CLEAN CODE!

Rules, Smells und Diskussionen

AGENDA

1. **Intro**
2. **Regeln**
3. **Smells**
4. **Wie man clean wird**
5. **Einsatz in der Praxis** (Diskussion)

WAS IST CLEAN CODE?

“THE TOTAL COST OF OWNING A MESS”

ROBERT C. MARTIN

NAMEN

- Sprechend, Aussagekräftig
- Eindeutig
- Umfassend
- Standard Fachausdrücke
- Keine Abkürzungen!!!
- Stringent einsetzen

NAMEN

```
var pageid: String!



var ppOverlayEndShown = false



// For iPad
let itemsPerRow: CGFloat = 4



func logoutUser() {
    user = nil

    TVNowUserDefaults.userObject = nil
    favoriteFormatIds.removeAll()

    #if !APPLETVAPP
    ApplicationShortcutHandler.registerDefaults(for: UIApplication.shared)
    DispatchQueue.main.async {
        let appDelegate = UIApplication.shared.delegate as? AppDelegate
        appDelegate?.updateFormatList()
    }
    #endif
    CSSubscriptionManager.shared().unregisterAllSubscriptions()
}
```

FORMATTIERUNG

- Newspaper-Metapher (von oben nach unten)
- High-Level vor Low-Level Methoden
- Zusammengehöriger Code vertikal nah beinander
- Nicht zusammengehöriger Code vertikal durch Leerzeilen getrennt
- Instanz-Variablen immer oben und nicht verteilt
- Regeln als Code-Style vom Team definiert und umgesetzt

METHODEN

- Möglichst kurz
- Verantwortlich für 1 Sache
- Gleiches Level der Abstraktion
- Einfache/kurze `try-catch` Blöcke
- Max. 3 Argumente
- Keine Flag-Argumente

METHODEN

```
func setupView() {
    nameTextField.text = profile?.name
    nameTextField.clearButtonMode = .whileEditing

    let fontSize: CGFloat = Util.isAppleTV() ? 30 : 15
    let attributedString = NSMutableAttributedString(string: "KIDS PROFIL", attributes: [
      .font: UIFont(name: "Roboto-Regular", size: fontSize)!,
      .foregroundColor: UIColor(white: 1.0, alpha: 1.0)
    ])

    checkBox.isChecked = profile?.type == .kids
    checkBox.isEnabled = profile == nil
    checkBox.isUserInteractionEnabled = profile == nil
    kidsProfileLabel.attributedText = attributedString
    kidsProfileLabel.textColor = checkBox.isEnabled ? (profile?.type == .kids ? .tiffanyBlue : .white) : .lightGray

    if let profileType = profile?.type, profileType != .kids {
        kidsProfileView.isHidden = true
    }

    titleLabel.text = profile != nil ? "Profil bearbeiten" : "Neues Profil"
    saveButton.isEnabled = profile != nil

    profileImageView.layer.cornerRadius = profileImageView.frame.width / 2
    profileImageView.image = UIImage(named: profile?.imageId ?? "profileImage02")

    deleteProfileButton.isHidden = profile?.type == .default || profile == nil
    deleteProfileButton.layer.borderWidth = 1
    deleteProfileButton.layer.borderColor = UIColor.buttonBorder.cgColor

    #if !APPLETVAPP
    let rect = CGRect(x: 0, y: 0, width: 15, height: nameTextField.frame.height)
    let paddingView = UIView(frame: rect)
    nameTextField.leftView = paddingView
    nameTextField.leftViewMode = .always
    #endif

    let attributes = [NSAttributedString.Key.foregroundColor: UIColor(white: 1, alpha: 0.7)]
    nameTextField.attributedPlaceholder = NSAttributedString(string: "Profilname",
                                                             attributes: attributes)
}
```

