Handout Clean Code / Code Smell

Aussagekräftige Namen

- Zweckbeschreibende Namen wählen
- Fehlinformationen vermeiden
- Unterschiede deutlich machen
- Aussprechbare Namen verwenden
- Suchbare Namen verwenden
- Codierung vermeiden
- Mentale Mappings vermeiden
- Witze und Wortspiele sparen
- Ein Wort pro Konzept
- Bedeutungsvollen Kontext verwenden
- Keinen überflüssigen Kontext hinzufügen

Funktionen

- Klein! (max. 20 Zeilen)
- Blöcke innerhalb if-, else-, while- usw. Anweisungen max. eine Zeile
- Eine Aufgabe erfüllen
- Code Top-down lesen
- Switch-Anweisungen: vermeiden. Nicht wiederholen!
- Beschreibende Namen verwenden
- Funktionsargumente (je weniger desto besser, max. 3)
- Verben und Schlüsselwörter verwenden
- Output-Argumente
- Funktionen sollten entweder etwas tun **oder** antworten
- Wiederholungen vermeiden

Kommentare

- Sind kein Ersatz für schlechten Code
- Gute Kommentare z.B. juristische oder informierende Kommentare, TODO und Warnungskommentare
- Schlechte Kommentare, z.B- Irreführende, Redundante, Tagebuchkommentare, Positionsbezeichner, Auskommentierter Code, Zu viele Infos, Unklarer Zusammenhang

Formatierung

- Vertikale Formatierung
- Vertikale Dichte
- Vertikaler Abstand
- Vertikale Anordnung
- Horizontale Formatierung
- Horizontale Offenheit und Dichte
- Horizontale Ausrichtung
- Einrückungen

Objekte und Datenstrukturen

- Datenabstraktion (bspw. durch interface)
- Law Of Demeter: Sprich nur zu Freunden und nicht zu Fremden (Die Methode sollte keine Methoden von Objekten aufrufen, die von einer der erlaubten Funktionen zurückgegeben werden / ein Modul sollte nichts über das Innere der Objekte wissen die es manipuliert)
- Zugkatastrophe

Fehler Handling

- Ausnahmen statt Rückgabecodes
- Try-Catch-Finally-Anweisungen zuerst schreiben
- Ausnahmen mit Kontext auslösen
- Den normalen Ablauf definieren
- Keine Null zurückgeben
- Keine Null übergeben

Klassen

- Klassenaufbau (Kapselung)
- Einkapselung
- Klassen sollten klein sein!
- Nicht zu viele Methoden in einer Klasse
- Single Responsibility
- Änderungen einplanen
- Änderungen isolieren

Tests

- Unzureichende Tests vermeiden
- Ein Konzept pro Test
- Coverage-Tool verwenden um Lücken in Teststrategie aufzudecken
- Triviale Tests nicht überspringen
- Grenzbedingungen testen
- Tests sollten schnell sein