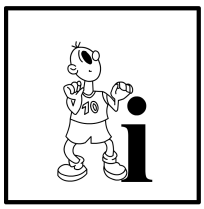


Der Java-Editor bietet sogenannte Hilfsfunktionen um effizienter programmieren zu können. Ab Heute werden wir einige dieser Funktionen kennenlernen.

Eine Klasse um Attribute erweitern

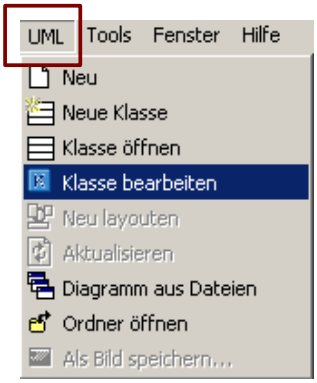
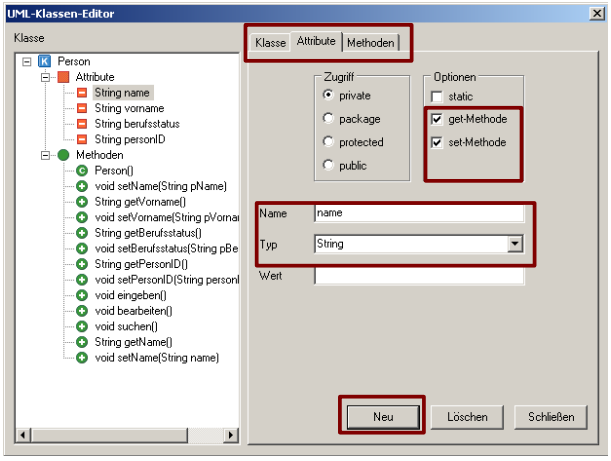


Abbildung 01:
UML-Klassen-Editor

Klicken Sie in der *Menü-Leiste* auf *UML >> Klasse bearbeiten*, um die aktuelle Klasse modifizieren zu können.

Sie öffnen damit den UML-Klassen-Editor.



Abblindung 02:
UML-Klassen-Editor

Der UML-Klassen-Editor bietet eine grafische Benutzeroberfläche, um Klassen, Attribute und Methoden hinzufügen/ändern zu können.

Vorteil: Der UML-Klassen-Editor erzeugt im Hintergrund den benötigten Java-Quellcode.

Attribute hinzufügen: Der Reiter *Attribute* sollte aktiv sein.

Klicken Sie auf *Neu* >> fügen Sie den *Namen* und den *Typ* des Attributs ein. Mit dem *Schließen* des Fensters wir der Quellcode im Hintergrund produziert.

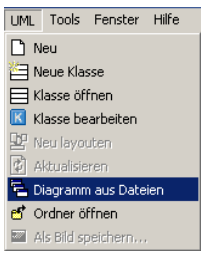
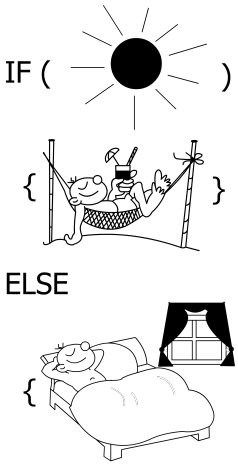


Abbildung 03:
UML-Diagramm erstellen

Exkurs: Nutzen Sie die Funktion UML >> Diagramm aus Dateien öffnen, um ein Klassendiagramm Ihres Systems zu erstellen. Hinweis: Alle relevanten Dateien müssen dazu geöffnet sein.



Kontrollstrukturen in Methoden

In der Aufgabe soll die Fachklasse `Person.java` um die folgende Methode erweitert werden:

```
//Variante: IF-Anweisung
public void identitaetPruefen(String pBenutzername, String pPasswort) {
   String mBn = this.getBenutzername();
   String mPs = this.getPasswort();
    if(mBn.equals(pBenutzername) && mPs.equals(pPasswort)){
     this.setErfolgreich(true);
     System.out.println("Anmeldung erfolgreich!!");
    }else{
     System.out.println("Anmeldung fehlgeschlagen!!");
    }
}
```

Parameter

Lokale Attribute (m: memory)

Bedingung (B1)

Bedingung (B2)

Der String-Vergleich mit der Methode „equals()“

`A1.equals(A2)`	Für den Vergleich zweier Werte mit komplexem Datentyp (u.a. String): *z.B. A1 = mBn, A2 = pBenutzername* Wert in *Attribut1* … ist gleich... Wert in *Attribut2*? | Rückgabewert ist *„true“* oder *„false“*

Operatoren im Fall eines nummerischen Wertevergleichs

Bedingungen enthalten in der Regel Vergleichsoperatoren. Sie liefern den Rückgabewert *„true“* (für wahr) oder *„false“* (für falsch).

