CARL VON OSSIETZKY universität OLDENBURG

Teil

Objektorientierte Programmierung

Unterrichtseinheit 33

Abstrakte Klassen

Dr. Dietrich Boles

Gliederung

- Motivation
- Definition
- Beispiel Graphik
- Beispiel Prüfung
- Beispiel java.util.Calendar
- Zusammenfassung

Motivation

Prinzip der Vererbung:

- Aus existierenden Klassen können spezialisierte Klassen abgeleitet werden

Prinzip der abstrakten Klassen:

- Aus mehreren ähnlichen Klassen kann eine gemeinsame Oberklasse abstrahiert werden
- Sinn und Zweck: Teilvererbung und Ausnutzung der Polymorphie!

Schema:

```
class Rechteck  { void draw() {...} ...}
class Kreis     { void draw() {...} ...}
class Linie     { void draw() {...} ...}

"abstrakte" class Graphik {
  "abstrakte Methode" void draw();
}
```

Definition (1)

Definition: Abstrakte Klasse

- Eine abstrakte Klasse ist eine bewusst **unvollständige** Oberklasse, in der von einzelnen Methodenimplementierungen abstrahiert wird (**"abstrakte" Methoden**!)

```
<abstrakte Klasse> ::= „abstract“ „class“ <Kbezeichner> „{„
                       { <attribut> |
                         <methode> |
                         <konstruktor>
                         <abstrakte Methode>
                       } „}“
<abstrakte Methode> ::= „abstract“ <typ>
                        <Mbezeichner> „(„ <paramListe> „)“ „;“
```

Definition (2)

Beispiel:

```
abstract class Graphik {
    ...
    abstract void draw();
}
```

- Fehlende Methodenrümpfe werden erst in abgeleiteten Unterklassen implementiert.
- Die Instantiierung abstrakter Klassen ist nicht möglich, d.h. es lassen sich keine Objekte vom Typ <Kbezeichner> erzeugen
- Es lassen sich wohl aber Objektvariablen vom Typ <Kbezeichner> definieren; damit kann Polymorphie ausgenutzt werden!

Beispiel Graphik (1)

```
abstract class Graphik {
  String name;
  Graphik(String str) { this.name = str; }
  String getName() { return this.name; }
  abstract void draw();
}

class Rechteck extends Graphik {
  float width, height;
  Rechteck(String str, float w, float h) {
    super(str); this.width = w; this.height = h;
  }
  // geerbt: getName
  // weitere Methoden ...
  // implementiert:
  void draw() {
    IO.println("Rechteck:" + this.name);
} }
```

Beispiel Graphik (2)

```
class Kreis extends Graphik {
  float radius;
  Kreis(String str, float r) {
    super(str); this.radius = r;
  }
  // geerbt: getName
  // weitere Methoden ...
  // implementiert:
  void draw() {
    IO.println("Kreis:" + this.name);
} }
class Linie extends Graphik {
  ...
  // implementiert:
  void draw() {
    IO.println("Linie:" + this.name);
} }
```

Beispiel Graphik (3)

```
class Set {
  Graphik[] elems;
  int next;

  Set(int size) {
    this.elems = new Graphik[size];
    this.next = 0;
  }
  void add(Graphik obj) {
    if (this.next < this.elems.length)
      this.elems[this.next++] = obj;
  }
  Graphik[] get() {
    return this.elems;
} }
```

Beispiel Graphik (4)

```
class GraphikProbe {
  public static void main(String[] args) {
    Rechteck r1 = new Rechteck("Rechteck 1", 10, 20);
    Kreis k1 = new Kreis("Kreis 1", 50);
    Linie l1 = new Linie("Linie 1", 40);
    Rechteck r2 = new Rechteck("Rechteck 2", 15, 15);
    Graphik g1 = new Graphik("Graphik 1"); // Fehler!
    Set menge = new Set(6);
    menge.add(r1);    // Ausnutzung der Polymorphie!
    menge.add(r2);
    menge.add(k1);
    menge.add(l1);
    Graphik[] elements = menge.get();
    for (int i=0; i<elements.length; i++)
      if (elements[i] != null)
        elements[i].draw(); // Dynamisches Binden!
} } }
```

Beispiel Pruefung (1)

Es soll eine Prüfung durchgeführt werden.
Die Prüfung besteht aus einem Quiz, das eine Menge an Prüflingen zu absolvieren haben. Ein Quiz besteht aus einer Menge an Fragen. Bei den Fragen handelt es sich um Fragen unterschiedlichen Typs (Wahr/Falsch, MultipleChoice, ...).
Bei einer richtigen Antwort bekommt der Prüfling eine der Frage zugeordneten Punktzahl.

