Code Academy

Dėstytojas

Edvinas Kesminas

Unit Testing

Data

Šiandien išmoksite

01 Unit Testing

02 CICD Unit Test Sample

Unit Tests

- Unit testing yra programinės įrangos kūrimo praktika, kurioje atskirai testuojami mažiausi programos komponentai, vadinami vienetais arba moduliais.
- Šie vienetai yra tikrinami, ar jie veikia pagal programuotojo numatytą elgesį ir ar teisingai reaguoja į įvestį.

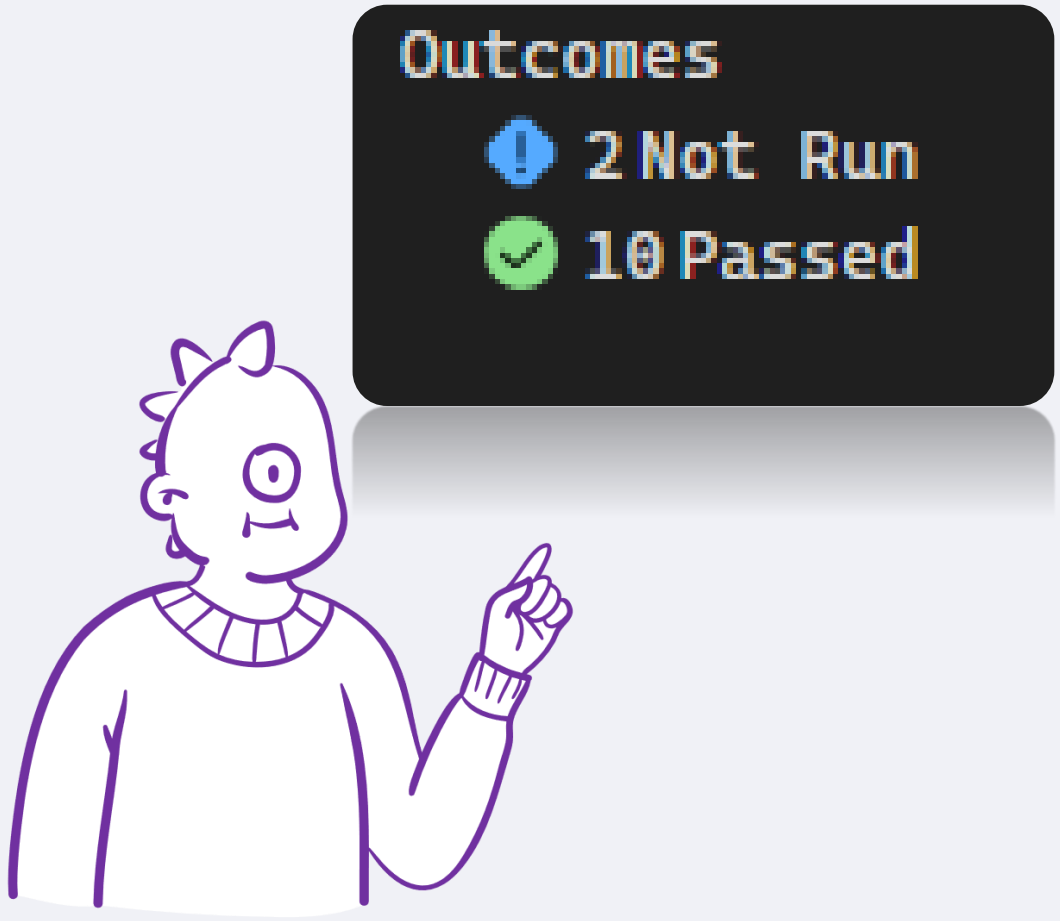
```
public static int GetTextLength(string text, bool includeLeadSpace = false)
{
    if (!includeLeadSpace)
    {
        return text.Trim().Length;
    }
    else
    {
        return text.Length;
    }
}
```

...inis testavimas padeda užtikrinti
...ramos kokybę. Testuojant atskirus
...ar
...arba
...ulis

