MAKALAH
Usefulness and Effectiveness of Test-First Development and Junit Framework

KELOMPOK SQUAD

1	NASRULLAH KHOMAENI	5200411208
2	CLAUDIO ORLANDO DE ARAUJO	5200411214
3	RIDHO ICHVAN	5200411256

UNIVERSITAS TEKNOLOGI YOGYAKARTA
PROGRAM STUDI TEKNIK INFORMATIKA

TAHUN 2021/2022

KATA PENGANTAR

Puji syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa. Atas rahmat dan hidayah-Nya, penulis bisa menyelesaikan makalah yang berjudul “JUNIT."

Makalah ini berisi tentang pengetahuan akan Junit. Penulis menyadari ada kekurangan pada Makalah ini. Oleh sebab itu, saran dan kritik senantiasa diharapkan demi perbaikan karya penulis. Penulis juga berharap semoga karya ilmiah ini mampu memberikan pengetahuan tentang Junit Dalam Pengembangan Aplikasi Berbasis Web

.

DAFTAR ISI

BAB I
PENDAHULUAN

A. Latar Belakang

Pada saat ini perkembangan teknologi perangkat lunak telah berkembang pesat dan menjadi pendukung utama bagi sebuah perusahaan. Suatu perusahaan atau lembaga yang menempatkan teknologi perangkat lunak menjadi salah satu pendukung dalam kemajuan perusahaan dapat mencapai rencana strategis organisasi. Perangkat lunak (*software*) sendiri merupakan program komputer yang terasosiasi dengan dokumentasi perangkat lunak seperti dokumentasi kebutuhan, model desain, dan cara penggunaannya (*user manual*). Perangkat lunak pada saat ini sudah menjadi kebutuhan umum di setiap usaha karena dengan memanfaatkan teknologi perangkat lunak akan membantu suatu perusahaan untuk memecahkan sebuah permasalahan yang terjadi.

JUnit adalah simple unit testing framework untuk menulis repeatable tes di Java**.** JUnit telah menjadi sangat penting dalam pengembangan test-driven development dan merupakan salah satu kerangka pengujian standar untuk Java developers. TestNG – Java, Open Source

A. Rumusan Masalah

1. Apa itu Junit?
2. Apa tujuan Junit?
3. Apa saja Unit dan Test dalam Junit?
4. Apa Manfaat Junit?

B. Tujuan Penelitian

5. Mengetahuhi apa itu Junit
6. Mengetahuhi tujuan Junit
7. Mengetahuhi Unit dan Test dalam Junit

8. Mengetahuhi manfaat Junit

BAB II
PEMBAHASAN

A. Pengertian Junit

Unit adalah kerangka kerja pengujian unit untuk Java bahasa pemrograman. JUnit sangat penting dalam pengembangan berbasis test dan merupakan salah satu keluarga kerangka kerja pengujian unit, yang secara kolektif dikenal sebagai xUnit yang berasal dari SUnit. JUnit sebagian besar digunakan oleh pengembang. JUnit dirancang untuk pengujian unit, yang Ini benar-benar proses pengkodean, bukan proses pengujian. Tetapi banyak penguji atau insinyur QA juga diminta untuk menggunakan JUnit untuk pengujian unit. Menulis lebih banyak tes akan membuat lebih produktif, tidak kurang produktif. Tes harus dilakukan sesegera mungkin. pada tingkat unit kode. JUnit membuat pengujian unit lebih mudah dan Cepat. Kelas JUnit adalah kelas penting, digunakan dalam menulis dan menguji JUnits. Beberapa kelas penting adalah Assert, TestCase dan TestResult. Assert berisi satu set assert Metode. Kasus Uji berisi kasus uji yang menentukan fixture untuk menjalankan beberapa tes.