Demo

Eine Prüfung läuft so ab, dass zunächst das Quiz vorbereitet wird, d.h. es werden eine Menge an Fragen eingegeben.
Anschließend wird die Prüfung durchgeführt, d.h. die Prüflinge müssen der Reihe nach die Fragen beantworten.
Danach wird eine Rangliste nach den erreichten Punkten erzeugt und die Ergebnisse der Prüfung werden ausgegeben.

Beispiel Pruefung (2)

- Identifikation von Objekten/Klassen:
  - Fragen
  - Quiz
  - Prüfling
  - Prüfung

- Identifikation von Beziehungen zwischen Objekten:
  - Ein Quiz besteht aus mehreren Fragen
  - Eine Prüfung besteht aus einem Quiz, an dem mehrere Prüflinge teilnehmen

- Identifikation von Beziehungen zwischen Klassen:
  - Fragen (mit unterschiedlichen Ausprägungen):
    - WahrFalsch-Fragen
    - Multiple-Choice-Fragen
    - ...

Beispiel Pruefung (3)

- Identifikation von Eigenschaften und Funktionen von Objekten
  - Prüfung:
    - Quiz
    - Menge an Prüflingen
    - Vorbereiten
    - Durchführen
    - Ergebnisse bekannt geben
  - Prüfling:
    - Name
    - Erreichte Punkte
    - Punkte hinzufügen
  - Quiz:
    - Titel
    - Menge an Fragen
    - Frage hinzufügen
    - Frage liefern
  - Frage:
    - Fragetext
    - Erzielbare Punkte
    - Frage stellen
    - Frage beantworten
  - Wahr-Falsch-Frage:
    - Spezieller Fragetyp
    - Wahr/falsch
  - Multiple-Choice-Frage:
    - Spezieller Fragetyp
    - Menge an Antworten
    - Richtige Antwort
  - ...

Beispiel Pruefung (4)

```
abstract class Frage {

  String text;     // Fragetext
  int punkte;      // zu erreichende Punktzahl

  Frage(String text, int punkte) {
    this.text = text; this.punkte = punkte;
  }

  // Frage auf den Bildschirm ausgeben
  void frageStellen() { IO.println(this.text); }

  // Frage beantworten durch Prüfling, Antwort auswerten
  // und Punkte vergeben
  abstract void frageBeantworten(Pruefling person);

  int getPunkte() { return this.punkte; }
}
```

Beispiel Pruefung (5)

```
// Klasse, die Wahr/Falsch-Fragen realisiert
class WahrFalschFrage extends Frage {
  boolean richtig;  // richtig oder falsch

  WahrFalschFrage(String text, int punkte,
                  boolean richtig) {
    super(text, punkte); this.richtig = richtig;
  }

  // Frage beantworten durch Prüfling, Antwort auswerten
  // und Punkte vergeben
  void frageBeantworten(Pruefling person) {
   boolean ant = IO.readChar("Wahr o. Falsch (w/f)?") == 'w';
   if (ant == this.richtig) {
     IO.println("Richt. Antw.: "+punkte + " Punkte");
     person.neuePunkte(this.punkte);
    } else {
       IO.println("Falsche Antwort: 0 Punkte");
} }}
```

Beispiel Pruefung (6)

```
// Klasse, die Multiple-Choice-Fragen realisiert
class MCFrage extends Frage {
  String[] antworten;  // moegliche Antworten
  int      richtigIndex;  // Index der richtigen Antwort

  MCFrage(String text, int punkte,
          String[] antworten, int richtigIndex) {
    super(text, punkte);
    this.antworten = antworten;
    this.richtigIndex = richtigIndex;
  }

  // Frage auf den Bildschirm ausgeben
  void frageStellen() {
     super.frageStellen();
     for (int f=0; f<this.antworten.length; f++) {
       IO.println("(" + f + "): " + this.antworten[f]);
  }  }
```

