- Wir werden nun das Thema Richtlinien vertiefen

- Um die allgemeine Lesbarkeit der Programme zu erhöhen, und um einen Standard zu etablieren, werden insbesondere **Richtlinien für die Codierung** vorgegeben (siehe `[LL10], [Hof13]`):
  - Wahl der Bezeichner
  - Layout (z.B. Einrückung)
  - Aufbau von Schnittstellen
  - Kommentierung
  - ...

- Notationsstile für Bezeichner
  - Pascal Case
    - `BackgroundColor`
    - `LinkedList`
  - Camel Case
    - `indexOf`
    - `toString`
  - Upper Case
    - `MAXEINTRAEGE`
    - `MAX_EINTRAEGE`
  - Lower Case
    - `indexof`
    - `index_of`

- Ungarische Notation
  - Variablenname wird um ein Präfix ergänzt, um den Datentyp vorzuhalten

Typ	Beispiel
Character	`char cKennzeichen;`
Byte	`byte byAlter;`
Integer	`int nAnzahl;`
String	`String sname;`
Pointer	`int* pAnzahl`
...	`...`

Verletzt das DRY-Prinzip

sollte nur für untypisierte Sprachen verwendet werden

- Abhängig von der Programmiersprache sollte zur Vereinheitlichung ein *Coding Style* vorgegeben werden
- Ein *Coding Style* enthält Richtlinien für
  - Schreibweise von Bezeichnern
  - Anweisungen
  - Einrückungstiefe
  - maximale Zeilenlänge
  - Umbrüche / Leerzeichen
  - Deklarationsschemata
  - Dateiorganisation und -namen
  - Programmierpraktiken
- Man kann sich an einem vorhandenen *Coding Style* orientieren
  - Java Coding Style
  - .NET Coding Style
  - Linux Kernel Coding Style
  - ...

Werkzeugunterstützung

- Java Coding Style
  - Namenkonvention

Kategorie	Notation	Beispiel
Package	Lower case	`java.awt.event`
Class	Pascal case	`class LinkedList`
Interface	Pascal case	`interface WindowListener`
Methode	Camel case	`indexOf`
Variable	Camel case	`index`
Konstante	Upper case	`SPEED_OF_LIGHT`

  - Einrückung: 4 Leerzeichen
  - Zeilenlänge: 80 Zeichen
  - Umbrüche: nach Komma, vor Operator
  - ...

http://www.oracle.com/technetwork/java/javase/documentation/codeconvtoc-136057.html

- Linux Kernel Coding Style
  - Namenskonvention: Lower case
  - Einrückung: 8 Zeichen
  - Zeilenlänge: 80 Zeichen
  - Umbrüche: Umgebrochene Zeilen werden immer kürzer und werden nach rechts eingerückt
  - ...

```
int ggt(int a, int b)
{
        while (a != b) {
                if (a > b)
                        a=a-b;
                else
                        b=b-a;
        }
        return a;
}
```

- .NET Coding Style (Design guidelines)
  - Namenskonvention: *Camel case* für Parameter und Klassenelemente (protected), ansonsten *Pascal case*

    http://msdn.microsoft.com/en-us/library/ms229042.aspx

- Die Notationskonventionen haben sich auf das Layout und die Syntax des Quelltextes bezogen
- Es können aber auch Sprachkonventionen getroffen werden
- Spachkonventionen berücksichtigen semantische Besonderheiten einer Sprache
  - Verbot von Sprachkonstrukten
  - Eingeschränkte (spezielle) Anwendung von Sprachkonstrukten

- MISRA-C Sprachkonvention
  - Wird seit 1998 von der *Motor Industry Software Reliability Association* (MISRA) vorgeschlagen

    Guidelines for the Use of the C Language in Vehicle Based Software

  - Regelsatz soll klassische Programmierfehler in der Programmiersprache C vermeiden
  - Erhöhung der Qualität eingebetteter Software durch die Vermeidung von fehleranfälligen Programmstrukturen
  - Mit der Version MISRA-2004 wurde die automatisierte Prüfung der Regel verbessert und der Anwendungsbereich ausgedehnt

    Guidelines for the Use of the C Language in Critical Systems

  - Es existieren 141 Regeln, die in 21 Kategorien aufgeteilt sind

- Regelkategorie der MISRA-2004-Sprachkonvention

Regeln	Kategorie	Regeln	Kategorie
(1.1)-(1.5)	Übersetzungsumgebung	(12.1)-(12.3)	Ausdrücke
(2.1)-(2.4)	Spracherweiterungen	(13.1)-(13.7)	Kontrollstrukturen
(3.1)-(3.6)	Dokumentation	(14.1)-(14.10)	Kontrollfluss
(4.1)-(4.2)	Zeichensatz	(15.1)-(15.5)	Switch-Konstrukt
(5.1)-(5.7)	Bezeichner	(16.1)-(16.10)	Funktionen
(6.1)-(6.5)	Datentypen	(17.1)-(17.6)	Pointer und Arrays
(7.1)	Konstanten	(18.1)-(18.4)	Struct und Union
(8.1)-(8.12)	Deklarationen und Def.	(19.1)-(19.17)	Präprozessor
(9.1)-(9.3)	Initialisierung	(20.1)-(20.12)	Standardbibliothek
(10.1)-(10.6)	Typkonversion (Arithmetik)	(21.1)	Laufzeitfehler
(11.1)-(11.5)	Typkonversion (Pointer)		

aus [Hof13]

