN – ERSTELLEN VON INSTANZEN

from Methode:

```
LocalDateTime ldt = LocalDateTime.now();

LocalDate ld1 = LocalDate.from(ldt);
LocalDate ld2 = LocalDate.from(ld1);
LocalTime lt = LocalTime.from(ldt);
```

now Methode

```
LocalDate.now();
LocalTime.now();
LocalDateTime.now();
Period zero = Period.ZERO;
```

Period.now() // EXISTIERT NICHT !!!!!!

Die statische Konstante ZERO gibt eine periode von 0 zurück

```
LocalDate ld = LocalDate.now();                 // Hat nur Datumswerte !!!!!!
LocalDateTime ldt = LocalDateTime.from(ld);     // DateTimeException at run time
LocalTime lt = LocalTime.from(ld);              // DateTimeException at run time
```

Die plus und minus Methoden

LocalDate, ***LocalDateTime*** und Period Klassen haben :

- plusDays()
- plusWeeks()
- plusMonth()
- plusYears()

- minusDays()
- minusWeeks()
- minusMonths()
- minusYears()

LocalTime und ***LocalDateTime*** haben :

- plusNanos()
- plusSeconds()
- plusMinutes()
- plusHours()

- minusNanos()
- minusSeconds()
- minusMinutes()
- minusHours()