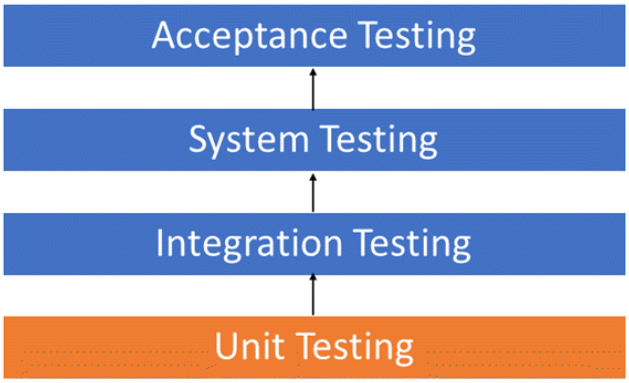
B. Tujuan Junit

1. Dapat mempermudah dalam melakukan pengembangan berbais test
2. Dapat mempermudah dalam pengujian unit
3. Mempermudah dalam memeriksa hasil unit

C. Unit dan Test dalam Junit

UNIT TESTING adalah jenis pengujian perangkat lunak di mana masing-masing unit atau komponen suatu perangkat lunak diuji. Tujuannya adalah untuk memvalidasi bahwa setiap unit kode perangkat lunak melakukan seperti yang diharapkan. Unit Testing dilakukan selama pengembangan (fase pengkodean) aplikasi oleh pengembang. Unit testing mengisolasi bagian kode dan memverifikasi kebenarannya. Unit dapat berupa fungsi, metode, prosedur, modul, atau objek individual. Dalam SDLC, STLC, V Model, pengujian unit adalah pengujian tingkat pertama yang dilakukan sebelum pengujian integrasi. Unit testing adalah teknik pengujian WhiteBox yang biasanya dilakukan oleh pengembang. Padahal, dalam dunia yang praktis karena krisis waktu atau keengganan pengembang untuk menguji, insinyur QA juga melakukan pengujian unit. Terkadang pengembang perangkat lunak berupaya menghemat waktu dengan melakukan pengujian unit minimal. Ini adalah mitos karena melewatkan pengujian unit mengarah ke biaya

perbaikan cacat yang lebih tinggi selama pengujian sistem, pengujian Integrasi dan bahkan Pengujian Beta setelah aplikasi selesai. Pengujian unit yang tepat dilakukan pada tahap pengembangan menghemat waktu dan uang pada akhirnya. Di sini, adalah alasan utama untuk melakukan pengujian unit.



Tingkat Pengujian Unit

1. Tes unit membantu untuk memperbaiki bug di awal siklus pengembangan dan menghemat biaya.
2. Ini membantu pengembang untuk memahami basis kode dan memungkinkan mereka untuk membuat perubahan dengan cepat
3. Tes unit yang baik berfungsi sebagai dokumentasi proyek
4. Tes unit membantu penggunaan kembali kode. Migrasikan kode Anda **dan** tes Anda ke proyek baru Anda. Tweak kode hingga tes berjalan lagi.

Cara melakukan Pengujian Unit

Unit Testing terdiri dari dua jenis

- Manual
- Otomatis

Pengujian unit umumnya otomatis tetapi masih dapat dilakukan secara manual. Rekayasa Perangkat Lunak tidak mendukung satu sama lain tetapi otomatisasi lebih disukai. Pendekatan manual untuk pengujian unit dapat menggunakan dokumen petunjuk langkah demi langkah.

Di bawah pendekatan otomatis-

- Pengembang menulis bagian kode dalam aplikasi hanya untuk menguji fungsinya. Mereka nantinya akan berkomentar dan akhirnya menghapus kode tes ketika aplikasi siap digunakan.

- Pengembang juga dapat mengisolasi fungsi untuk mengujinya dengan lebih ketat. Ini adalah praktik pengujian unit yang lebih menyeluruh yang melibatkan salin dan tempel kode ke lingkungan pengujiannya sendiri daripada lingkungan alaminya. **Mengisolasi kode membantu mengungkapkan ketergantungan yang tidak perlu antara kode yang sedang diuji dan unit atau ruang data lain** dalam produk. Ketergantungan ini kemudian dapat dihilangkan.