Beispiel Pruefung (7)

```
  // Frage beantworten durch Prüfling, Antwort auswerten
  // und Punkte vergeben
  void frageBeantworten(Pruefling person) {
    int antwort = IO.readInt("Auswahl: ");
    if (antwort == this.richtigIndex) {
      IO.println("Richtige Antwort: " + this.punkte +
                 " Punkte");
      person.neuePunkte(this.punkte);
    } else {
      IO.println("Falsche Antwort: 0 Punkte!");
      IO.println("Richtig Antwort ist " + this.richtigIndex);
    }
  }
}
```

Beispiel Pruefung (8)

```
class Quiz {
  Frage[] fragen;     // Menge an Fragen
  String titel;       // Titel des Quizes
  int aktuellerIndex; // aktuelle Anzahl an Fragen-1
  int naechsterIndex; // Schleifenvariable

  // Konstruktor
  Quiz(String titel, int maxFragen) {
    this.titel = titel;
    this.aktuellerIndex = -1;
    this.fragen = new Frage[maxFragen];
    for (int i=0; i<this.fragen.length; i++)
      this.fragen[i] = null;
    this.naechsterIndex = -1;
  }

  String getTitel() { return this.titel; }
```

Beispiel Pruefung (9)

```
  // Frage hinzufuegen
  void neueFrage(Frage f) {
    if (this.aktuellerIndex < this.fragen.length-1)
      this.fragen[++this.aktuellerIndex] = f;
  }

  // liefert zyklisch die naechste Frage oder null,
  // falls keine (mehr)vorhanden ist
  Frage liefereNaechsteFrage() {
    if (this.fragen.length == 0) return null;
    Frage f = null;
    if (this.naechsterIndex < this.aktuellerIndex)
      f = this.fragen[++this.naechsterIndex];
    else
      this.naechsterIndex = -1;
    return f;
  }
}
```

Beispiel Pruefung (10)

```
class Pruefling {
  String name;  // Name der Person
  int punkte;   // bisher erzielte Punkte

  Pruefling(String name) {
    this.name = name;
    this.punkte = 0;
  }

  String getName() { return this.name; }
  int getPunkte() { return this.punkte; }

  void neuePunkte(int anzahl) {
    this.punkte += anzahl;
  }
}
```

Beispiel Pruefung (11)

```
class Pruefung {

  // Hauptprogramm
  public static void main(String[] args) {
    Pruefung klausur = new Pruefung();
    klausur.vorbereiten();
    klausur.durchfuehren();
    klausur.ergebnisseBekanntgeben();
  }

  Quiz pruefung;
  Pruefling[] studenten;

  Pruefung() {
    this.pruefung = null;
    this.studenten = null;
  }
```

Beispiel Pruefung (12)

```
void vorbereiten() {
  IO.println("Fragen eingeben");
  IO.println("----------------");
  String titel = IO.readString("Titel des Quizes: ");
  int anzahl = IO.readInt("Anzahl Fragen: ");
  this.pruefung = new Quiz(titel, anzahl);
  // Fragen eingeben
  for (int i=0; i<anzahl; i++) {
    Frage f = this.frageErzeugen(i+1);
    this.pruefung.neueFrage(f);
  }
}
```

Beispiel Pruefung (13)

```
public Frage frageErzeugen(int nummer) {
   // spaeter Factory-Pattern
   int typ = IO.readInt(
     "Fragetyp: Wahr/Falsch (1), Multiple Choice (2)?");
   switch (typ) {
     case 1: return erzeugeWahrFalschFrage(nummer);
     case 2:
     default: return erzeugeMCFrage(nummer);
   }
}

private Frage erzeugeWahrFalschFrage(int nummer) {
   String text = IO.readString("Frage "+ (nummer) + ": ");
   boolean wahr = IO.readChar("Wahr/falsch(w/f)?") == 'w';
   int punkte = IO.readInt("Erreichbare Punkte: ");
   return new WahrFalschFrage(text, punkte, wahr);
}
```

