Stoffabgrenzung technische Praktiken

BBZW, Modul 426

Patrick Bucher

07.06.2024

Unittesting

1. Sie können die Vorteile von Unit Tests gegenüber manuellem Testen benennen.
2. Sie können Unit Tests für eine gegebene Funktion implementieren.
3. Sie können eine gegebene Testmethode verstehen und beurteilen.
4. Sie können die wichtigsten Assertions sinnvoll einsetzen.
5. Sie können die drei Bereiche einer Testmethode voneinander abgrenzen (Arrange, Act, Assert bzw. Given, When, Then).

Für Assertions wird an der Prüfung Dokumentation zur Verfügung gestellt.

Unterlagen

- Folien *Unit Testing mit xUnit in C#*
- Skript *Clean Agile* (Kapitel 5.1 *Test-Driven Development*)
- Repository rule-of-three
- Repository unittest-csharp
- ergänzend: Videos zum Thema Unit Tests

Simple Design

1. Sie können die Ziele von Simple Design benennen.
2. Sie können das Problem von *Design Weight* (“Gewicht des Designs”) erklären und Auswirkungen davon benennen.
3. Sie können den Zusammenhang zwischen komplexen Anforderungen und komplexem Design erklären.

Unterlagen

- Folien *Simple Design*
- Skript *Clean Agile* (Kapitel 5.3 *Simple Design*)

Pair Programming

1. Sie können die Motivation, die Ziele und den Nutzen von Pair Programming benennen.
2. Sie können die verschiedenen Ansichten zum Thema Pair Programming von Robert C. Martin (*Clean Agile*) und Kent Beck (*Extreme Programming*) wiedergeben und gegeneinander abgrenzen.
3. Sie können mögliche Probleme beim Pair Programming mit passenden Lösungen dazu benennen.

Unterlagen

- Skript *Clean Agile* (Kapitel 5.4 *Pair Programming*)
- Skript *Pair Programming (Auszug)* von Kent Beck
- Skript *Pair Programming (Auszug)* von Robert C. Martin

Clean Code

1. Sie wissen, warum Lesbarkeit und Verständlichkeit wichtige Kriterien für guten Programmcode sind.
2. Sie können die Vorteile von einheitlicher Code-Formatierung gegenüber unformatiertem Code benennen und die entsprechenden Regeln anwenden.
3. Sie verstehen die Regeln zur Benennung von Bezeichnern und können diese anwenden.
4. Sie können nützliche von unnützen Kommentaren unterscheiden und den Quellcode sinnvoll kommentieren.
5. Sie können das Prinzip der Wiederverwendbarkeit auf Code und Werte anwenden.
6. Sie können die Regeln der Klarheit von Programmcode anwenden.

Unterlagen

- Folien *Clean Code*
- Repository RedRiggedRaffle

Refactoring

1. Sie verstehen das Ziel von Refactoring und können diesen Vorgang gegenüber Neuentwicklung und Fehlerkorrektur abgrenzen.
2. Sie verstehen den engen Zusammenhang zwischen Unit Testing und Refactoring.
3. Sie wissen, was ein *Code Smell* ist, und welche Rolle es beim Refactoring spielt.
4. Sie können *Code Smells* an bestehendem Programmcode erkennen und geeignete Refactorings anwenden, um den betreffenden Code zu verbessern.
5. Sie verstehen die Vorgehensweise beim Refactoring in drei Schritten und können diese auf bestehenden Code anwenden.

Unterlagen

- Folien *Refactoring*
- Skript *Clean Agile* (Kapitel 5.2 *Refactoring*)
- Repository refactoring