

Pamulang University

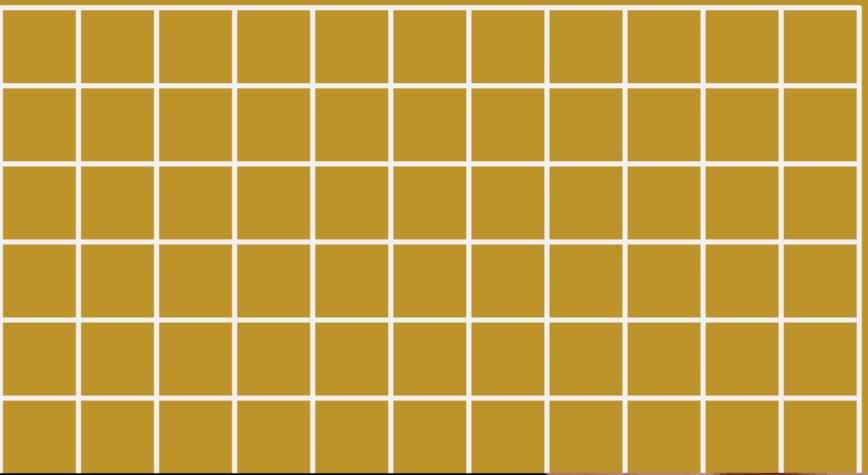
WHITEBOX DAN UNIT TEST SERTA CI/CD PADA PROJECT PYTHON

Presentation by

201011401630 - NELSONIUS OLA AMAN

Overview

- Pengenalan Whitebox Testing
- Unit Test
- Implementasi Whitebox Testing
- Implementasi Unit Test
- Pengenalan CI/CD
- Contoh Konfigurasi
- Kesimpulan
- Referensi/Sumber Materi



Pengenalan Whitebox Testing

Whitebox testing, juga dikenal sebagai pengujian struktur, adalah jenis pengujian perangkat lunak yang dilakukan dengan memeriksa struktur internal dari kode sumber.

Tujuan utamanya adalah untuk memahami dan memeriksa jalur eksekusi kode secara detail. Dalam whitebox testing, pengujian dilakukan dengan mempertimbangkan logika internal dari program dan bagaimana kode sumber dieksekusi.

Proses whitebox testing melibatkan analisis kode sumber, termasuk mengidentifikasi dan menguji semua jalur eksekusi yang mungkin dalam program.

Whitebox testing berguna untuk menemukan kesalahan logika, masalah aliran kontrol, atau ketidaksesuaian dengan spesifikasi desain yang mungkin tidak terdeteksi dalam pengujian blackbox, di mana hanya fungsi dan masukan yang diperiksa tanpa memperhatikan detail implementasi.

Pada umumnya, pengembang perangkat lunak atau insinyur QA (Quality Assurance) yang bekerja dengan akses langsung ke kode sumber biasanya melakukan whitebox testing.

Teknik umum yang digunakan dalam whitebox testing



1 Statement Coverage

Memeriksa apakah setiap pernyataan dalam kode dieksekusi



2 Branch Coverage

Memeriksa apakah setiap cabang (pernyataan kondisional) dalam kode dieksekusi.



3 Path Coverage

Memeriksa semua kemungkinan jalur eksekusi dalam kode



Implementasi Whitebox

Implementasi Whitebox testing membantu memastikan bahwa kode perangkat lunak berfungsi dengan benar sesuai dengan logika internalnya.

Ini juga membantu dalam menemukan kesalahan logika, masalah aliran kontrol, atau ketidaksesuaian dengan spesifikasi desain yang mungkin tidak terdeteksi dalam pengujian blackbox.

Selain itu, code coverage digunakan untuk memantau sejauh mana kode telah diuji, memastikan cakupan pengujian yang baik, dan meningkatkan kualitas perangkat lunak.

```
: > Users > Amar Rama.DESKTOP-MR0RO2R > Documents > Nelsonius Ola Aman_2.py
1 def divide(x, y):
2     if y == 0:
3         raise ValueError("Cannot divide by zero")
4     return x / y
5
6 import pytest
7
8 def test_divide_positive_numbers():
9     result = divide(10, 2)
10    assert result == 5
11
12 def test_divide_by_zero_raises_exception():
13    with pytest.raises(ValueError):
14        divide(10, 0)
15
16 if __name__ == '__main__':
17    pytest.main()
```

Unit Test

Unit test adalah jenis pengujian perangkat lunak yang digunakan untuk menguji komponen terkecil dari kode sumber, seperti fungsi, metode, atau kelas secara terisolasi.