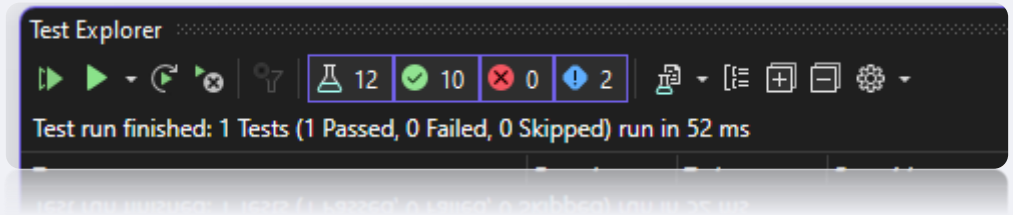
```
[TestMethod()]
| 0 references
public void GetTextLength_LeadWhiteSpacedText_ReturnsLength()
{
    // Arrange
    string fakeName = " Hello ";
    //int expected = 7;
    int expected = 5;

    // Act
    int actual = Program.GetTextLength(fakeName);

    // Assert
    Assert.AreEqual(expected, actual);
}
```

Unit Tests

- Vienetinis testavimas padeda supaprastinti klaidų paiešką ir taisymą.
- Kiekvienas vienetas yra testuojamas atskirai, todėl klaidas lengviau lokalizuoti ir ištaisyti.
- Kai tik randama klaida, galima nukreipti pastangas tik į tą konkretų vienetą, o ne į visą sistemą.
- Tai sutaupo laiko ir resursų, padeda išvengti nereikalingo "debuginimo" ir pagerina produktyvumą.



