3 Highscore Kapselung

Kapseln Sie Ihre `Highscore`-Methoden aus Serie 2 in eine **wiederverwendbare** Klasse. Häufig spricht man von *"Separation of Concerns"*. Gemeint ist damit, dass sinngemäss Zusammengehöriges als Einheit erkennbar ist und sich von dessen Verwendung und anderen Einheiten abgrenzt.

3.1 Kapselung

Im vorliegenden Beispiel hängen alle Methoden ausser `main` direkt mit dem Highscore zusammen. Wir wollen nun diese Methoden schön ordentlich in eine eigene Klasse "`Highscore`" packen.

a) Falls Ihre Lösung noch Fehler aufweist, finden Sie Hilfe in den Folien von der zweiten Übungslektion.

b) Erstellen Sie im Datei-Explorer o.ä. für alle Fälle ein Backup ihres Projekts oder mindestens der entsprechenden Java-Datei.

c) Fügen Sie Ihrem Projekt eine neue Klasse "`Highscore`" **ohne** eigene `main`-Methode hinzu.

d) Verschieben Sie (mittels Ausschneiden & Einfügen) **alles** ausser der `main`-Methode aus der Klasse `HiLoSimple` in die Klasse `Highscore`. Die neue Klasse ist damit in einer ersten Version bereits syntaktisch korrekt. Sie enthält das Feld mit dem Highscore-Array und drei Methoden.

e) Die `main`-Methode in der HiLo-Klasse muss aber noch angepasst werden. Zur Zeit fehlt jeglicher Zugriff auf die verschobenen Methoden. Deswegen erstellen wir zu Beginn der `main`-Methode ein Objekt der `Highscore`-Klasse:

```
1 Highscore hs = new Highscore();
```

*Nebenbei: Das nennt sich Wiederverwendung durch Aggregatation. Aggregatation bezeichnet eine **has-a-Relation**. Sprich: Unsere HiLo-Klasse **hat einen** Highscore. Logisch, oder?*

Den Objektnamen (hier "`hs`") dürfen Sie frei wählen. Ich habe mich für eine Abkürzung entschieden, weil es kürzer ist. Die ausgeschriebene Form "`highscore`" wäre aber auch gut. Per Konvention, sollten Objekt (wie Variablen) kleingeschrieben werden.

f) Bei zwei Zeilen zeigt Eclipse nun einen Fehler an:

```
1 insertScore(score);
2 showHighscore();
```

Beheben Sie dies, indem Sie den beiden Methoden-Aufrufen die Objektreferenz ("`hs.`") voranstellen:

```
1 hs.insertScore(score);
2 hs.showHighscore();
```

g) Eclipse hat jetzt noch eine Warnung parat: "The static method insertScore(int) from the type Highscore should be accessed in a static way". Das ist einfach zu lösen, entfernen Sie sämtliche `static` aus Ihrer Klasse `Highscore`. Die `static` in der `HiLo`-Klasse müssen bleiben.

h) Das war schon alles. Testen Sie, ob die Applikation immer noch funktioniert.

*Nebenbei: Schritte wie das Kapseln einer Klasse nennt man **Refactoring**. Im Allgemeinen, dient Refactoring, um die Code-Qualität zu verbessern ohne die Funktion des Programmes zu beeinflussen.*

3.2 Umbenennen

Mit **Refactoring** machen wir gleich weiter. Und zwar sind die Methodennamen in unserer Klasse `Highscore` noch nicht sehr elegant. Wir wählten die Namen ursprünglich, um klarzustellen, dass diese Methoden der Verwaltung des Highscore dienen. Diese Zugehörigkeit ist nun durch die Klasse gegeben und muss sich **nicht mehr** in schwerfälligen Methodennamen spiegeln.

Die Wörter `Score` und `Highscore` dürfen aus den **Methodennamen** verschwinden (Konstruktor ausgenommen). Eclipse unterstützt Sie in diesem Unterfangen:

a) Klicken Sie auf einen Methodennamen, so dass Eclipse die Zeile hellblau hinterlegt.

b) Klicken Sie nun mit der rechten Maustaste auf den Methodennamen.

c) Aus dem Kontextmenü wählen Sie "Refactor" und dann "Rename" (Shift+Alt+R).

d) Nun können Sie den Namen ändern. Aus `showHighscore()` wird z. B. ein einfaches `show()`.

e) Bestätigen Sie Ihre Eingabe mit Enter. Eclipse passt den Namen automatisch an allen Stellen im Programm an.

f) Wiederholen Sie dies für alle drei Methoden der Klasse `Highscore` (ausser dem Konstruktor).

Das Umbenennen mag perfektionistisch anmuten, aber nur so offenbart sich die Eleganz der Objektorientierung.