Tujuannya adalah untuk memastikan bahwa setiap unit kode bekerja dengan benar sesuai dengan spesifikasi fungsional yang telah ditentukan.

Unit test fokus pada menguji unit kode secara individu tanpa memperhatikan interaksi dengan komponen lain dalam sistem.

Karakteristik penting dari unit test

Terisolasi

Unit test mengisolasi komponen yang diuji sehingga hanya satu unit kode yang diuji dalam setiap tes. Hal ini memungkinkan pengujian yang efektif terhadap fungsionalitas unit tersebut.

Berulang

Unit test harus dapat dijalankan berulang kali dan menghasilkan hasil yang konsisten. Ini memungkinkan untuk mendekripsi perubahan dalam perilaku unit kode seiring waktu.

Otomatisasi

Unit test seringkali diotomatisasi, sehingga dapat dijalankan secara berkala dan dengan cepat, menghemat waktu dan sumber daya dalam pengujian.

Menggunakan Assertion

Dalam unit test, assertion (pernyataan perbandingan) digunakan untuk memeriksa hasil operasi unit terhadap ekspektasi yang telah ditentukan. Jika hasilnya sesuai dengan ekspektasi, tes dianggap berhasil; jika tidak, tes gagal.

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The top navigation bar includes 'Welcome', 'Untitled', 'Unitest.py X', 'whitebox.py', and 'settings.json'. The left sidebar shows a file tree with 'Unitest.py' selected. The main editor area contains Python test code:

```
1 import unittest
2
3 class TestStringMethods(unittest.TestCase):
4
5     def test_upper(self):
6         self.assertEqual('foo'.upper(), 'FOO')
7
8     def test_isupper(self):
9         self.assertTrue('FOO'.isupper())
10        self.assertFalse('Foo'.isupper())
11
12     def test_split(self):
13         s = 'hello world'
14         self.assertEqual(s.split(), ['hello', 'world'])
15         # check that s.split fails when the separator is not a string
16         with self.assertRaises(TypeError):
17             s.split(2)
18
19 if __name__ == '__main__':
20     unittest.main()
```

Below the editor are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected), and 'PORTS'. The terminal window displays PowerShell commands and output:

```
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\Users\Amar Rama\Desktop-MR0R02R\Downloads> & 'C:\Users\Amar Rama\Desktop-MR0R02R\AppData\Local\Microsoft\WindowsApps\python3.11.exe' 'c:\Users\Amar Rama\Desktop-MR0R02R\.vscode\extensions\ms-python.python-2023.20.0\pythonFiles\lib\python\debugpy\adapters\...\...\debugpy\launcher' '64445' '--' 'C:\Users\Amar Rama\Desktop-MR0R02R\Downloads\Unitest.py'
...
Ran 3 tests in 0.003s
OK
PS C:\Users\Amar Rama\Desktop-MR0R02R\Downloads>
```

The status bar at the bottom shows 'Ln 19, Col 20' and 'Python 3.11.6 64-bit (Microsoft Store)'.

Implementasi Unit Testing

Implementasi unit testing membantu memastikan bahwa setiap unit kode berfungsi dengan benar dan tidak memengaruhi unit lain.

Mendukung prinsip pengembangan perangkat lunak yang baik seperti modularitas, pemeliharaan kode, dan perbaikan yang lebih mudah.

Unit testing juga membantu dalam menemukan kesalahan atau masalah pada tahap awal pengembangan, yang dapat mengurangi biaya dan waktu yang diperlukan untuk perbaikan di masa mendatang.

Continuous Integration (CI) dan Continuous Deployment (CD)



Continuous Integration (CI) dan Continuous Deployment (CD) adalah praktik dalam pengembangan perangkat lunak yang dirancang untuk meningkatkan efisiensi, kualitas, dan kecepatan pengiriman perangkat lunak.