„`A1 == A2`“	Für den Vergleich zweier numerischer Werte: *z.B. A1 = 5, A2 = 10* Wert in *Attribut1* … ist gleich... Wert in *Attribut2*? | Rückgabewert ist *„false“*
„`B1 && B2`“	Für die UND-Verkettung mehrerer Bedingungen: *Bedingung1* … und … *Bedingung2* … muss korrekt sein, dann wird ein *„true“* geliefert (IF-Zweig) ...sonst (ELSE)... wird ein *„false“* geliefert (ELSE-Zweig).
„`A1 < A2`“	Für den Vergleich zweier Werte: *z.B. A1 = 5, A2 = 10* Wert in *Attribut1* … ist größer als... Wert in *Attribut2*? | Rückgabewert ist *„true“*
„`A1 > A2`“	Für den Vergleich zweier Werte: *z.B. A1 = 3.50, A2 = 5.20* Wert in *Attribut1* … ist kleiner als... Wert in *Attribut2*? | Rückgabewert ist *„false“*
„`A1 >= A2`“	Für den Vergleich zweier Werte: *z.B. A1 = 10, A2 = 7 pBenutzername* Wert in *Attribut1* … ist größer oder gleich... Wert in *Attribut2*? | Rückgabewert ist *„true“*

„A1 <= A2“	Für den Vergleich zweier Werte: *z.B. A1 = mBn, A2 = pBenutzername* Wert in Attribut1 … ist kleiner oder gleich... Wert in Attribut2?

Exkurs: Methoden enthalten vielfach Rechenoperator, um Werte zu berechnen. In der Regel rechnet man mit Werten vom Typ Integer, double oder long.

„A1 + A2“	Für die Addition zweier Werte: Wert in Attribut1 … plus... Wert in Attribut2?
„A1 - A2“	Für die Subtraktion zweier Werte: Wert in Attribut1 … minus... Wert in Attribut2?
„A1 * A2“	Für die Multiplikation zweier Werte: Wert in Attribut1 … mal... Wert in Attribut2?
„A1 / A2“	Für die Division zweier Werte: Wert in Attribut1 … durch... Wert in Attribut2?

Ergänzung der Test-Methode

in der `StarterKlasse.java`

```
public static void main(String[] args){
    //Hier muss der Quellcode ergaenzt werden
}
```

```
//Start-Methode wird hier zum testen der Fachklasse genutzt
public static void main(String[] args) {
    Person person1 = new Person();
    //setzt die Attributwerte fuer das Objekt person1 im Produkt-Pflege-System
    person1.setName("Bellamy");
    person1.setVorname("Myra");
    person1.setBerufsstatus("Mitarbeiterin der Einkaufsabteilung");
    person1.setPersonID("MA101");

    //Benutzername und Passwort setzen
    person1.setBenutzername("mbellamy");
    person1.setPasswort("ps4MyAccount!");

    //Produziert die Testausgaben auf der Konsole
    System.out.println("#####################################################");
    System.out.println("###### DIE DATEN WURDEN ERFOLGREICH GESETZT!######");
    System.out.println("Name:"+person1.getName());
    System.out.println("Vorname:"+person1.getVorname());
    System.out.println("Berufsstatus:"+person1.getBerufsstatus());
    System.out.println("###### HERZLICHEN GLUECKWUNSCH!####################");
    System.out.println("#####################################################");

    /*Hier wird die Eingabe über die Benutzeroberflaeche simmuliert
     *Testfall1: bei korrekter Eingabe von Benutzername und Passwort*/
    person1.identiteatPruefen("mbellamy","ps4MyAccount!");

    /*Hier wird die Eingabe über die Benutzeroberflaeche simmuliert
     *Testfall2: bei fehlerhaften Eingabe von Benutzername und/oder Passwort*/
    person1.identiteatPruefen("mbelli","ps4MyAccount!");
}
```

Setzen des Benutzernamens und des Passworts für Myra Bellamy

Fall1: richtige Eingabe

Fall2: falsche Eingabe

Ergebnis des Tests