Teil 3: Programmieren von Klassen in Java

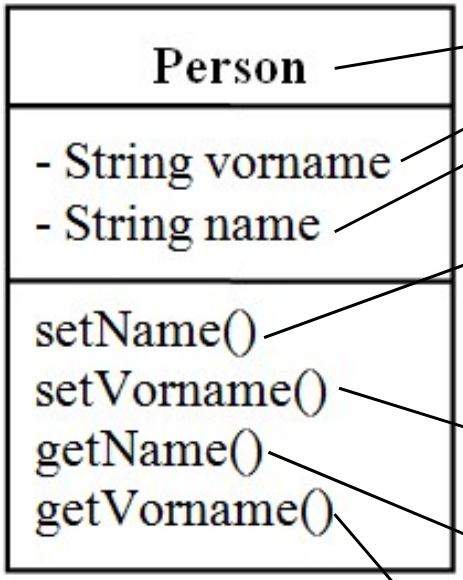
Übungsaufgaben

Modul „Grundlagen der objektorientierten Programmierung mit Java“

Prof. Dr. Cornelia Heinisch

Eine Klasse Person programmieren

UML-Notation:

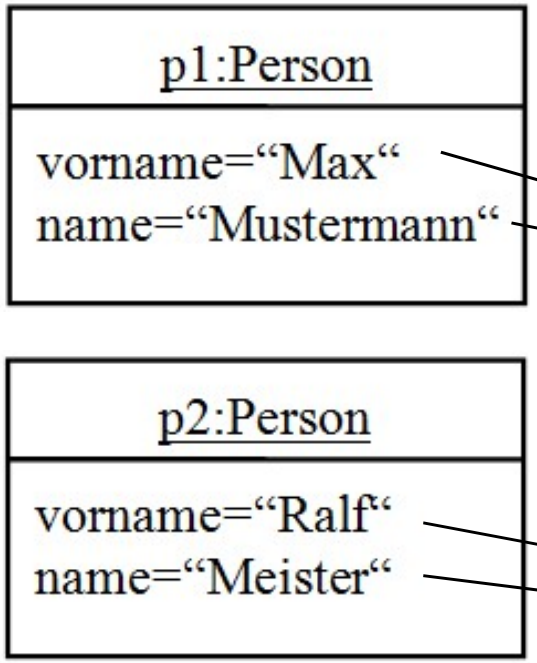



```
// Datei: Person.java

public class Person // Klassendeklaration der Klasse Person
{
   private String vorname;  // privates Datenfeld vorname
   private String name;  // privates Datenfeld name

   // Methode, um den Namen zu setzen.
   public void setName (String n)
   {
      name = n;
   }

   // Methode, um den Vornamen zu setzen.
   . . . . . // Versuchen Sie es selbst.

   // Methode, um den Namen abzufragen.
   public String getName()
   {
      return name;
   }

   // Methode, um den Vornamen abzufragen.
   . . . . . // Versuchen Sie es selbst.
}
```

Teil 3 Übungsaufgaben

Personen-Objekte erzeugen und die Klasse Person testen

UML-Notation:




```
// Datei: TestPerson.java

public class TestPerson
{
   public static void main (String[] args)
   {
      // Objekt der Klasse Person anlegen.
      Person p1 = new Person();

      // Datenfelder mit Werten belegen.
      p1.setVorname ("Max");
      p1.setName ("Mustermann");

      // Nochmal ein Objekt der Klasse Person anlegen.
      Person p2;
      p2 = new Person();

      // Datenfelder mit Werten belegen.
      . . . . . // Versuchen Sie es selbst.
      . . . . . // Versuchen Sie es selbst

      // Namen und Vornamen der Personen ausgeben.
      System.out.println(p1.getName()+" "+ p1.getVorname());
      System.out.println(p2.getName()+" "+ p2.getVorname());
   }
}
```