- Auszug aus dem MISRA-2004-Regelsatz

Regel	Beschreibung
(1.4)	„The compiler/linker shall be checked to ensure that 31 character significance and case sensitivity are supported for external identifiers.“
(2.2)	„Source code shall only use `/* ...*/` style comments“
(3.2)	„The character set and the corresponding encoding shall be documented“
(4.2)	„Trigraphs shall not be used“
(5.1)	„Identifiers shall not rely on the significance of more than 31 characters“
(6.4)	„Bit fields shall only be defined to be of the type `unsigned int` or `signed int`“
(7.1)	„Octal constants (other than zero) and octal escape sequences shall not be used
(8.5)	„There shall be no definitions of objects or functions on a header file“
(9.1)	„All automatic variables shall have been assigned a value before being used“
(10.6)	„A U-suffix shall be applied to all constants of unsigned type“
(14.1)	„There shall be no unreachable code“

nach [Hof13]

– Für natürlichsprachige Spezifikationen kann z.B. die Befolgung von sprachlichen Regeln als Richtlinie vorgegeben werden

	Regel	**Erläuterung (Beispiele)**
R1	Formulieren Sie jede Anforderung im Aktiv	nicht „die Eingabe wird gelöscht“
R2	Drücken Sie Prozesse durch Vollverben aus	nicht „ist“ und „hat“
R3	Vermeiden Sie unvollständig spezifizierte Verben	Wer macht was?
R4	Vermeiden Sie unvollständig spezifizierte Bedingungen	„Wenn-dann-sonst“
R5	Überprüfen Sie Universalquantoren	Gibt es bei „nie“, „immer“, „alle“ wirklich keine Einschränkungen?
R6	Vermeiden Sie Nominalisierungen	Besser Satz mit „anmelden“ als mit „Anmeldung“
R7	Präzisieren Sie unbestimmte Substantive	Gibt es einen oder mehrere „Bediener“?
R8	Klären Sie die Zuständigkeiten	Wer erzwingt bzw. verhindert etwas?
R9	Vermeiden Sie implizite Annahmen	

nach [LL10] und Rupp et al.

Manuelle Prüfmethoden

Manuelle Prüfmethoden

- Diagnostische Verfahren, um das existierende Qualitätsniveau eines Produktes zu ermitteln
- Ziele der Prüfung sind
  - die Feststellung von Mängeln, Fehlern, Inkonsistenzen und Unvollständigkeiten
  - die Feststellung von Verstößen gegen Vorgaben, Richtlinien, Standards und Pläne
  - formale Abnahme des Prüfobjekts
- Für semantische Überprüfungen geeignet
- Beschreibungsform der Prüfobjekte
  - informal (z.B. Pflichtenheft)
  - semiformal (z.B. Pseudocode)
  - formal (z.B. Quellcode, OOA-Modell)
- Breiter Einsatzbereich (Analyse, Entwurf, Codierung, Integration)

- Folgende Voraussetzungen müssen für den Einsatz manueller Prüfmethoden erfüllt sein:
  1. Aufwand und Zeit müssen eingeplant sein
  2. Jeder Prüfer ist in der Prüfmethode geschult
  3. Prüfergebnisse dürfen nicht für die Beurteilung von Mitarbeitern verwendet werden (Vorgesetzte nehmen nicht an der Prüfung teil)
  4. Prüfmethode muss dokumentiert sein und Einhaltung der Methode muss geprüft werden

- Im Folgenden werden vier manuelle Prüfmethoden vorgestellt

1. Durchsicht
   - Prüfung wird vom Entwickler alleine durchgeführt
   - Hilfreich: Artefakt ausdrucken und Bildschirm verlassen, ruhige Umgebung
   - Selbstverständliche Maßnahme, bevor ein Arbeitsergebnis weitergegeben wird
   - Durchsicht ist effizienter als ein Test (siehe `[LL10]`)

2. Stellungnahme

- Der Autor gibt das Artefakt an Dritte weiter und bittet diese, das Artefakt zu beurteilen
- Auf Grundlage der Rückmeldungen kann er dann das Artefakt überarbeiten

- Die Stellungnahme weist eine Reihe von Nachteilen auf:
  - Aufwand ist nicht geplant
  - Autor ist gleichzeitig Moderator
  - Die Prüfergebnisse sind nicht dokumentiert, die Nacharbeit wird nicht kontrolliert