- Seorang coder umumnya menggunakan Kerangka UnitTest untuk mengembangkan kasus uji otomatis. Menggunakan kerangka kerja otomatisasi, pengembang kode membuat kriteria ke dalam tes untuk memverifikasi kebenaran kode. Selama pelaksanaan kasus uji, kerangka kerja mencatat kasus uji. Banyak kerangka kerja juga akan secara otomatis menandai dan melaporkan, secara ringkas, kasus pengujian yang gagal ini. Tergantung pada tingkat keparahan kegagalan, kerangka kerja dapat menghentikan pengujian selanjutnya.

- Alur kerja Unit Testing adalah 1) Buat Kasus Uji 2) Review / Pengerjaan Ulang 3) Baseline 4) Jalankan Uji Kasus.

Teknik Pengujian Unit

Teknik cakupan kode yang digunakan dalam pengujian terpadu tercantum di bawah ini:

- Cakupan Pernyataan
- Cakupan Keputusan
- Cakupan Cabang
- Cakupan Kondisi
- Cakupan Mesin Finite State

TestResult berisi metode untuk mengumpulkan hasil melaksanakan kasus uji coba. JUnit TestCase adalah kelas dasar di Kerangka kerja JUnit. Kasus uji mendefinisikan fixture untuk dijalankan beberapa tes. Untuk menentukan kasus uji: Menerapkan subkelas TestCase; Menentukan variabel instans yang menyimpan keadaan fixture; Menginisialisasi keadaan fixture dengan mengesampingkan setup; Pembersihan setelah tes dengan mengesampingkan teardown. Setiap tes berjalan

dalam perlengkapannya sendiri sehingga tidak ada efek samping di antara tes Berjalan. JUnit TestSuite adalah kelas kontainer di Junit Kerangka. Test Suite memungkinkan pengelompokan beberapa kasus uji ke dalam koleksi dan menjalankannya bersama-sama. Kelas TestSuite Tidak lagi didukung di JUnit. Untuk menjalankan hanya tes yang dipilih, posisikan kursor pada nama metode uji dan gunakan pintasan. Untuk melihat hasil tes JUnit, Eclipse menggunakan tampilan JUnit yang menunjukkan hasil tes. Pilih tes unit individual dalam tampilan ini, klik kanan pada mereka dan pilih Jalankan untuk mengeksekusinya lagi.

D. Manfaat Penggunaan Junit

Jika menemukan bug di awal kode, yang membuat kode kami lebih dapat diandalkan. JUnit berguna bagi pengembang, yang bekerja di lingkungan berbasis tes. Pengujian unit memaksa pengembang untuk membaca kode lebih dari menulis. Anda mengembangkan kode yang lebih mudah dibaca, dapat diandalkan, dan bebas bug yang membangun kepercayaan diri selama pengembangan.

BAB III
PENGGUNAAN TOOLS DEVOPS

Metodologi

Dalam Tes-Pertama Pengembangan, pengembang menulis pengujian unit otomatis untuk fungsionalitas baru, mereka akan mengimplementasikannya. Ini adalah sebuahproses rekayasa perangkat lunak yang mengikuti smallsiklus pengembangan. Tes unit otomatis untuk semua aplikasi dapat ditulis dalam dua cara: Sebelum kode implementasi dan setelah implementasi kode. Jika tes unit ditulis sebelum implementasi kode apa pun maka diketahui sebagai Test-Driven Development (atau) Test-First Development. Jika Tes unit ditulis setelah implementasi kode maka itu adalah dikenal sebagai Tes setelah Pengembangan (atau) Tes-Terakhir Pengembangan.

4.1 Siklus Pengembangan Tes-Pertama

Uji siklus model pengembangan pertama terdiri dari:
1. Menulis Tes: Tes unit (manual atau otomatis,lebih disukai otomatis) pertama kali ditulis untuk menjalankan fungsionalitas yang ditargetkan untuk pengembangan. Sebelum menulis tes , pengembang bertanggung jawab untuk memahami persyaratan dengan baik. Tes unit juga harus berisi pernyataan untuk mengkonfirmasi kriteria lulus/gagal unit tes.