Teil 3 Übungsaufgaben

Aufgabe 1

- Legen Sie eine neue Projektmappe `Teil_3` in Eclipse an.
- Fügen Sie zwei Klassen als Quellcode-Dateien hinzu: `Person` und `TestPerson`.
- Tippen Sie den Quellcode für die Klassen `Person` und `TestPerson` ein, bzw. vervollständigen Sie diesen.
- Bringen Sie das Programm zur Ausführung.
- Die Ausgabe des Programmes ist dann:

```
Mustermann Max
Meister Ralf
```

Erläuterungen: Kommentare, Klassendeklaration und -definition

Kommentare:

- Ein Text hinter einem Doppelschrägstrich ist ein Kommentar.
- Ein Kommentar dient der Dokumentation und hat keinen Einfluss auf die Ausführung des Programmes.

Klassendeklaration und Klassendefinition:

- Eine Klasse wird in Java durch das Schlüsselwort `class` deklariert.
- Hinter dem Schlüsselwort `class` steht der Klassenname.
- Üblicherweise wird die Klasse als `public` deklariert. Dies bedeutet, dass die Klasse für alle anderen Klassen sichtbar (und damit verwendbar) ist.
- Innerhalb der geschweiften Klammern `{ }` folgt die so genannte Klassendefinition mit den Datenfeldern und Methoden.
- Datenfelder werden mit dem Schlüsselwort `private` als privat gekennzeichnet. Auf private Datenfelder können nur die Methoden der eigenen Klasse zugreifen.
- Methoden werden üblicherweise mit dem Schlüsselwort `public` als öffentlich sichtbar gekennzeichnet. Methoden, die mit `public` gekennzeichnet sind, können von anderen Klassen aus aufgerufen werden.

Erläuterungen: Objekte erzeugen, Methoden und Datenfelder

Objekte erzeugen:

- Damit man die Methoden einer Klasse aufrufen kann, muss man zuerst ein Objekt erzeugen. Die Klasse ist ja nur der Bauplan (der Datentyp) für die Objekte (Variablen).
- Ein Objekt kann in Java nur durch den `new`-Operator erzeugt werden.

Datenfelder mit Werten initialisieren:

- Nach dem Anlegen eines Objektes können den einzelnen Datenfeldern Objekt-individuelle Werte zugewiesen werden.
- Ein Methodenaufruf erfolgt immer zu einem Objekt, da die Methoden die Objekt-individuellen Werte lesen oder verändern.

Der Datentyp String

- Die Datenfelder `name` und `vorname` sind vom Datentyp `String`. In einer Variablen vom Datentyp `String` (z. B. `name` oder `vorname`) kann eine Zeichenkette abgelegt werden. Zeichenketten vom Datentyp `String` können über den Operator `+` miteinander verkettet werden. Diese Verkettungsfunktion kommt bei der Ausgabe mit Hilfe von `System.out.println()` zum Einsatz.

Erläuterungen: Testklasse und Methode main()

Testklasse:

- Es wird als guter Programmierstiel angesehen, wenn man zum Test einer Klasse eine eigene Testklasse schreibt.
- Mit diesem Vorgehen ist der eigentliche Programmcode stets vom Programmcode für den Test separiert.
- Die Testklassen können dann bei einer späteren Auslieferung der Software einfach weggelassen werden und verbrauchen keinen unnötigen Speicherplatz.

Die Methode main():

- Jede Java-Anwendung benötigt eine Methode `main()`, da mit dem Aufruf der `main()`-Methode eine Java-Anwendung ihre Ausführung beginnt.
- Beim Aufruf des Java-Interpreters mit `java TestPerson` wird vom Interpreter in der Klasse `TestPerson` die Methode `main()` gesucht und ausgeführt. Fehlt die Methode `main()`, oder ist diese nicht `public`, oder nicht `static`, so bricht der Interpreter mit einer Fehlermeldung ab.
- Das Schlüsselwort `static` kennzeichnet die Methode `main()` als Klassenmethode. Dies bedeutet, dass die Methode `main()` – ohne die Existenz eines Objektes – direkt (durch den Interpreter) aufgerufen werden kann.