3. Review (nach [LL10]):

- Formalisierter Prozess zur Überprüfung von schriftlichen Dokumenten
- Rollen
  - *Moderator* : leitet das Review, ist kein Vorgesetzter
  - *Autor* : Urheber des Prüflings, nimmt an der Review-Sitzung teil, um auf Nachfrage Unklarheiten zu beseitigen; ist kein Verteidiger des Prüflings
  - *Gutachter* : Mitarbeiter, der den Prüfling beurteilen kann
  - *Notar* : führt in der Review-Sitzung das Protokoll (evtl. der Moderator)
  - *Review-Team* : alle Teilnehmer des Reviews ohne Autor

- Ablauf eines Reviews
  (1) Autor beantragt ein Review beim Manager (z.B. Projektleiter)
  (2) Manager plant das Review und bestellt einen Moderator
  (3) Moderator (und Manager) legen die Aspekte fest, nach denen das Prüfobjekt später begutachtet werden soll
  (4) Für jeden Aspekt wird mindestens ein kompetenter Prüfer ausgewählt
  (5) Manager/Moderator führen eine Eingangsprüfung durch
  (6) Moderator verteilt mit der Einladung zur Review-Sitzung das Prüfobjekt, den Prüfauftrag und die Referenzunterlagen an die Gutachter
  (7) Jeder Gutachter prüft in der Vorbereitung das Artefakt nach den zugeteilten Gesichtspunkten
  (8) In der Review-Sitzung tragen die Gutachter die entdeckten Mängel vor. Sie erheben, gewichten und protokollieren die Befunde (max. 2 Stunden)
  (9) In einer dritten Stunde können sich Gutachter und Autor ohne Regeln und Protokolle austauschen (Austausch von Lösungsideen)
  (10) Manager entscheidet über Art und Umfang der Nacharbeit
  (11) Die Nacharbeiten werden vom Autor durchgeführt
  (12) Bei umfangreichen Änderungen wird das überarbeitete Dokument einem weiteren Review unterzogen

- Review-Regeln
  - Review-Sitzung ist auf zwei Stunden beschränkt
  - Moderator bricht die Sitzung ab, wenn sie nicht erfolgreich durchgeführt werden kann
  - Das Resultat, nicht der Autor wird geprüft
  - Die Rollen werden nicht vermischt
  - Allgemeine Stilfragen außerhalb der Richtlinien werden nicht diskutiert
  - Entwicklung und Diskussion von Lösungen ist nicht Aufgabe des Reviews
  - Jeder Gutachter darf seine Befunde angemessen präsentieren
  - Die Ergebnisse der Gutachter werden sofort protokolliert

- Die Befunde werden gewichtet:
  - *Kritischer Fehler* (Prüfling unbrauchbar)
  - *Hauptfehler* (Nutzbarkeit merklich beeinträchtigt)
  - *Nebenfehler* (Nutzbarkeit wenig beeinträchtigt)
  - *Gut* (kein Mangel festgestellt)
- Das Review-Team gibt eine der folgenden Empfehlungen ab:
  - *Akzeptieren ohne Änderung*
  - *Akzeptieren mit Änderung*
  - *Nicht akzeptieren*
- Das Review-Protokoll wird von allen Teilnehmern unterschrieben

Hinweis:

- In der Literatur ist auch eine weitere Unterscheidung zwischen Inspektion und Review zu finden `[Bal08]`
- Das Review ist dort weniger formalisiert als die Inspektion

- Der Aufwand für ein Review wird unterschiedlich eingeschätzt
  - In [Bal08] wird als optimale Prüfgeschwindigkeit in der Vorbereitung eine Seite (+/- ½) pro Stunde angegeben
    (es wird darauf hingewiesen, dass die individuelle Prüfgeschwindigkeit im Verhältnis 1:10 variieren kann)
  - Von Frühauf, Ludewig und Sandmayr wurden im Jahr 1995 die folgenden Daten veröffentlicht (aus [Bal08]):

	Dokument	**Code**
Maximaler Umfang für Review	50 Seiten	20 Seiten
Zahl der Gutachter	5 (+ Moderator + Autor)	3 (+ Moderator + Autor)
Review-Vorbereitung relativ	10 Seiten/h	5 Seiten/h
Aufwand Review-Vorbereitung	25 Stunden	12 Stunden
Aufwand Review-Sitzung absolut	14 Stunden	10 Stunden
Summe Review-Aufwand	5 Personentage	3 Personentage
Erstellungsaufwand relativ	2 Seiten/Tag	1 Seite/Tag
Erstellungsaufwand absolut	25 Personentage	20 Personentage
Review zu Erstellungsaufwand	20%	15%

4. Structured Walkthrough

- Einfachere Variante des Review (Aufwand ist geringer, aber auch der Nutzen ist geringer)
- Der Autor lädt Gutachter (Kollegen) zu einer Walkthrough-Sitzung ein
  - Hier ist, falls möglich, ein etwas formalerer Ansatz ratsam
  - Autor meldet Artefakt zur Abnahme beim Projektleiter an
  - Der Projektleiter lädt zur Walkthrough-Sitzung ein
- Der Autor moderiert die Walkthrough-Sitzung und stellt dort seine Arbeitsergebnisse vor
- Die eingeladenen Gutachter stellen (spontane, vorbereitete) Fragen