



# Programų testavimas ir taisymas

# Testavimo tipai

- Vienetinis testavimas (Unit Testing)
- Integracinis testavimas (Integration Testing)
- Funkcinis testavimas (Functional Testing)
- Regresinis testavimas (Regression Testing)
- Našumo testavimas (Performance Testing)
- Krovininis testavimas (Load Testing)
- Naudotojo patirties testavimas (User Acceptance Testing, UAT)
- Saugumo testavimas (Security Testing)
- Automatizuotas testavimas (Automated Testing)

# Vienetinis testavimas (Unit Testing)

**Tikslas:** Testuoja atskiras programos dalis (vienetus), paprastai funkcijas ar metodus.

**Naudojimas:** Ankstyvoje kūrimo stadijoje, kad patikrintų, ar kiekviena dalis veikia tinkamai.

**Pavyzdys:** Testuoti, ar funkcija, skaičiuojanti kvadratinę šaknį, grąžina teisingą rezultatą.

# Integracinis testavimas (Integration Testing)

**Tikslas:** Patikrina, kaip skirtingos sistemos dalys veikia kartu.

**Naudojimas:** Įsitikinti, kad duomenų perdavimas tarp modulių yra tinkamas.

**Pavyzdys:** Testuoti sąsają tarp vartotojo registracijos ir duomenų bazės įrašų saugojimo.

# Funkcinis testavimas (Functional Testing)

**Tikslas:** Patikrina, ar programa atlieka tai, ką ji turėtų pagal reikalavimus.

**Naudojimas:** Atliekamas simuliuojant realius vartotojų veiksmus.

**Pavyzdys:** Patikrinti, ar prisijungimo funkcija leidžia prieigą tik su teisingais duomenimis.

<https://www.zaptest.com/lt/kas-yra-funkcinis-testavimas-tipai-pavyzdziai-kontrolinis-sarasas-ir-igyvendinimas>