Beispiel Pruefung (14)

```
private Frage erzeugeMCFrage(int nummer) {
   String text = IO.readString("Frage "+ (nummer) + ": ");
   int anzahl = IO.readInt("Anzahl an Antworten: ");
   String[] antworten = new String[anzahl];
   for (int i = 0; i < anzahl; i++) {
     antworten[i] = IO.readString("Antwort " + i + ": ");
   }
   int richtigIndex =
      IO.readInt("Index der richtigen Antwort: ");
   int punkte = IO.readInt("Erreichbare Punkte: ");
   return
      new MCFrage(text, punkte, antworten, richtigIndex);
}
```

Beispiel Pruefung (15)

```
void durchfuehren() {
  IO.println("Pruefung");
  IO.println("--------");
  int anzahl = IO.readInt("Anzahl Prueflinge: ");
  this.studenten = new Pruefling[anzahl];
  // alle Prueflinge abfragen
  for (int i=0; i<anzahl; i++) {
    IO.println("Pruefling " + (i+1) + " ist an der Reihe");
    this.studenten[i] =
      new Pruefling(IO.readString("Name: "));
    Frage f = null;
    // alle Fragen der Pruefung stellen
    while ((f = this.pruefung.liefereNaechsteFrage()) !=
            null) {
      f.frageStellen();
      f.frageBeantworten(this.studenten[i]);
    }
} }
```

Beispiel Pruefung (16)

```
void ergebnisseBekanntgeben() {
  this.ranglisteErstellen();
  IO.println("Pruefungsergebnisse");
  IO.println("--------------------");
  IO.println("Quiz: " + this.pruefung.getTitel());
  for (int i=0; i<this.studenten.length; i++) {
    IO.println("Platz " + (i+1) + ": " +
               this.studenten[i].getName() + " mit " +
               this.studenten[i].getPunkte() + " Punkten");
  }
}
```

Beispiel Pruefung (17)

```
  private void ranglisteErstellen() {
    // Bubblesort nach erreichten Punkten
    boolean veraendert = false;
    do {
      veraendert = false;
      for (int i=0; i<this.studenten.length-1; i++) {
        if (this.studenten[i].getPunkte() <
            this.studenten[i+1].getPunkte()) {
          Pruefling help = this.studenten[i];
          this.studenten[i] = this.studenten[i+1];
          this.studenten[i+1] = help;
          veraendert = true;
        }
      }
    } while (veraendert);
  }
}
```

Beispiel java.util.Calendar (1)

```
package java.util;

public abstract class Calendar {
    public static final int JANUARY;
    public static final int YEAR;
    public static final int MONTH;
    ...
    public void set(int field, int value)
    public int get(int field)
    ...
}

public class GregorianCalendar extends Calendar {
    public GregorianCalendar()   // aktuelle Zeit
    ...
}
```

Beispiel java.util.Calendar (2)

```
import java.util.Calendar;
import java.util.GregorianCalendar;


class Datum {
   public static void main(String[] args) {
      Calendar cal = new GregorianCalendar();
      IO.println("Jahr: " + cal.get(Calendar.YEAR));
      IO.println("Monat: " + cal.get(Calendar.MONTH));
      IO.println("Tag: " + cal.get(Calendar.DAY_OF_MONTH));
      cal.set(Calendar.YEAR, 2000);
      cal.set(Calendar.MONTH, Calendar.JANUARY);
      cal.set(Calendar.DAY_OF_MONTH, 1);
      if ((new GregorianCalendar()).after(cal))
            IO.println("im 21. Jahrhundert");
   }
}
```

Zusammenfassung

- Abstrakte Klasse: eine bewusst unvollständige (Ober)Klasse, in der von einzelnen Methodenimplementierungen abstrahiert wird
- Fehlende Methodenrümpfe werden erst in abgeleiteten Unterklassen implementiert
- Die Instantiierung abstrakter Klassen ist nicht möglich; es lassen sich wohl aber Objektvariablen definieren, womit Polymorphie/dynamisches Binden ausgenutzt werden kann