CI/CD adalah praktik kunci dalam DevOps yang memungkinkan pengembangan perangkat lunak yang lebih cepat, responsif, dan handal.

Praktik ini membantu dalam mengintegrasikan perubahan kode lebih cepat, menguji secara otomatis, mengotomatiskan pengiriman perangkat lunak, dan meningkatkan efisiensi tim pengembangan.

Continuous Integration (CI)

CI adalah praktik di mana pengembang secara teratur menggabungkan kode yang mereka tulis ke dalam repositori bersama. Setiap kali perubahan kode dimasukkan, sistem otomatis akan menjalankan serangkaian tes (unit test, integrasi test, dll.) untuk memeriksa apakah perubahan tersebut tidak merusak fungsionalitas yang ada. Praktik CI membantu dalam mendeteksi dan memperbaiki masalah sejak dini, menghindari konflik integrasi besar, dan memungkinkan pengembang untuk bekerja dalam lingkungan yang lebih stabil.

Continuous Deployment (CD)

CD adalah ekstensi dari CI yang mengotomatiskan proses pengiriman perangkat lunak ke lingkungan produksi setelah berhasil melewati tahap CI. Dalam model CD, perangkat lunak diuji, dikemas, dan diterapkan ke produksi dengan otomatis. CD memungkinkan pengiriman perangkat lunak yang lebih cepat dan lebih terkontrol, serta membantu mengurangi risiko kesalahan manusia dalam proses pengiriman.

Langkah –Langkah Konfigurasi Project Python

Type  to search

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

Owner *  NelsonBahy Repository name *  testing-QA is available.

Great repository names are short and memorable. Need inspiration? How about [literate-system](#) ?

Description (optional)

 Public Anyone on the internet can see this repository. You choose who can commit.

 Private You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file This is where you can write a long description for your project. [Learn more about READMEs](#).

Code Issues Pull requests Actions Projects Wiki Security Insights

Get started with GitHub Actions

Build, test, and deploy your code. Make code reviews, branch management, and issue triaging work together.

Skip this and [set up a workflow yourself](#) →

Search workflows

Suggested for this repository

 Simple workflow By GitHub Start with a file with the minimum necessary structure.

Configure

Deployment

Deploy Node.js to Azure  Deploy to Amazon ECS  Build

NelsonBabay / testing-QA

Type to search | [Issues](#) | [Pull requests](#) | [Actions](#) | [Projects](#) | [Wiki](#) | [Security](#) | [Insights](#) | [Settings](#)

testing-QA Public

Set up GitHub Copilot
Use GitHub's AI pair programmer to autocomplete suggestions as you code.
[Get started with GitHub Copilot](#)

Add collaborators to this repository
Search for people using their GitHub username or email address.
[Invite collaborators](#)

Quick setup — if you've done this kind of thing before

[Set up in Desktop](#) or [HTTPS](#) [SSH](#) git@github.com:NelsonBabay/testing-QA.git

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

Page 12

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

testing-QA /

Drag files here to add them to your repository
Or [choose your files](#)

Commit changes

Add files via upload

Add an optional extended description...

Kesimpulan

Whitebox testing adalah pendekatan yang memeriksa struktur internal kode sumber untuk memahami dan memverifikasi fungsionalitasnya secara rinci.

Unit test adalah bagian dari whitebox testing yang fokus pada pengujian komponen terkecil dari kode sumber, seperti fungsi atau metode, untuk memastikan bahwa setiap unit bekerja dengan benar.

CI/CD adalah praktik yang mengotomatisasi pengujian dan pengiriman perangkat lunak, memungkinkan pengembangan yang lebih cepat dan lebih terkendali

Dengan menerapkan unit test dan CI/CD dalam proyek Python, Anda dapat meningkatkan kualitas perangkat lunak, meningkatkan efisiensi pengembangan, dan merilis perangkat lunak dengan kecepatan dan kepercayaan yang lebih besar.

Reference

- <https://docs.python.org/3/library/unittest.html>
- <https://www.dicoding.com/blog/white-boxtesting/>
- [https://www.youtube.com/watch?v=PsO5dZqBck Y](https://www.youtube.com/watch?v=PsO5dZqBckY)

TERIMAKASIH

