Objektorientierung II

Java Foundation Track by Carsten Bokeloh

Vererbung & Polymorphismus

- Warum Vererbung
- super-Referenz
- Überschreiben von Methoden und Variablen
- java.lang.Object
- Abstrakte Klassen und Interfaces

Warum Vererbung

```
package inheritance;

public abstract class Person {
    /** Gibt den Wert des Objektes als int zurück.*/
    public abstract int hoeheDerSteuern();
}
```

Warum Vererbung

- Instanzen der Klasse Person sollen eine Instanzmethode hoeheDerSteuern() haben
- wir wissen für die konkreten Ausprägungen aber noch nicht, wie die Methode aussehen soll
- durchs Schlüsselwort abstract weiß der Compiler, dass diese Klasse keinen konkreten „Bauplan“ darstellt und somit nicht instantiierbar ist. Aber wozu das ganze?

Warum Vererbung

```
package inheritance;

public class Angestellter extends Person{
    private int steuersatz;
    private int gehalt;

    public Angestellter(int steuersatz, int gehalt) {
        this.steuersatz = steuersatz;
        this.gehalt = gehalt;
    }

    public int hoeheDerSteuern() {
        return steuersatz * gehalt;
    }


}
```

Warum Vererbung

- hoeheDerSteuern bei beamten wird anders berechnet
- hoeheDerSteuern für Kinder dürfte 0 sein.
- …

- um alle Steuern zu ermitteln iteriere ich also einfach über alle Personen und berechne die Steuern

Vererbung

- AllgemeineKlasse a = new AllgemeineKlasse()
- AllgemeineKlasse a = new SpezielleKlasse()
- aber nicht
- AllgemeineKlasse a = new AllgemeineKlasse();
- SpezielleKlasse s = a; //unzulässig
- SpezielleKlasse s = (SpezielleKlasse)a; //kompiliert erzeugt aber ClassCastException

- AllgemeineKlasse a = new SpezielleKlasse();
- SpezielleKlasse s = (SpezielleKlasse) a;

Vererbung

- Vererbung kann ich verhindern durch:

final

```
class Flower {

	public void fragrance() {
		System.out.println("Flower");
	}
}
public class Rose {

	public void fragrance() {
		System.out.println("Rose");
	}

}
public class Lily {

	public void fragrance() {
		System.out.println("Lily");
	}

}
public class Bouquet {

	public void arrangeFlowers() {
		Flower f1 = new Rose();
		Flower f2 = new Lily();
		f1.fragrance();
	}

}
```

a.) The output of the code is Flower

b.)The output of the code is Rose

c.)The output of the code is Lily

d.)The code fails to compile

super

- wenn ich mit dem was meine Eltern können zufrieden bin, ruf ich einfach super.
- auch in Konstruktiven wird zunächst super aufgerufen.

Überschreiben von Methoden und Variablen

- Dynamisches Binden
- Das Prinzip des Polymorphismus bzw. das dynamische Binden greift lediglich bei Methoden nicht bei Variablen
- Überschreiben verhindern ebenfalls durch **final**

Die Klasse java.lang.Object

- Urmutter aller Klassen
  - von Object erben **alle**
- public String toString()
- public boolean equals(Object obj)
- public int hashCode()
- !!!Wenn Du die equals Methode überschreibst, dann musst Du auch die hashCode-Methode überschreiben

equals

- The equals method implements an equivalence relation:
- It is **reflexive**: for any reference value x, x.equals(x) should return true.
- It is **symmetric**: for any reference values x and y, x.equals(y) should return true if and only if y.equals(x) returns true.
- It is **transitive**: for any reference values x, y, and z, if x.equals(y) returns true and y.equals(z) returns true, then x.equals(z) should return true.
- It is **consistent**: for any reference values x and y, multiple invocations of x.equals(y) consistently return true or consistently return false, provided no information used in equals comparisons on the object is modified.
- For any non-null reference value x, x.equals(null) should return false.

hashcode

- The general contract of hashCode is:
- Whenever it is invoked on the same object more than once during an execution of a Java application, the hashCode method must consistently return the same integer, provided no information used in equals comparisons on the object is modified. This integer need not remain consistent from one execution of an application to another execution of the same application.
- If two objects are equal according to the equals(Object) method, then calling the hashCode method on each of the two objects must produce the same integer result.
- It is not required that if two objects are unequal according to the equals(java.lang.Object) method, then calling the hashCode method on each of the two objects must produce distinct integer results. However, the programmer should be aware that producing distinct integer results for unequal objects may improve the performance of hashtables.

Im Buch gibt es mehr Infos unter Kapitel 5.8

http://openbook.galileocomputing.de/javainsel/javainsel_05_008.html#dodtpb085bfe8-9ffe-48a2-bfb4-87c3b1b69ee8

Übung

Schreiben Sie eine Klasse Aktenordner mit den Instanzvariablen anzahlBlätter(int) und farbe(String). Schreiben Sie einen Kontsruktor, der diese Variablen mit sinnvollen Werten belegt. Daraufhin erzeugen Sie für das Objekt eine sinnvolle toString Methode.
Implementieren Sie in der selben Klasse ein main Methode und installieren ein oder mehrere Aktenordner und machen Verwendung von der toString Methode.

Abstrakte Klassen und Interfaces

- Abstrakte Klassen können abstrakte Methoden und ausimplementierte Methoden haben
- Interfaces enthalten ausschließlich abstrakte Methoden. Deswegen kann das Wort abstract auch weggelassen werden
- Eine Klasse kann mehrere Interfaces implementieren aber nur von einer abstrakten Klasse abgeleitet sein.

Frage

- Die Klasse java.lang.Math stellt eine Sammlung von mathematischen Standardfunktionen dar, die allesamt als static definiert sind. Weil die Klasse über keine Instanzmethoden oder -variablen verfügt, wäre eine Erzeugung von Objekten dieser Klasse recht unsinnig. Um dies zu verhindern, haben ihre Programmierer einen Trick angewendet. Wie konnten sie eine Instantiierung verhindern

Ein Interface Farben wird von zwei anderen Interfaces FarbenHell und FarbenDunkel abgeleitet.
Alle drei Interfaces definieren int-Konstanten, hinter welchen sich Namen von Farben verbergen. Das Interface Farben definiert zusätzlich eine Methode defFarben() mit einem Parameter vom Typ int.
Definieren Sie eine Klasse FarbenAuswahl, die das Interface Farben implementiert.
Für die Zuordnung der Farbennummern soll ein statisches String-Array definiert werden. Bei der Initialisierung im static-Block werden verschiedene Möglichkeiten gezeigt, wie man die in den Interfaces definierten Konstanten ansprechen kann. In der implementierten Methode defFarben() wird die, über das im Methodenaufruf übergebende Argument, ausgewählte Frage am Bildschirm angezeigt.
Die Klasse FarbenTest erzeugt ein Objekt der Klasse Farbenauswahl und ruft in einer for-Schleifederen Methode für die konstanten Werte von 1-6 auf.