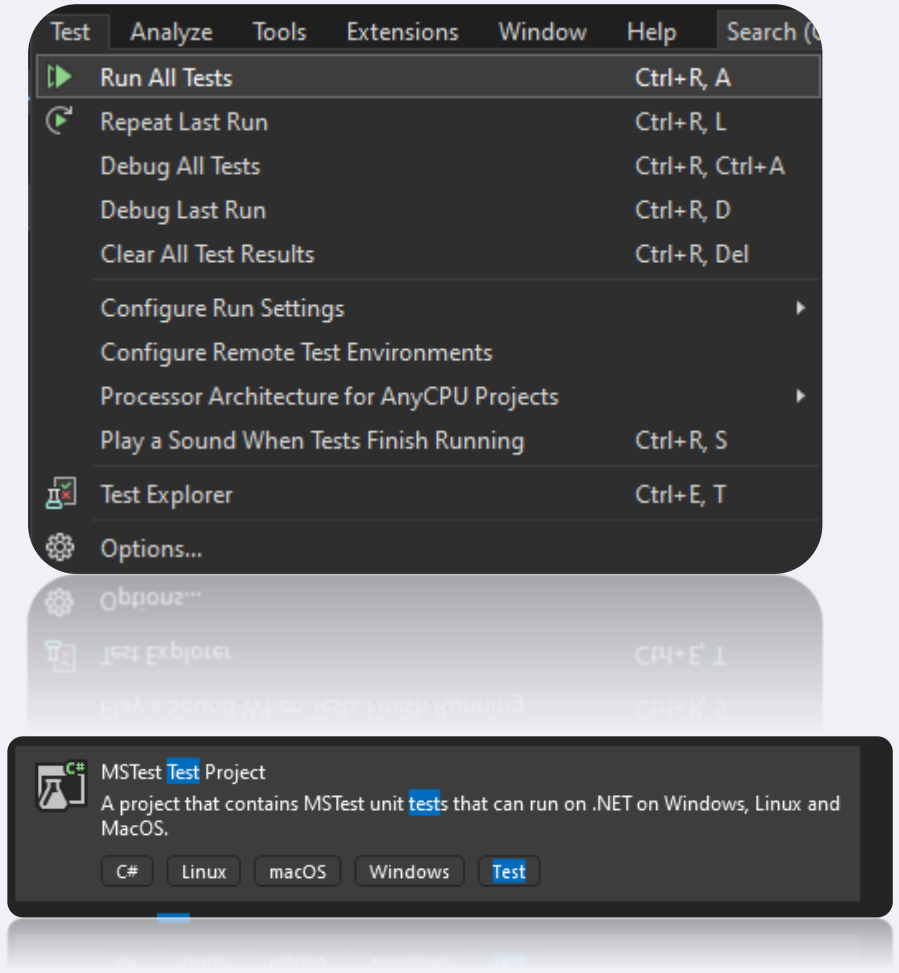
Unit Tests



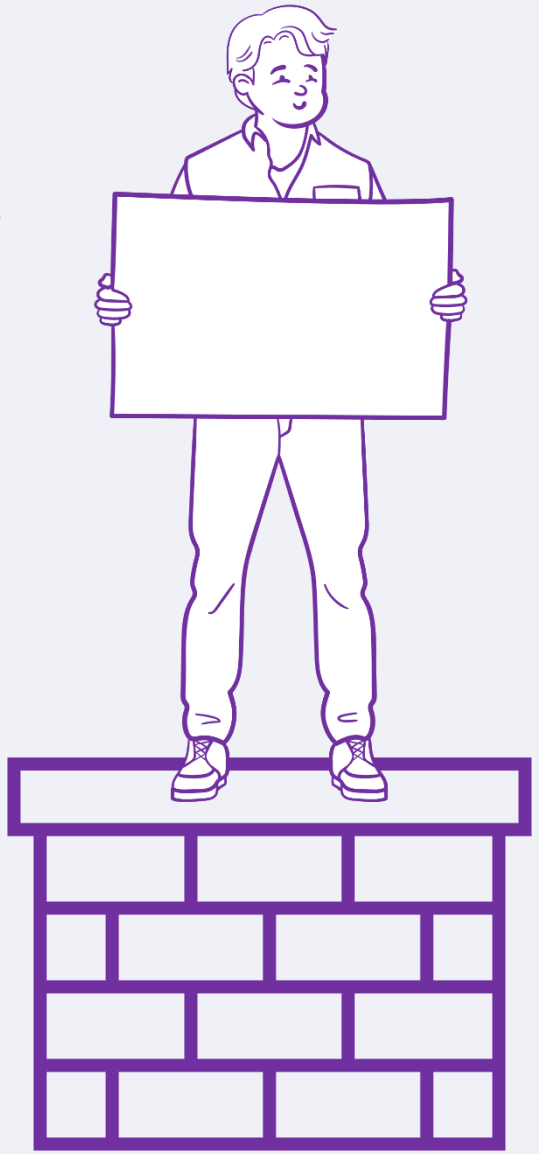
- Vienetinis testavimas leidžia užtikrinti, kad jau parašytas kodas veikia net po pakeitimų. Įdiegus vienetinį testavimą, bet koks naujas kodas yra testuojamas prieš integraciją į didesnę sistemą.
- Tai padeda užtikrinti, kad naujos funkcijos ar pakeitimai nekenkia jau egzistuojančiam kodui ir nesukelia klaidų.

Unit Tests

- Vienetinio testavimo scenarijai gali būti lengvai kartojami. Vienetinių testų scenarijai yra automatizuoti, todėl juos galima lengvai paleisti ir kartoti daug kartų.
- Tai yra naudinga atliekant pakartotinį testavimą po kodinių pakeitimų arba norint įsitikinti, kad funkcionalumas išlieka stabilus.



