Clean Code - Bezeichner

Ingo Köster

Diplom-Informatiker (FH)

Auswahl von Bezeichnern

- Gute Namen für Variablen, Methoden, Klassen, etc. benötigt unter Umständen viel Zeit
  - *„Jeder Trottel kann Code schreiben den ein Computer verstehen kann. Ein guter Entwickler schreibt Code den Menschen verstehen können“* Martin Fowler
- Diese Zeit ist gut investiert, da sie im Laufe der Entwicklungszeit viel Zeit einspart
- Entwickler lesen Code deutlich öfter als sie Code schreiben
  - Bis zu Faktor 10

Aussagekraft

- Ein Name sollte aussagen
  - warum er existiert
  - was er tut
  - wie er benutzt wird

- Benötigt ein Name einen Kommentar, ist sein Zweck nicht klar

```
// Dauer in Minuten
double d;
```

- Besser

```
double dauerInMinuten;
```

Was macht diese Methode?

```
public static bool Check(string data)
{
      int sum = 0;
      int len = data.Length;
      for(int i = 0; i < len; i++)
      {
         int add = (data[i] - '0') * (2 - (i + len) % 2);
         add -= add > 9 ? 9 : 0;
         sum += add;
      }
      return sum % 10 == 0;
   }
```

Aussagekraft

- Namen wie `data`, `text`, `temp`, etc. haben in der Regel keine Aussagekraft
- Manche Namen haben sich in der Softwareentwicklung jedoch etabliert, obwohl sie wenig/keine Aussagekraft haben
  - `i` bei for Schleifen
  - `temp` als Variable für den Dreieckstausch (nur in diesem Falle)
  - `e` bei der Verwendung von Events
- Es gibt weitere Beispiele

Unterschiede deutlich machen

- Es gibt gewisse „Leerwörter“, die keine Aussagekraft haben
- Was ist der Unterschied zwischen
  - `Car, CarData, CarInfo?`
  - `Nachricht, dieNachricht?`
- Zusätze zu Bezeichnern sollten nicht String oder gar Variable sein
  - `int zahlVariable;`
  - `String nameString;`

Fehlinformationen

- Fehlinformationen, die beim Lesen in die Irre führen, sind zu vermeiden

- Eine Variable wie hp könnte sein
  - Hewlett Packard
  - Horse Power
  - Hypotenuse
  - Etc.

- Eine Variable mit List (z.B. `carList`) im Namen sollte auch eine Liste sein, da ein solcher Name gewisse Erwartungshaltungen bei Entwicklern weckt

Eindeutigkeit

- Namen sollten sich nicht nur geringfügig unterscheiden
  - `DoSomethingWithL();`
  - `DoSomethingWithI();`

- Besonderes Augenmerk sollte bei der Verwendung von den Buchstaben l, I und O verwendet werden

- Insbesondere, wenn diese alleine oder an wichtigen Stellen stehen, können diese beim Lesen mit der 1 oder 0 verwechselt werden

Eindeutigkeit - Lesbarkeit

```
int l = 1;
int O = 0;
```

```
if (l == O)
{
}
```

Aussprechbare Namen

› Die Wörter von natürlichen Sprachen sind aussprechbar

› Bezeichner im Quellcode sollten dies auch sein

```
bool tgif;
```

› Aussprechbare Bezeichner sind innerhalb des Quelltextes leichter zu suchen

Codierungen vermeiden

- Codierungen für z.B. Typen sollten vermieden werden
- Insbesondere bei stark Typisierten Sprachen ist es nicht nötig den Typ anzugeben
  - Entwicklungsumgebung prüft zur Laufzeit
- Sind Klassen und Methoden klein, ist die Deklaration auf dem gleichen Bildschirm sichtbar
- Wird die Variable `double gehaltDouble` zum Datentyp decimal geändert muss auch der Variablenname geändert werden!

Codierungen

- Schnittstellen werden häufig (gerade in C#) kodiert
- Diese Beginnen mit einem großen I (z.B. `IEnumerable`)
- Hilft Klassen und Schnittstellen zu unterscheiden
- Diese Codierung sollte innerhalb einer Sprache wie C# weiter durchgesetzt werden

Klassen- und Methodennamen

- Klassenname
  - Ist bis auf wenige Ausnahmen Singular
  - Sollte kein Verb sein bzw. enthalten

- Methodenname
  - Sollte ein Verb enthalten
  - Sollte klarstellen was die Methode macht
    - nicht wie sie etwas macht

Ein Wort pro Konzept

› Für ein Konzept sollte ein Name gewählt werden und dieser konsequent verwendet werden

› Für das Laden von Daten könnten die Namen

`Fetch, Get, Retrieve, Load`, etc.

verwendet werden

› Gibt es z.B. in einer Klasse die Methoden `Fetch` und `Load` ist durch den Namen nicht ersichtlich, worin sich diese unterscheiden

Kontext – Bedeutung von Bezeichnern

```
static void Main(string[] args)
{
   int dauerInMinuten;
}


class Prüfung
{
  public int DauerInMinuten { get; set; }
}
```

Was dauert eine gewisse Anzahl von Minuten?

Auswahl des Namens

- Ein Bezeichner sollte Fachbegriffe aus der Informatik verwenden
  - Z.B. Namen von Algorithmen, mathematische Begriffe, Entwurfsmuster, etc.
- Quelltext wird von Programmierern gelesen und sollte auch für diese geschrieben werden
- Alle anderen Bezeichner sollten aus dem Umfeld der zu entwickelnden Software verwendet werden

Magic Numbers

- Bei der Entwicklung sollte auf Zahlen mitten im Quelltext verzichtet werden
  - Sog. Magic Numbers

- Welche Bedeutung hat die 360?

```
int summe = x * 360;
```

- Besser
  - `const int TageImBankjahr = 360;`
  - …
  - `int summe = x * TageImBankjahr;`

Konstanten

- Für die Schreibweise von Konstanten gibt es verschiedene Ansichten
- Oft werden für Konstanten nur große Buchstaben verwendet
  - `const int TAGE_IM_BANKJAHR = 360;`

- Beim Lesen von Quelltext ziehen solche Bezeichner sehr viel Aufmerksamkeit auf sich
- In C# werden Konstanten daher häufig mit großem Anfangsbuchstaben verwendet
  - `const int TageImBankjahr = 360;`

Projekte und Projektmappen

- Bei der Benennung von Projekten und Projektmappen sollte der Name ebenfalls mit Bedacht gewählt werden
  - Der gewählte Name wird zum Namespace und ist daher ein Bezeichner im Quelltext
- Leerzeichen, Sonderzeichen und Ziffern führen erfahrungsgemäß an späterer Stelle (z.B. bei Codegenerierung) oft zu Schwierigkeiten
  - Idealerweise nur kleine und große Buchstabenstaben verwenden -> CamelCase-Schreibweise
- Außerdem sollte das Projekt niemals wie eine der Klassen im Projekt genannt werden!
  - Schlechtes Beispiel: Projekt heißt „Person“ eine Klasse in dem Projekt ebenfalls „Person“