METHODEN

```
@objc func placeCornerLogo() {
    if player?.playbackState != .content ||
        playable.type == .live || playable.isStream {
        cornerLogoView?.removeFromSuperview()
        cornerLogoView = nil
        return
    }

    if let playersize = player?.avPlayerLayer?.videoRect.size {
        let fl = Double(round(1000 * (playersize.width / playersize.height)) / 1000)
        guard fl == 1.778 || fl == 1.333 else {
            cornerLogoView?.removeFromSuperview()
            cornerLogoView = nil
            return
        }
    }

    if let cornerLogoStation = playable.cornerLogoName {
        let ratName = (playable.is169AspectRatio()) ? "16x9" : "4x3"
        let logoName = cornerLogoStation + "_" + ratName
        let cornerLogo = UIImage(named: logoName)
        #if APPLETVAPP
        let overlayFrame = overlayPosition(playerSize: view.frame.size)
        #else
        let overlayFrame = CGRect(origin: .zero, size: cornerLogoSize)
        #endif
        let imageView = UIImageView()
        imageView.frame = overlayFrame
        imageView.image = cornerLogo
        imageView.center = view.center
        imageView.alpha = 0.0

        // Add new corner logo if cornerLogoView isn't exists or imageView.frame isn't equals to cornerLogoView.frame.
        if cornerLogoView?.frame.equalTo(imageView.frame) != true {
            cornerLogoView?.removeFromSuperview()
            cornerLogoView = imageView
            player?.insertOverlayAsSubview(overlayView: imageView)

            UIView.animate(withDuration: 0.32, animations: {
                self.cornerLogoView?.alpha = 1.0
            }, completion: nil)
        }
    } else {
        cornerLogoView?.removeFromSuperview()
    }

    …
}
```

KLASSEN

- Möglichst klein
- 1 Grund sich zu ändern
- Möglichst hohe Kohäsion
  - Wenig Instanz-Variablen
  - Methoden sollten mind. 1 Instanz-Variable nutzen
- Interne Struktur verbergen

KLASSEN

```
class BaseViewController: UIViewController, PinEntryPresentable {
    var agofTag: String = "tobedefined"
    var agofCode: String = "code"
    var startupInProgress = false
    var impressionTrackingIsAllowed = true

    @IBOutlet weak var contentScrollView: UIScrollView!

    var chosenTeaser: LegacyTeaser?
    fileprivate var progressView: UIProgressView!
    fileprivate var maxNumberOfProcesses = 0

    fileprivate var teaserSetId = 0
    var requestId: Int = 0

    weak var favouriteButtonDelegate: FavouriteButtonDelegate?
    var livestreamPreRollEnabled: Bool = true

    var resumeTimeSetByDeeplink: TimeInterval = 0

    override var preferredStatusBarStyle: UIStatusBarStyle {
        return .lightContent
    }

    var parentalControlService: ParentalControlService?
    var triggerFskPinForChromecast: Bool = false

    override func viewDidLoad() {
        super.viewDidLoad()
        leaveBreadcrumb()

        title = title != nil ? title : chosenTeaser?.navigationBarTitle
        view.backgroundColor = .infinityBlueBackground
        initProgressView()
        parentalControlService = ParentalControlService()
    }

    func checkUserValidity() -> Bool {
        if let user = UserManager.sharedInstance.getUser() {
            if user.hasValidToken() {
                return true
            }
        }
        return false
    }
    …
}
```

KOMMENTARE

GOOD

- Absicht
- Erklärung
- Warnung über Konsequenzen
- ToDo (temporär)
- “Java Docs” für public API
  (https://nshipster.com/swift-documentation)

BAD

- Redundant (keine Zusatz-Information)
- Irreführend (nicht mehr aktuell)
- Schlecht geschrieben
- Auskommentierter Code
- “Java Docs” für private API

KOMMENTARE

```
func saveProfile(name: String, callback: ((Bool, Error?) -> Void)?) {
    if let profile = profile, let id = profile.id {
        // Update profile if exist
        userProfileService?.updateProfile(profileId: id, name: name) { success, error in
            callback?(success, error)
        }
    } else {
        // Create new profile
        let type: UserProfileType = checkBox.isChecked ? .kids : .other
        userProfileService?.createProfile(name: name, type: type.rawValue) { success, error in
            callback?(success, error)
        }
    }
}
```

```
/// Applies secondary-action button style (white bordered).
func applySecondaryStyle() {
    layer.borderColor = UIColor.white.cgColor
    layer.borderWidth = 1
    backgroundColor = .clear
    setTitleColor(.white, for: .normal)
    titleLabel?.font = .buttonTitle
    setTitle(titleLabel?.text, for: .normal)
}
```

```
// By default it is true
var useHapticFeedback: Bool = true
```

