Kompilieren von Packages

- Die Package `name` befindet sich im Unterverzeichnis `name` des aktuellen Verzeichnisses (folder, directory, Ordner).
- Man kompiliert die Datei `datei.java` in der Package `name` mit dem Befehl

  ```
  javac name/datei.java
  ```

  (Windows: `/` durch `\` ersetzen)
- Es genügt meistens, die Datei mit der `main()` oder `paint()`-Methode zu kompilieren, da benutzte Dateien automatisch mitkompiliert werden.
- Ausführen des Programms durch den Befehl `java name.Klasse` wobei `Klasse` die Klasse mit der `main`-Methode ist.
- Bei Applets verwendet man `code="name/Klasse.class"` oder auch `code="name.Klasse"`

Spezifikation von Methoden: Vor- und Nachbedingungen

Man kann das Verhalten von Methoden durch eine Vor- und Nachbedingung (wie in der Hoare-Logik) beschreiben.

Dazu fügt man in der javadoc Dokumentation an geeigneter Stelle eine *Vorbedingung* und eine *Nachbedingung* ein.

Die Vorbedingung bezieht sich auf die Werte der Parameter der Methode und die Werte der Instanzvariablen vor Ausführung der Methode.

Die Nachbedingung bezieht sich auf die Werte der Parameter (vor Ausführung der Methode), die Instanzvariablen nach Ausführung der Methode und den Rückgabewert. Auf die Werte der Instanzvariablen vor Ausführung der Methode kann man auch Bezug nehmen durch entsprechende Kennzeichnung, etwa durch `_alt`.

Vor- und Nachbedingungen sind für uns *informell*.

Es gibt Werkzeuge wie JML, in denen Vor- und Nachbedingungen formalen Status haben und überprüft werden.

Spezifikation von Methoden: Vor- und Nachbedingungen

Beispiel:

```
/** Einzahlen.
   @param betrag der eizuzahlende Betrag.
   Vorbedingung: betrag >= 0.
   Nachbedingung: kontostand = kontostand_alt + betrag
*/
public void einzahlen(double betrag) {
    kontostand = kontostand + betrag;
}
```

Spezifikation von Methoden und Klassen: Invarianten

Oft soll eine bestimmte Bedingung an die Instanzvariablen von jeder Methode erhalten werden.

Statt diese in jede Vor- und Nachbedingung aufzunehmen kann man solch eine Bedingung als *Invariante* der Klasse spezifizieren.

Jede Methode muss dann diese Invariante nach ihrer Ausführung garantieren; im Gegenzug darf die Invariante vor Ausführung jeder Methode angenommen werden.

```
/** Klasse fuer Bankkonten mit Ueberziehungsmoeglichkeit.
    (Invariante: kontostand >= -limit).
*/
public class Bankkonto {
    /** Der Kontostand. */
    private double kontostand;
    /** Ueberziehungslimit. */
    private double limit;
```

Spezifikation von Klassen und Methoden: Zusammenfassung

Man kann die Beschreibung des Verhaltens von Methoden und Klassen in Vor- und Nachbedingungen, sowie Invarianten gliedern.

Die Vorbedingung einer Methode legt Bedingungen an ihren Aufrufkontext fest. Die Methode ist immer so aufzurufen, dass die Vorbedingung erfüllt ist.

Die Nachbedingung einer Methode spezifiziert den Zustand des Objekts nach Aufruf der Methode. Die Methode ist so zu implementieren, dass die Nachbedingung immer erfüllt ist, vorausgesetzt die Vorbedingung war erfüllt.

Die Invariante einer Klasse ist eine implizite Vor- und Nachbedingung für alle Methoden einer Klasse: Alle Methoden (einschließlich der Konstruktoren) sind so zu implementieren, dass die Invariante erhalten wird.

Verwender der Klasse können dann voraussetzen, dass alle Objekte der Klasse die Invariante erfüllen.

Spezifikation von Klassen und Methoden: Zusammenfassung

Vor- und Nachbedingungen sind für uns informell. Will man sie formalisieren, so treten nichttriviale Schwierigkeiten auf (exakter Geltungsbereich von Invarianten, Formulierung von Bedingungen ohne Bezugnahme auf private Instanzvariablen, . . .).

Es existieren solche Formalisierungen, z.B.: JML, die zudem das Erfülltsein von Bedingungen teilweise automatisch überprüfen.