Programmierkurs Java

UE 17 - Polymorphie

Dr.-Ing. Dietrich Boles

Gliederung

- Definition
- Beispiel
- Protokolleinschränkung
- Typumwandlung
- Beispiel Stack
- Zusammenfassung

Definition

Polymorphie:

- Objektvariablen als auch Objekte haben einen Typ:

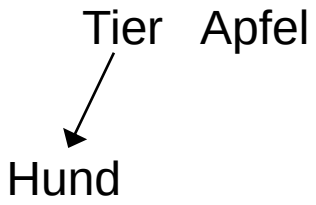
```
T1 objVar = new T2();
```

- Polymorphie ist die Fähigkeit einer Objektvariablen vom Typ T_1, auf Objekte von Klassen eines anderen Typs T_2 verweisen zu können
- in Java: T_2 muss Unterklasse von T_1 sein
- Protokoll des Objektes ist auf das Protokoll von T_1 eingeschränkt, d.h. über die Objektvariable lassen sich nur die Methoden von T_1 aufrufen
- Hintergrund: Protokoll von T_1 ist Teilmenge von Protokoll von T_2
- Relevanz: Compilierzeit

Beispiel

- Sei **Hund** eine direkte oder indirekte **Unterklasse** von **Tier** .

```
Tier t = new Hund(); // Polymorphie
```




```
static void streicheln(Tier obj) { ... }
Hund bello = new Hund();
streicheln(bello);  // Polymorphie


Hund max = new Tier();  // Fehler

Tier t = new Apfel();  // Fehler
```

Protokolleinschränkung

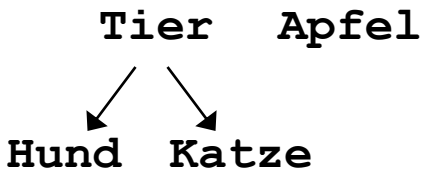
```
class Person {
  String name;
  Person(String name) { this.name = name; }
  void print() { IO.println(name); }
}
class Mitarbeiter extends Person {
  String buero;
  Mitarbeiter(String name, String buero) {
    super(name); this.buero = buero;
  }
  void changeBuero(String neuB) { buero = neuB; }
}
class PersonenTest {
  public static void main(String[] args) {
    Mitarbeiter kai = new Mitarbeiter("Kai Meyer", "E 67");
    Person karl = new Mitarbeiter("Karl Schmidt", "E 50");

    kai.changeBuero("E 69");  // ok!
    karl.changeBuero("E 68"); // Fehler (Protokolleinschr.)
} }
```

Typumwandlung

- Typumwandlung (Typecast) von Oberklasse auf Unterklasse explizit möglich (allerdings u.U. unsicher!)

```
class Tier { ... }
class Hund extends Tier { ... }
class Katze extends Tier { ... }
class Apfel { ... }
```




```
Tier t  = new Hund();
Hund h1 = t;          // Syntaxfehler!
Hund h2 = (Hund)t;    // ok
Katze k = (Katze)t;   // syntaktisch ok,
                      // jedoch Laufzeitfehler
Apfel a  = (Apfel)t; // Syntaxfehler
```

Beispiel Stack (1)

- Vorteil der Polymorphie: **Wiederverwendbarkeit**

```
class Stack {
    Object[] store;  // zum Speichern von Daten
    int current;     // aktueller Index

    Stack(int size) {
      store = new Object[size]; current = -1;
    }

    boolean isFull() { return current == (store.length-1); }
    boolean isEmpty() { return this.current == -1; }

    void push(Object value) {
      this.store[++this.current] = value;
    }
    Object pop() { return this.store[this.current--]; }
}
```

Erinnerung: `Object` ist (indirekte) Oberklasse jeder Klasse in Java

Beispiel Stack (2)

```
class Rueckwaerts {
  public static void main(String[] args) {

    Stack haufen = new Stack(10);

    while (!haufen.isFull()) {
      String eingabe = IO.readString("Eingabe:");
      haufen.push(eingabe);
    }

    while (!haufen.isEmpty()) {
      String str = (String)haufen.pop();
      System.out.println(str);
    }
} }
```

Beispiel Stack (3)

```
class RueckwaertsInt {
  public static void main(String[] args) {

    Stack haufen = new Stack(10);

    while (!haufen.isFull()) {
      int zahl = IO.readInt("Eingabe:");
      haufen.push(zahl);
    }

    while (!haufen.isEmpty()) {
      int zahl = (Integer)haufen.pop();
      System.out.println(zahl);
    }
  }
}
```

Beispiel Stack (4)

```
class Akte {
  int nummer;
  Akte(int n) { this.nummer = n; }
  void print() { IO.println(this.nummer); }
}

class Verwaltung {
  public static void main(String[] args) {
    Stack haufen = new Stack(8);
    while (!haufen.isFull()){
      haufen.push(new Akte(IO.readInt("Zahl: ")));
    }
    while (!haufen.isEmpty()) {
      Akte akte = (Akte)haufen.pop();
      akte.print();
    }
  }
}
```

Zusammenfassung

- Polymorphie: Fähigkeit einer Objektvariablen vom Typ T_1, auf Objekte von Klassen eines anderen Typs T_2 verweisen zu können, wobei in Java T_2 Unterklasse von T_1 sein muss
- Vorteil der Polymorphie: flexible Wiederverwendbarkeit von Klassen