Anmerkungen zur Sichtbarkeit

Im Text zur Vererbung [OO-02 Vererbung] heißt es:

protected

Gegenüber der Methode `gibAktuelleFigur()` der ursprünglichen Klassendefinition hat sich außerdem die Angabe zur Sichtbarkeit geändert. Weshalb das notwendig ist, kann man leicht feststellen, wenn man es bei `private` belässt: Bei dieser Definition kann die Methode, die nun in einer anderen Klasse steht, nicht aufgerufen werden. `protected` macht eine Methode innerhalb des gesamten Paketes sichtbar, nicht aber nach außen.

Paketsichtbarkeit

Nun gibt es allerdings auch den Begriff der Paketsichtbarkeit. Sie ist dann gültig, wenn es zu einem Attribut keinen Modifizierer der Sichtbarkeit gibt, die Deklaration also beispielsweise einfach

```
int x;
```

lautet. BlueJ gibt beispielsweise im Mustercode für eine neue Klasse diese Sichtbarkeit vor.
Dabei stellt sich die Frage: Was unterscheidet protected von Paketsichtbarkeit? In Galileo-Computing "Java ist eine Insel" findet man eine übersichtliche Zusammenstellung zur Sichtbarkeit:
"In Java gibt es vier Sichtbarkeiten und drei Sichtbarkeitsmodifizierer:

- *Öffentliche Typen und Eigenschaften deklariert der Modifizierer `public`. Die Typen sind überall sichtbar, also kann jede Klasse und Unterklasse aus einem beliebigen anderen Paket auf öffentliche Eigenschaften zugreifen. Die mit public deklarierten Methoden und Variablen sind überall dort sichtbar, wo auch die Klasse sichtbar ist. Bei einer unsichtbaren Klasse sind auch die Eigenschaften unsichtbar.*
- *Der Modifizierer `private` ist bei Typdeklarationen seltener, da er sich nur dann einsetzen lässt, wenn in einer Compilationseinheit (also einer Datei) mehrere Typen deklariert werden. Derjenige Typ, der den Dateinamen bestimmt, kann nicht privat sein, doch andere Typen, und innere Klassen, dürfen unsichtbar sein – nur der sichtbare Typ kann sie dann verwenden. Die mit private deklarierten Methoden und Variablen sind nur innerhalb der eigenen Klasse sichtbar. Eine Ausnahme bilden innere Klassen, die auch auf private Eigenschaften der äußeren Klasse zugreifen können. Wenn eine Klasse erweitert wird, sind die privaten Elemente für Unterklassen nicht sichtbar.*
- *Während private und public Extreme darstellen, liegt die Paketsichtbarkeit dazwischen. Sie ist die Standard-Sichtbarkeit und kommt ohne Modifizierer aus. Paketsichtbare Typen und Eigenschaften sind nur für die Klassen aus dem gleichen Paket sichtbar, also weder für Klassen noch Unterklassen aus anderen Paketen.*
- *Der Sichtbarkeitsmodifizierer `protected` hat eine Doppelfunktion: Zum einen hat er die gleiche Bedeutung wie Paketsichtbarkeit, und zum anderen gibt er die Elemente für Unterklassen frei. Dabei ist es egal, ob die Unterklassen aus dem eigenen Paket stammen (hier würde ja die Standard-Sichtbarkeit reichen) oder aus einem anderen Paket. Eine Kombination aus private protected wäre wünschenswert, um die Eigenschaften nur für die Unterklassen sichtbar zu machen*

und nicht gleich für die Klassen aus dem gleichen Paket."

Weiter wird die Historie diskutiert. Man würde sich nämlich eine funktionierende Kapselung in der Vererbungslinie wünschen und dazu hat es auch schon Lösungen gegeben, die allerdings später von den Sprachentwicklern verworfen wurden.
So ist leider keine vollständige Kapselung von Daten möglich. Dass es auch dafür Begründungen gibt, ist zwar nachvollziehbar, stellt aber aus meiner Sicht einen echten Mangel von Java dar.