2. Jalankan untuk gagal / buat kompilasi: Karena fitur ini belum ada diimplementasikan, unit test yang ditulis pada Langkah 1 adalah pasti gagal. Langkah ini pada dasarnya merupakan langkah validasi untuk tes unit tertulis, karena tes tidak boleh lulus bahkan jika tidak ada kode yang ditulis untuk itu. Sering kali pengujian unit dilakukan secara otomatis dan di sana kemungkinan tes gagal karena sintaks atau kesalahan kompilasi. Sanitasi tes dengan menghapus kesalahan ini juga merupakan bagian penting dari langkah ini.

3. Menerapkan fungsionalitas (lengkap / sebagian): Ini langkah melibatkan pengembangan bagian dari fungsionalitas untuk dimana unit test ditulis dan akan divalidasi.

4. Membuat tes untuk lulus: Setelah tes unit untuk yang dikembangkan kode telah berlalu, pengembang memperoleh keyakinan bahwa kode memenuhi persyaratan.

5. Pemfaktoran ulang kode: Tes unit mungkin telah lulus, tetapi refactoring kode mungkin masih diperlukan karena alasan termasuk:menangani kesalahan secara elegan, melaporkan hasilnya diformat yang diperlukan, atau mengukir sub rutin dari yang tertulis kode untuk dapat digunakan kembali.

6. Mengulangi siklus: Tes unit/set tes unit adalah refactored untuk memenuhi fungsionalitas baru atau mendorong ke arah penyelesaian fungsionalitas.

4.2 Tes Tulis

Pengujian di TFD agak mirip dengan pengujian unit denganperbedaan bahwa mereka ditulis untuk perilaku, bukan untukmetode. Adalah penting bahwa tes ditulis sedemikian rupa sehinggaurutan independen, yaitu hasilnya tetap samaterlepas dari urutan di mana tes dijalankan. Kapanmenulis tes, harus diingat bahwa tesharus berkonsentrasi pada pengujian perilaku yang benar, yaitu jika asistem harus menangani banyak input, pengujian harus mencerminkanbeberapa masukan.

4.3 Jalankan Tes

Tes dijalankan untuk memverifikasi apakah tes tertulis berguna atau tidak. Dikasus, uji kasus berlalu, ini menunjukkan bahwa tes tidak berharga.Ini juga memvalidasi bahwa test harus bekerja dengan benar dan bahwa tes baru tidak salah lulus tanpa membutuhkan kode baru. Tes baru juga harus gagal untuk alasan yang diharapkan. Ini meningkatkan kepercayaan diri bahwa itu sedang menguji hal yang benar, dan akan berlalu hanya dalam kasus-kasus yang dimaksudkan.

4.4 Tulis Kode

Penulisan kode sebenarnya adalah proses pembuatantes kerja, yaitu menulis kode yang lolos tes, misalnya, dengan mengganti mengembalikan nilai dari beberapa metode dengan konstanta. Ini menyediakan cara cepat untuk membuat tes lulus. Ini memiliki efek sementara memberikan kepercayaan diri programmer untuk melanjutkan refactoring dan juga menangani kontrol ruang lingkup dengan mulai dari satu contoh konkret dan kemudian menggeneralisasi dari sana. Implementasi abstrak didorong melalui merasakan duplikasi antara tes dan kode.

4.5 Pemfaktoran Ulang

Refactoring adalah proses meningkatkan internal struktur dengan mengedit kode kerja yang ada, tanpa mengubah perilaku eksternalnya. Hal ini penting, karena integritas desain perangkat lunak

tersebar dari waktu ke waktu karena akumulasi tekanan modifikasi, peningkatan dan perbaikan kerusakan. Sekarang kode dapat dibersihkan seperlunya.Ide dari refactoring adalah untuk melakukan modifikasi sebagai serangkaian langkah-langkah kecil tanpa memperkenalkan cacat baru ke dalam sistem.