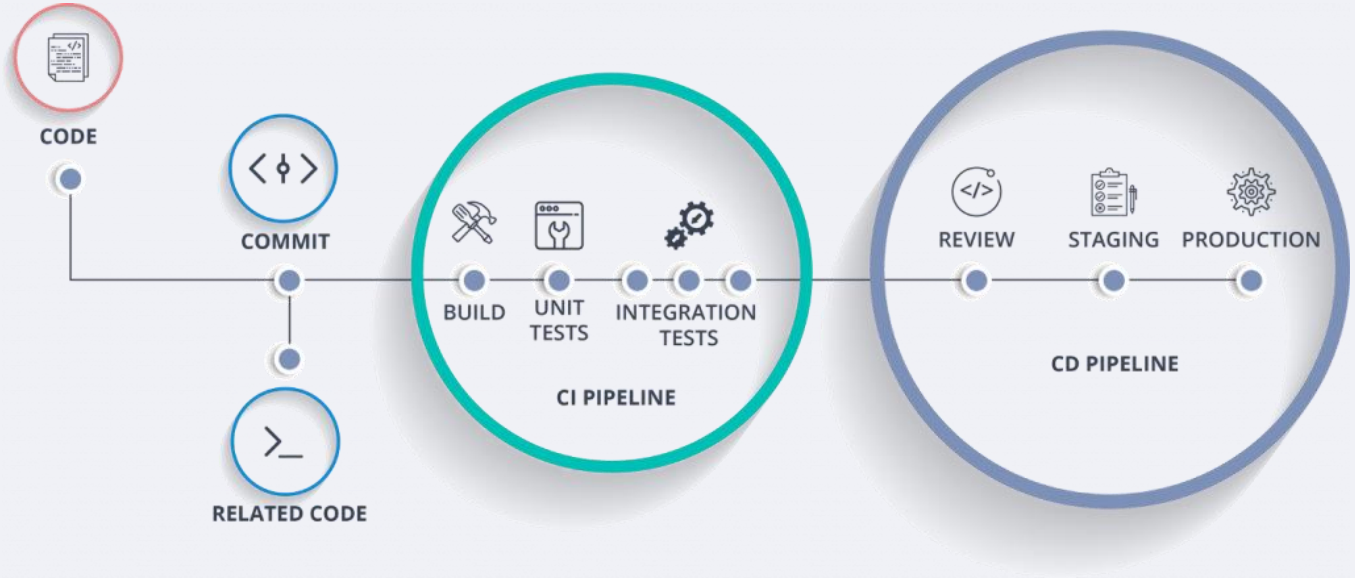
Unit Tests



- Vienetinio testavimo procesas prisideda prie geresnės kodavimo praktikos.
- Norint testuoti vienetus, kodas turi būti parašytas atskiram testavimui skirtame būsenoje.
- Tai skatina rašyti aiškų ir išsamų kodą, gerą kodavimo stilių ir modularumo principų laikymąsi.
- Be to, tai skatina programuotojus mąstyti apie kodo projektavimą ir suderinamumą su kitais moduliais.

CICD (Continuous Integration and Continuous Delivery)

- CICD yra procesas, kuriuo siekiama automatizuoti programinės įrangos diegimą, testavimą ir pristatymą, užtikrinant greitą ir stabilų produkto vystymą. Tai apima nuolatinį kodavimo, testavimo ir pristatymo procesą, kuriame kiekviena kodo pakeitimo versija yra automatiškai testuojama ir, jei testai sėkmingai įvykdomi, taip pat automatiškai pristatoma į produkcinę aplinką.



Užduotis nr. 1

- Sukurkite skaičiuotuvo programą. Jūsų programa turėtų mokėti atspausdinti į ekraną meniu parinktis ir turi veikti iki kol būna įvedama ‚q' raidė. Skaičiuotuvas turėtų mokėti +,- ,*,/,pakelti laipsniu ir ištraukti šaknį. Kiekvienam iš metodų parašykite po unit testą.

- Sukurkite 3 pamokos (If)užduočių metodus ir kiekvienam iš jų sukurkite po 2 testus.
- Sukurkite 4 pamokos (Switch)užduočių metodus ir kiekvienam iš jų sukurkite po 2 testus.
- Sukurkite 5 pamokos (String manipulation) užduočių metodus ir kiekvienam iš jų sukurkite po 2 testus.
- Sukurkite 6 pamokos (While) užduočių metodus ir kiekvienam iš jų sukurkite po 2 testus.

  Sukurkite 7 pamokos (Methods) parašytų metodų unit testus.

Projektas nr. 1

- Savo 3 pamokos projekto funkcionalumą išjudinkite į atskirus metodus. Kurdami metodus bandykite stipriai akcentuoti testuojamą kodą (Turėtų kažkas grįžti iš pačio metodo). Parašę metodus juos visus padenkite testais. Kiekvienas scenarijus turėtų turėti nors 1 testą.

Nuorodos

https://learn.microsoft.com/en-us/visualstudio/test/walkthrough-creating-and-running-unit-tests-for-managed-code?view=vs-2022

https://learn.microsoft.com/en-us/dotnet/core/testing/unit-testing-best-practices

https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/nullable-value-types

https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/operators/null-forgiving