3.3 Parameter

Jetzt sind es immer genau 10 Plätze in der Highscore-Tabelle. Zu Verbesserung der Wiederverwendbarkeit, möchten wir die **Anzahl** der Highscoreplätze **konfigurierbar** machen.

a) Definieren Sie in der `Highscore`-Klasse eine private `int`-Variable `size`. Durch die Initialisierung mit 10 stellen wir sicher, dass die Tabellen-Grösse per Default 10 ist.

```
1 private int size = 10;
```

b) Das Setzen von `size` von aussen wollen wir mittels einem Konstruktorparameter implementieren. Der Konstruktor ist eine spezielle Methode, die per Definition den gleichen Namen trägt, wie die Klasse. Der Konstruktor wird beim Erstellen eines Objekts (mit `new`) automatisch aufgerufen.

```
1 Highscore hs = new Highscore();
```

Hier bezeichnet `Highscore` (auf der linken Seite der Zuweisung) die Klasse. Die Klasse liefert den Bauplan für das zu erzeugende Objekt. Die Variable `hs` speichert die Referenz auf das neu erstellte Objekt. Nach dem `new` folgt `Highscore()` auf der rechten Seite der Zuweisung. Der Aufruf `Highscore()` bezeichnet nicht direkt die Klasse, sondern den zugehörigen Konstruktor. Die Klammern sind das Indiz, dass es sich um einen Methodenaufruf handelt.

c) Sie mögen nun einwenden, dass wir bis jetzt noch gar keinen Konstruktor definiert haben. Es stimmt dass wir noch keinen expliziten Konstruktor definiert haben. Jedoch erstellt Java hinter den Kulissen stets einen Default-Konstruktor.

Der Default-Konstruktor akzeptiert aber **keine** Parameter. Daher erstellen wir nun einen eigenen Konstruktor.

```
1 public Highscore(int s) {
2 }
```

d) Nun müssen Sie das Feld `size` im Konstruktor noch auf den gegeben Parameterwert `s` setzen. Füllen Sie den Konstruktor mit der Entsprechenden Zuweisung.

e) Die Grösse der Highscore-Tabelle wird in folgender Zeile festgelegt:

```
1 private int[] highscore = new int[10];
```

Wenn wir nun dort die Grösse 10 einfach durch `size` ersetzen, wird immer der Default-Wert von `size` verwendet, weil der Konstruktor noch gar nicht dazu kam, einen neuen Wert zu definieren. Stattdessen müssen wir den `Array` nun erst im Konstruktor initialisieren. Die Deklaration `private int[] highscore;` verbleibt an Ort und Stelle. Aber im Konstruktor kommt folgende Zeile hinzu:

```
1 highscore = new int[s];
```

f) Finden Sie alle Stellen in Ihren Algorithmen, an denen nun die Variable `size` anstelle einer fixen Zahl verwendet werden muss?

g) Als letzter Schritt müssen wir den neuen Konstruktor nun auch verwenden. Es gibt keinen Default-Konstruktor mehr. Java erstellt diesen nur, wenn kein anderer Konstruktor vorhanden ist. Schreiben Sie dazu die gewünschte `Highscore`-Grösse (z. B. 12) in die Klammern des Konstruktor-Aufrufs in der `main`-Methode.

3.4 Spielerangaben

Diese Teilaufgabe ist **freiwillig** und für alle die noch etwas tüfteln wollen. Im bisherigen Code besteht der Highscore nur aus einer Liste von Punktezahlen. Der Spieler, der diese Punktezahl jeweils erreicht hat, kommt namentlich nicht zu Ehren.

Ziel dieser Aufgabe ist es, dies zu ändern. Fügen Sie Ihrem Projekt eine weitere Klasse "`Player`" hinzu. Die Klasse soll folgende Daten kapseln:

- `String name; // Spielername`
- `int score; // Punktestand`

Das genaue Design einer Klasse ist Geschmackssache. Meine Ideen sind meilenweit von einer absoluten Wahrheit entfernt.

Persönlich würde ich den Spielernamen als `public final String name;` in die Klasse einbauen. Der Modifizierer `final` erlaubt nur ein **einmaliges** Beschreiben. Das heisst die `String`-Variable kann von ausserhalb der Klasse nicht beschrieben werden. Beschrieben wird die String-Variable vom Konstruktor, der einen entsprechenden Parameter entgegennimmt. Der Spielername ist damit ab der Initialisierung mit dem Objekt fix verbunden. Wenn der Name ändert, muss dadurch ein neues Objekt erstellt werden.

Den `score` hingegen können sie als öffentliche `int`-Variable implementieren. So fallen auch die Anpassungen in `main` einfach aus: Anstelle von `score = ...;` heisst es dann einfach `p.score = ...;`, wenn `p` das lokale `Player`-Objekt ist.

Erweitern Sie Ihre Highscoreklasse, so dass sie neu mit Objekten von `Player` anstelle von `int`-Zahlen arbeitet. Ergänzen Sie auch die `main`-Methode, so dass der Spielername vor dem Spiel jeweils abgefragt wird. Falls das nicht klappen sollte, verwenden Sie einen fixen Namen im Code (z. B. "`Fritz`").

*Nebenbei: Findige StudentInnen beschwerten sich bereits, weil die `Highscore`-Tabelle nur während der Ausführungszeit bestehen bleibt. Das ist richtig und selbstverständlich noch nicht brauchbar. Das gewünschte Feature ist **Persistenz**. Haben Sie Geduld, wir kommen noch dazu.*