Mock (Testovací náhrada)

Mockování je klíčovou součástí jednotkového testování. Umožňuje izolovat testovaný kus kódu od jeho okolí, jako jsou databáze, externí API nebo souborový systém.

1. Proč používat Mocky?

Existuje několik hlavních důvodů, proč místo skutečného objektu použít mock:

- **Rychlost:** Skutečný dotaz do databáze nebo na internet trvá dlouho. Mock v paměti odpoví okamžitě.
- **Předvídatelnost:** Mock se chová přesně tak, jak mu řeknete. Můžete nasimulovat i těžko reprodukovatelné situace (např. výpadek serveru).
- **Izolace:** Pokud test selže, víte přesně, že je chyba v testovaném kódu, a ne v externí službě, která má zrovna výpadek.
- **Bezpečnost:** Nechcete, aby testovací skript při každém spuštění posílal skutečné e-maily klientům nebo strhával peníze z kreditních karet.

2. Rozdíl: Mock vs. Stub

Tyto pojmy se často pletou, ale mají rozdílný význam:

- **Stub:** Pasivní náhrada. Pouze vrací předem definovaná data (např. „Vždy vrať číslo 10“).
- **Mock:** Aktivní náhrada. Navíc si pamatuje, jak s ním bylo zacházeno. Můžete se ho zeptat: „Byla funkce 'odeslat_platbu' zavolána právě jednou a se správnými parametry?“.

3. Jak vypadá Mock v praxi?

Představte si funkci, která zjišťuje počasí a podle toho doporučí oblečení. Místo toho, aby se funkce skutečně připojovala k serveru s počasím, podstrčíte jí mock:

```
// Definice mocku: "Kdykoliv se tě někdo zeptá na teplotu, řekni, že je
-10°C"
weatherMock.setup(temp => -10);

// Spuštění testu
result = doporuc_oblečení(weatherMock);

// Ověření (Assertion)
assertEquals("Vezmi si zimní bundu", result);
```

4. Mockování externích služeb (API)

Při vývoji mikroslužeb se často používají nástroje pro mockování celých HTTP serverů. Programátor pak nemusí mít spuštěno všech 20 služeb, stačí mu jejich mocky, které simulují správné odpovědi.

5. Úskalí: Příliš mnoho mockování

Pokud zamockujete úplně všechno, může se stát, že vaše testy sice projdou („v izolaci vše funguje“), ale po nasazení aplikace se zhroutí, protože skutečné komponenty spolu nespolupracují tak, jak jste předpokládali. Proto je nutné doplňovat unit testy také **integračními testy**.

> **Zajímavost:** Existují pokročilé knihovny (např. Mockito pro Javu, Jest pro JavaScript nebo unittest.mock pro Python), které dokáží automaticky vygenerovat mock objekt na základě existující třídy nebo rozhraní.

Zpět na Algoritmy