4.6 Duplikasi Kode

Duplikat dalam berbagai bentuknya adalah kematian kebaikan kode. Duplikasi adalah masalah yang dimiliki beberapa orang memperingatkan tentang hal itu panjang lebar. Jika pengembang membenci duplikat kode, pengembang perlu melihat lebih dekat untuk mendeteksi apa itu sedang mengerjakan. Sepertinya ada sesuatu yang berguna atau penting sedang dilakukan. Pengembang harus meletakkannya di satu tempat.Pengembang dapat melakukan ini dengan mengekstrak kode duplikat ke dalam metode terpisah yang kemudian dapat dipanggil dari beberapalokasi. Jika duplikasi berada dalam hierarki pewarisan,Pengembang mungkin dapat mendorong kode yang digandakan ke atas hirarki. Jika struktur beberapa kode diduplikasi tetapi tidakdetailnya, Pengembang dapat mengekstrak bagian yang berbeda dan membuat metode template dari struktur umum.

4.7 Alat Pengujian Unit TFD: Junit

JUnit adalah kerangka kerja pengujian unit untuk Java bahasa pemrograman. JUnit telah menjadi penting dalam pengembangan pengembangan yang digerakkan oleh tes dan merupakan salah satu dari keluarga kerangka pengujian unit, yang secara kolektifdikenal sebagai xUnit yang berasal dari SUnit. JUnit kebanyakan digunakan oleh pengembang. JUnit dirancang untuk pengujian unit, yang benar-benar merupakan proses pengkodean, bukan proses pengujian. Tapi banyak penguji atau insinyur QA juga diharuskan menggunakan JUnit untuk pengujian satuan. Menulis lebih banyak tes akan membuat lebih produktif,tidak kalah produktif. Tes harus dilakukan sesegera mungkin pada tingkat unit kode. JUnit membuat pengujian unit lebih mudah dan lebih cepat. Kelas JUnit adalah kelas penting, digunakan dalam menulisdan menguji JUnit. Beberapa kelas penting adalah Assert,TestCase dan TestResult. Assert berisi satu set asertmetode. Test Case berisi test case yang mendefinisikan perlengkapan untuk menjalankan beberapa tes.TestResult berisi metode untuk mengumpulkan hasil dari mengeksekusi kasus uji. JUnit TestCase adalah kelas dasar dikerangka kerja JUnit. Kasus uji menentukan perlengkapan yang akan dijalankan beberapa tes. Untuk mendefinisikan kasus uji: Terapkan subkelas dari Kasus cobaan; Tentukan variabel instan yang menyimpan status perlengkapan; Inisialisasi status perlengkapan dengan mengesampingkan pengaturan;Pembersihan setelah pengujian dengan mengesampingkan teardown. Setiap tes berjalan dalam perlengkapannya sendiri sehingga tidak ada efek samping di antara tes berjalan. JUnit Test Suite adalah kelas kontainer di JUnit kerangka. Test Suite memungkinkan pengelompokan beberapa kasus uji ke dalam koleksi dan menjalankannya bersama-sama. Kelas Test Suite tidak lagi didukung di JUnit. Untuk menjalankan hanya tes yang dipilih, posisikan kursor pada nama metode pengujian dan gunakan

jalan pintas. Untuk melihat hasil tes JUnit, Eclipse menggunakan tampilan JUnit yang menunjukkan hasil pengujian.Pilih tes unit individu dalam tampilan ini, klik kanan pada mereka dan pilih Jalankan untuk menjalankannya lagi.

BAB IV
PENUTUP

Perbandingan

Metode DevOps

DevOps memiliki fokus utama pada budaya pengembangan software yang menekankan responsiveness.

Metode Waterfall

Metode Waterfall memiliki alur yang cenderung runtut dari atas ke bawah. Mulai dari tahap conception, initiation, analysis hingga deployment, setiap tahapan dilakukan secara berurutan.

Metode RAD

Metode RAD ini lebih berfokus pada pembuatan prototype secara cepat dan mengandalkan feedback dari user.

Metode Agile

Metode Agile memiliki fokus pada perubahan sambil mempercepat proses delivery produk.

Metode Prototyping

Metode Prototyping, proses pengerjaan yang dilakukan hanya bergantung pada satu orang saja yaitu pada pembuat perangkat lunak.

Dari Perbandingan Beberapa Metode Diatas Metode yang baik digunakan sesuai dengan kebutuhan. Seperti contohnya Metode yang kami rasa baik untuk di gunakan pada projek kami yaitu Metode RAD.