# Regresinis testavimas (Regression Testing)

**Tikslas:** Užtikrina, kad nauji pakeitimai nepažeistų anksčiau veikusių funkcijų.

**Naudojimas:** Po kodo pakeitimų ar atnaujinimų.

**Pavyzdys:** Patikrinti, ar nauja funkcija (pvz., vartotojų profiliai) netrikdo kitų esamų funkcijų.

# Našumo testavimas (Performance Testing)

**Tikslas:** Įvertina, kaip sistema veikia pagal apkrovą, greitį ir stabilumą.

**Naudojimas:** Įsitikinti, kad programa gali dirbti efektyviai esant dideliame vartotojų skaičiui.

**Pavyzdys:** Patikrinti, kiek vartotojų vienu metu gali naudoti internetinę parduotuvę.

# Krovininis testavimas (Load Testing)

**Tikslas:** Nustato sistemos ribas, priverčiant ją veikti maksimaliai apkrauta.

**Naudojimas:** Analizuoti, kaip sistema susidoroja su ekstremaliomis sąlygomis.

**Pavyzdys:** Testuoti, kaip svetainė veikia per didžiausio lankomumo laiką



# Naudotojo patirties testavimas (User Acceptance Testing, UAT)

**Tikslas:** Patikrinti, ar programinė įranga atitinka galutinių vartotojų poreikius.

**Naudojimas:** Prieš paleidžiant produktą į rinką.

**Pavyzdys:** Leisti vartotojams išbandyti programą ir gauti jų atsiliepimus

# Saugumo testavimas (Security Testing)

**Tikslas:** Aptikti sistemos pažeidžiamumus ir apsaugoti ją nuo įsilaužimų.

**Naudojimas:** Užtikrinti, kad jautrūs duomenys yra apsaugoti.

**Pavyzdys:** Testuoti, ar slaptažodžiai yra tinkamai užšifruoti

# Automatizuotas testavimas (Automated Testing)

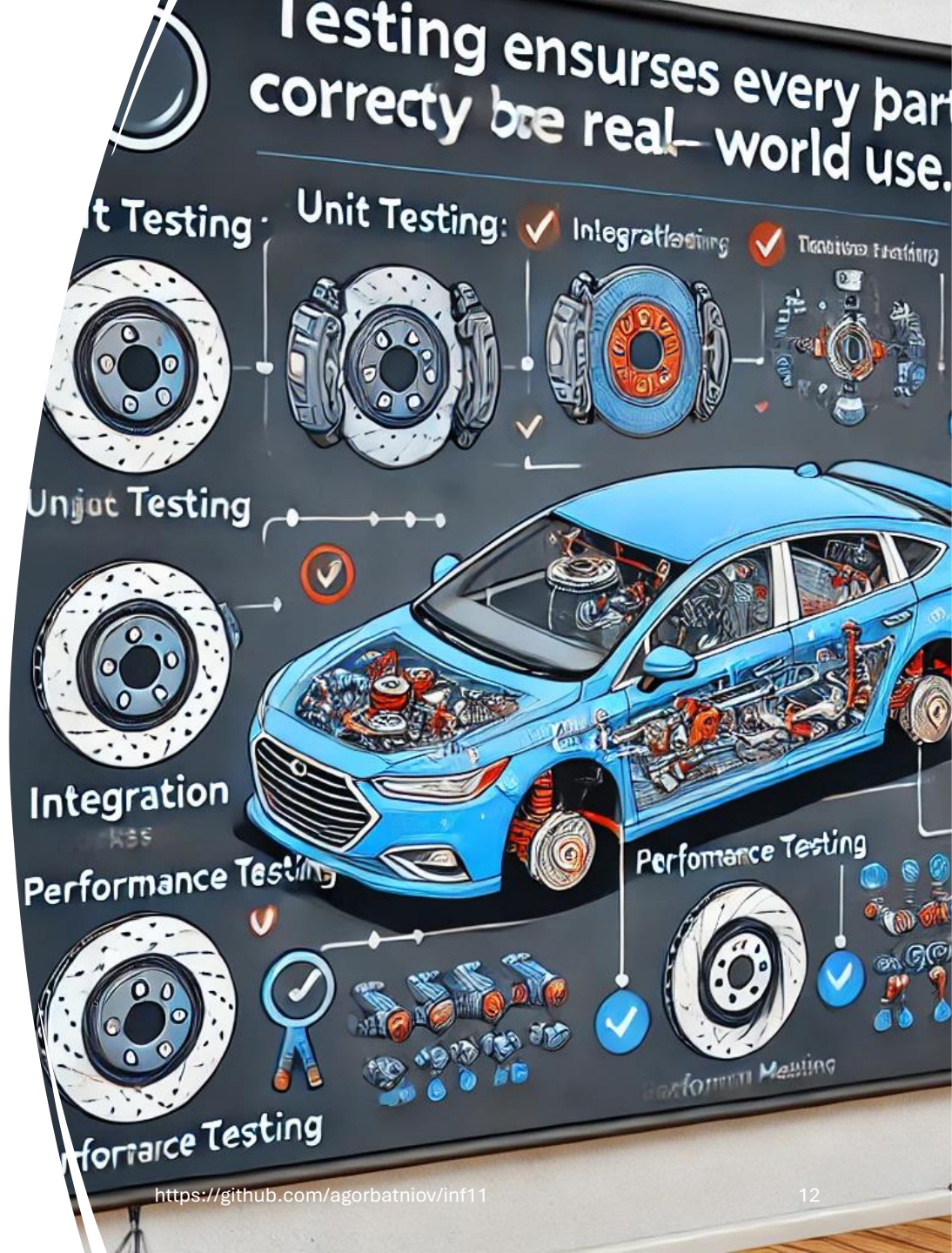
**Tikslas:** Naudoja testavimo įrankius automatizuotam kodo tikrinimui.

**Naudojimas:** Taupo laiką, kai testai dažnai kartojami.

**Pavyzdys:** Naudojant Selenium, testuoti vartotojo sąsajas

# Automobilio stabdžių sistemos testavimu

Testavimo procesai automobilių pramonėje užtikrina, kad stabdžiai veiktų patikimai ir saugiai realiame pasaulyje. Panašiai programų testavimas užtikrina, kad programinė įranga veiktų be klaidų ir atitiktų reikalavimus.



# Automobilio stabdžių sistemos testavimu

## 1. Vienetinis testavimas (Unit Testing):

- Testuojama kiekviena stabdžių sistemos dalis atskirai, pvz., stabdžių diskai, kaladėlės ar hidraulinis siurblys.

## 2. Integracinis testavimas (Integration Testing):

- Tikrinama, kaip veikia stabdžių sistemos dalys kartu – ar stabdžių kaladėlės susiliečia su diskais tinkamu momentu, ar veikia hidraulinis slėgis.

## 3. Našumo testavimas (Performance Testing):

- Atliekamas stabdžių veikimo testavimas realiomis sąlygomis, pvz., ekstremalaus stabdymo metu ar važiuojant šlapia kelio danga.

## 4. Regresinis testavimas (Regression Testing):

- Patikrinama, ar po pakeitimų (pvz., pakeitus stabdžių kaladėles) sistema veikia taip pat patikimai, kaip anksčiau.

# Vienetinio testavimo sąvoka (Unit Testing)

Vienetinis testavimas (*Unit Testing*) – tai procesas, kai testuojamos atskiros programos dalys (pvz., funkcijos, metodai ar moduliai), siekiant užtikrinti jų teisingą veikimą.

**Tikslas:** Anksti aptikti klaidas. Užtikrinti kiekvieno komponento patikimumą prieš juos integruojant.

**Nauda:** Lengviau lokalizuoti problemas.  
Supaprastina kodo priežiūrą ir atnaujinimus.

**Pavyzdys:** Funkcija, kuri grąžina dviejų skaičių sumą, testuojama su skirtingomis įvestimis: teigiamais, neigiamais ir nuliniais skaičiais.

# Vienetinio testavimo scenarijus

Funkcija `add` grąžina dviejų skaičių sumą.

`assert` naudojamas testuoti rezultatą.

Jei rezultatas neteisingas, išmetama klaida.

## Rezultatas:

Jei visi testai praeina, programa tęsia darbą.

Jei testas nepraeina, parodoma klaida, nurodanti neteisingą funkcijos elgesį.

```
def add(a, b):  
    return a + b  
  
# Testavimo scenarijai  
# Teisingas atvejis  
assert add(2, 3) == 5  
# Neigiamas ir teigiamas  
assert add(-1, 1) == 0  
# Nulis kaip įvestis  
assert add(0, 0) == 0  
  
print("Visi testai sėkmingai")
```