UNIT TESTS

- Alles ausreichend testen
- Sauber und strukturiert (wie Production-Code)
- Clean Test == Lesbarer Test
- Wiederholbar!
- 1 Konzept pro Test
- 1 Assert pro Test

UNIT TESTS

```
class FormatGridViewControllerTests: XCTestCase {
    var sut = FormatGridViewController(dataService: FormatDataServiceMock())

    let items = [
        FormatAZItem(headline: "A", formats: [Format(id: "1"), Format(id: "5")]),
        FormatAZItem(headline: "A", formats: [Format(id: "2"), Format(id: "3")]),
        FormatAZItem(headline: "A", formats: [Format(id: "3")])
    ]

    override func setUp() {
        super.setUp()
        sut.items = items
    }

    func testControllerIsSetupCorrectly() {
        XCTAssertNotNil(sut.dataService)
    }

    func testCollectionViewShouldReturnCorrectNumbers() {
        let numbersOfSection = sut.numberOfSections(in: sut.collectionView!)
        XCTAssertEqual(items.count, numbersOfSection)
        let numbersOfItems = sut.collectionView(sut.collectionView!, numberOfItemsInSection: 1)
        XCTAssertEqual(items[1].formats.count, numbersOfItems)
    }

    func testCollectionViewShouldReturnAZTeaserCell() {
        let cell = sut.collectionView(sut.collectionView!, cellForItemAt: IndexPath(row: 1, section: 1))
        XCTAssertTrue(cell is FormatGridCell)
    }

    /*func testLoadDataShouldOnlyLoadKidsFormats() {
        sut.isKidsContent = true
        sut.loadData()
        sut.items.first?.formats.forEach {
            XCTAssertTrue($0.kids == true)
        }
    }*/

    …
}
```

SMELLS

Photo by Annie Spratt on Unsplash

CODE

- Duplizierter Code!!!
- Ungenutzter Code
- Code ist schwer zu verstehen
- Code ist unnötig komplex
- Code ist inkonsistent

- Eine Änderung führt zu einer Vielzahl nachfolgender Änderungen
- Eine Änderung führt zu Fehlern an vielen verschiedenen Stellen

IMPLEMENTIERUNG

- Zu umfangreiche Interfaces
- Zu viele öffentliche Konstanten, Methoden, Instanzvariablen
- Magic Numbers
- Nicht gekapselte Bedingungen (1)
- Negative Bedingungen (2)

(1)

```
if components.count > 3 && components[3] == "jahr" { }
```

```
if isYear(components) { }
```

(2)

```
guard !jwt.expired else { }
```

```
guard jwt.isValid else { }
guard jwt.expired == false else { }
```

Photo by Elena Rabkina on Unsplash

EMERGENT DESIGN

- Regel 1: Alles testen
- Regel 2: Kein duplizierter Code
- Regel 3: Absicht (des Programmierer) klar zum Ausdruck bringen
- Regel 4: Anzahl an Klassen und Methoden minimal halten

FORTLAUFENDE VERBESSERUNG

1. Tests für bestehende Funktionalität haben (andernfalls: schreiben)
2. Code inkrementell verbessern (Schritt für Schritt)
3. Tests sollten nicht fehlschlagen
4. Schritt 1-4 stetig im Entwicklungsprozess wiederholen

WENIGER

WTF/MINUTE

EINSATZ IN DER PRAXIS

QUELLEN

- Robert C. Martin (2009). Clean Code. Stoughton, Massachusetts, USA: Person Education Inc.
- https://gist.github.com/wojteklu/73c6914cc446146b8b533c0988cf8d29
- https://unsplash.com/photos/yCYVV8-kQNM/info (Photo by engin akyurt on Unsplash)
- https://unsplash.com/photos/aJN-jjFLyCU (Photo by Dan Gold on Unsplash)
- https://unsplash.com/photos/Bx2jFXcWsTQ (Photo by Elena Rabkina on Unsplash
- https://unsplash.com/photos/5TfCl4nj6B4 (Photo by Annie Spratt on Unsplash)
- https://unsplash.com/photos/d43jPSErVVY (Photo by Enrique Fernandez on Unsplash)