Unit Testing (Jednotkové testování)

Unit Testing je základní úrovní testování softwaru. Cílem je izolovat každou část programu a ukázat, že tyto jednotlivé části (jednotky) fungují přesně tak, jak mají. Testy píší sami vývojáři souběžně s psaním samotného kódu.

Jak Unit Test vypadá?

Jednotkový test je malý skript, který automaticky provede následující kroky (známé jako vzor **AAA**):

```
1. **Arrange (Příprava):** Nastaví se výchozí podmínky a připraví se
objekty.
2. **Act (Akce):** Zavolá se testovaná metoda s konkrétními parametry.
3. **Assert (Ověření):** Porovná se skutečný výsledek s tím očekávaným.
Pokud se shodují, test prošel.
```

Příklad (Pseudokód):

```
@Test
void testScteni() {
    // Arrange
    Kalkulacka kalk = new Kalkulacka();
    // Act
    int vysledek = kalk.secti(2, 3);
    // Assert
    assertEquals(5, vysledek);
}
```

Klíčové vlastnosti Unit Testů

Aby byly testy užitečné, musí splňovat pravidla **FIRST**:

- **F – Fast (Rychlé):** Musí běžet v řádu milisekund. Vývojář jich spouští stovky několikrát denně.
- **I – Independent (Nezávislé):** Jeden test nesmí záviset na výsledku jiného.
- **R – Repeatable (Opakovatelné):** Test musí dopadnout stejně při každém spuštění (nezávisle na čase nebo prostředí).
- **S – Self-validating (Samo-ověřující):** Test má jasný výsledek (prošel/neprošel), nikdo ho nemusí ručně zkoumat.
- **T – Timely (Včasné):** Testy se píší ideálně těsně před kódem nebo spolu s ním.

Proč jsou Unit Testy důležité?

- **Bezpečné refaktorování:** Pokud změníte vnitřní strukturu kódu, testy vám okamžitě řeknou, zda jste něco nerozbili.
- **Dokumentace:** Testy slouží jako příklady použití – ukazují ostatním vývojářům, co která metoda dělá.
- **Snížení technického dluhu:** Chyby nalezené v rané fázi jsou mnohem levnější na opravu než chyby nalezené v produkci.
- **Design kódu:** Pokud se kód špatně testuje, je to obvykle známka špatného návrhu (např. porušení SOLID principů).

Izolace: Mockování a Stubování

Aby byl test skutečně „jednotkový“, nesmí sahat do databáze nebo na internet. Pokud testovaná metoda vyžaduje externí službu, nahradí se tzv. **Mockem** (falešným objektem). * *Příklad:* Místo skutečného odeslání e-mailu použijeme „MockEmailSender“, který se jen tváří, že e-mail odeslal, a my si v testu ověříme, zda byl zavolán.

Populární nástroje

Jazyk	Nástroj (Framework)
Java	JUnit, TestNG
C# (.NET)	xUnit, NUnit
Python	pytest, unittest
JavaScript	Jest, Mocha

Související pojmy: Refaktorování, Technický dluh, SOLID principy, TDD (Test Driven Development), Integrační testování.