HSLU Hochschule Luzern

Objektorientierte Programmierung (OOP)

Übung SW05: Vererbung / Entwicklungsumgebung

Themen: Vererbung (O07) / Arbeit mit Entwicklungsumgebung (E01)
Zeitbedarf: ca. 240min.

Roland Gisler, Version 1.5.0 (HS24)

1 Vererbung

1.1 Lernziele

- Vererbungsbeziehungen umsetzten.
- Merkmale von guter und schlechter Vererbung erkennen.
- Verwendung abstrakter Klassen.

1.2 Grundlagen

Im Input O07-Vererbung finden Sie ein UML-Modell das die Klassen `Shape`, `Circle` und `Rectangle` modelliert. Beachten Sie auch die Grundlagen zu abstrakten Klassen und Interfaces im Input O06-Schnittstellen der Vorwoche.

1.3 Aufgaben

a.) Starten Sie wieder in einem neuen Projekt und setzen Sie das UML-Modell des Vererbungsbeispiels aus dem Input um. Sie können dafür einzelne Codefragmente aus dem Input übernehmen.

b.) Vervollständigen Sie die Implementation und testen Sie das Modell aus, indem Sie Objekte von `Circle` und `Rectangle` erzeugen.

c.) Sie sollen neu eine Klasse `Square` (Quadrat) ergänzen. Welche verschiedenen Varianten haben Sie dazu das objektorientiert umzusetzen? Überlegen Sie sich gut die Vor- und Nachteile der verschiedenen Varianten. Diskutieren Sie mit anderen Studierenden.

d.) Implementieren und testen Sie mindestens eine Ihrer Designvarianten der `Square`-Klasse!

e.) Vor zwei Wochen haben wir eine Übung zum `switch`-Statement gemacht, bei welchem es darum ging, von verschiedenen chemischen Elementen die Aggregatszustände abfragen zu können. Vielleicht haben Sie bereits da versucht mit Klassen zu arbeiten. Nun haben Sie das notwendige Rüstzeug: Entwerfen Sie ein Design, welches eine (abstrakte) Basisklasse nutzt und für jedes chemische Element eine eigenständige Klasse definiert!

f.) Implementieren Sie Ihr Design von e) und testen Sie auch dieses aus!

g.) Letzte Woche haben wir mit dem `Switchable`-Interface gearbeitet. Davon könnte es eine Spezialisierung geben: Das `CountingSwitchable`, welches die Abfrage der Anzahl Schaltvorgänge über die Methode `long getSwitchCount()` erlaubt. Setzen Sie diese zusätzliche Anforderung um und testen Sie!

h.) Jemand kommt auf die Idee ein `Named`-Interface zu definieren, mit welchem beliebige Objekte einen Namen (Methoden `void setName(final String name)` und `String getName()`) erhalten können. Welche verschiedenen Möglichkeiten haben Sie, dies für Ihre «geschalteten» Klassen oder Interfaces zu nutzen?

2 Entwicklungsumgebung

2.1 Lernziele

- Arbeit mit einer professionellen Entwicklungsumgebung.
- Projektstruktur von IDE-Projekten kennenlernen.
- Verwenden von Java-Packages.

2.2 Grundlagen

Alle bisherigen Übungen haben Sie in BlueJ gemacht. Ab sofort empfehle ich Ihnen alle Übungen welche **nicht** aus dem Buch «Objects First with Java» stammen, in der Entwicklungsumgebung Ihrer Wahl, basierend auf dem «Template»-Projekt lösen!

2.3 Aufgaben

a.) Installieren und Konfigurieren Sie die Entwicklungsumgebung Ihrer Wahl.
Befolgen Sie dazu die gesonderte und auf ILIAS abgelegt Anleitung **`OOP_JavaDevelopmentManual_jdk21.pdf`**, konkret die Kapitel **3** bis und mit **6**.

b.) Verwenden Sie das in der Anleitung von a) erwähnte und ebenfalls auf dem ILIAS zur Verfügung gestellte Template-Projekt (**`oop_maven_template_jdk21-5.2.0.zip`**[1]) und erstellen Sie in Ihrem gewünschten Arbeitsverzeichnis ein neues Projekt.

c.) Öffnen Sie das Projekt und schauen Sie es sich mittels der verschiedenen Views Ihrer IDE an. Was sehen Sie wo am besten? Verschaffen Sie sich einen Überblick! Lassen Sie sich von der grossen, vielleicht erschlagenden Funktionsvielfalt Ihrer IDE nicht abschrecken.

d.) Erstellen Sie eine Java-Packagestruktur, als Basisname empfehle ich Ihnen z.B. **`ch.hslu.oop`**, und dann erweitern Sie jeweils auf die Semesterwoche (z.B. **`sw05`**) oder auf die Themen-ID (**`oop07`** etc.). Auf diese Art können Sie alle Beispiele und Übungen des Modules OOP sauber geordnet in einem einzigen Projekt ablegen (was deutlich weniger Aufwand bedeutet).

e.) Nun machen wir ein Refactoring: Kopieren Sie ausgewählte Quellen der bisherigen Übungen (die **nicht** aus dem Buch kommen) aus den einzelnen BlueJ-Projekten in die jeweiligen Package-Verzeichnisse des IDE-Projektes.
Hinweis: Sie müssen dafür die Package-Deklaration in den Klassen anpassen, gute IDE's weisen Sie darauf hin und machen das (fast) automatisch. (Re-)Formatieren Sie gleichzeitig den Quellcode in Ihrer IDE!

f.) In der IDE können Sie nun nicht mehr interaktiv Objekte (wie in BlueJ) erstellen. Stattdessen implementieren Sie alle Schritte die Sie bisher manuell und interaktiv durchgeführt haben innerhalb einer **`public static void main(final String[] args)`**-Methode, wie Sie sie z.B. im Input **A01** am Schluss finden. Führen Sie diese **`main()`**-Methoden aus und testen Sie ob alles wie gewünscht funktioniert!

g.) Bereiten Sie **mindesten** die Übungen der **letzten beiden** Wochen auf diese Weise auf, noch besser aber alle!

Hinweis: Weil Sie in der IDE die Programme nicht mehr wie bei BlueJ interaktiv ausführen können, ist es auch nicht mehr so einfach möglich die aktuell vorhandenen Objekte zu untersuchen (inspect). Greifen Sie dafür temporär auf **`System.out.println(…)`** zurück, um nützliche Ausgaben zu produzieren, über welche Sie die korrekte Funktion Ihrer Programme verifizieren können. Keine Angst, wir werden in Kürze noch bessere Mittel und Wege kennenlernen. Wer schon Erfahrung mit Debugging hat, kann sich natürlich auch damit behelfen.

[1] Es kann auch eine neuere Version vorliegen.