

**LAPORAN PRAKTIKUM
PENGUJIAN PERANGKAT LUNAK**



Disusun Oleh :

NAMA : NUR MUHAMMAD SYAIFUDDIN

NIM : 32601900026

**JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM SULTAN AGUNG
SEMARANG**

2021

HALAMAN PENGESAHAN

Laporan Praktikum Pengujian Perangkat Lunak

Disusun Oleh :

Nur Muhammad Syaifuddin (32601900026)

Telah disetujui sebagai syarat untuk memenuhi mata kuliah Pengujian Perangkat Lunak. Jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Sultan Agung Semarang.

Hari :

Tanggal :

Asisten Praktikum

1. Sigit Ardianto

1.

Mengetahui,

Laboran

Dosen Praktikum

WerdhaWilubertha H, S.Kom

Andi Riansyah, ST., M.Kom

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT, yang telah memberikan rahmat, taufik serta hidayah-Nya, sehingga laporan Pengujian Perangkat Lunak dapat terselesaikan.

Tanpa lupa penulis mengucapkan terima kasih kepada :

1. Rektor UNISSULA Bapak Drs. H. Bedjo Santoso, M.T., P.h.D yang mengizinkan penulis menimba ilmu di kampus ini.
2. Dekan Fakultas Teknologi Industri Ibu Dr, Novi Marlyana, ST., MT.
3. Dosen pengampu Bapak Andi Riansyah, ST., M.Kom yang telah memberi ilmu tentang Pengujian Perangkat Lunak.
4. Orang tua penulis yang telah mengizinkan untuk menyelesaikan laporan ini.
5. Dan kepada semua pihak yang tidak dapat saya sebutkan satu persatu.

Penulis menyadari bahwa dalam penyusunan laporan ini masih terdapat banyak kekurangan, untuk itu penulis mengharap kritik dan saran dari pembaca untuk sempurnanya laporan ini. Semoga dengan ditulisnya laporan ini dapat menjadi sumber ilmu bagi setiap pembaca.

Semarang, 20 Mei 2021

Nur Muhammad Syaifuddin

DAFTAR ISI

HALAMAN JUDUL	i
HALAMAN PENGESAHAN	ii
KATA PENGANTAR.....	iii
DAFTAR ISI.....	iv
DAFTAR GAMBAR.....	vii
DAFTAR TABEL	x
BAB I PENGUJIAN <i>WHITEBOX</i> (1): PENGUJIAN <i>BASIS PATH</i>.....	1
1.1 Tujuan.....	1
1.2 Alat dan Bahan	1
1.3 Dasar Teori	1
1.3.1 Teknik Pengujian Perangkat Lunak	1
1.3.2 Tujuan Pengujian	1
1.3.3 Prinsip Pengujian	2
1.3.4 Perancangan Kasus Uji atau <i>Test Case</i>	2
1.3.5 Pengujian <i>Whitebox</i> atau <i>Glassbox</i>	2
1.3.6 <i>Basis Path Testing</i>	3
1.4 Prosedur Praktikum	4
1.5 Hasil Praktikum	7
1.6 Analisa.....	9
1.7 Kesimpulan.....	12
BAB II PENGUJIAN <i>WHITEBOX</i> (2): PENGUJIAN KONDISI.....	13
2.1 Tujuan.....	13
2.2 Alat dan Bahan	13
2.3 Dasar Teori	13
2.3.1 Teknik Pengujian Perangkat Lunak	13
2.4 Prosedur Praktikum	15
2.5 Hasil Praktikum	18
2.6 Analisa.....	21
2.7 Kesimpulan.....	26

BAB III PENGUJIAN <i>BLACKBOX</i> (1): PENGUJIAN ANALISIS KELAS EKUIVALENSI.....	27
3.1 Tujuan.....	27
3.2 Alat dan Bahan	27
3.3 Dasar Teori	27
3.3.1 <i>Blackbox Testing</i>	27
3.3.2 Analisis Partisi Kelas Ekuivalensi	28
3.4 Prosedur Praktikum	29
3.5 Hasil Praktikum	39
3.6 Analisa.....	44
3.7 Kesimpulan.....	48
BAB IV PENGUJIAN <i>BLACKBOX</i> (2): PENGUJIAN ANALISIS NILAI BATAS	49
4.1 Tujuan.....	49
4.2 Alat dan Bahan	49
4.3 Dasar Teori	49
4.3.1 Analisis Nilai Batas (<i>Boundary Value Analysis</i> atau BVA)	49
4.3.2 Petunjuk Pengujian BVA	50
4.4 Prosedur Praktikum	50
4.5 Hasil Praktikum	54
4.6 Analisa.....	57
4.7 Kesimpulan.....	59
BAB V PENGUJIAN FUNGSIONAL DENGAN SELENIUM IDE.....	60
5.1 Tujuan.....	60
5.2 Alat dan Bahan	60
5.3 Dasar Teori	60
5.3.1 Pengenalan Selenium	60
5.3.2 Selenium IDE	61
5.3.3 Mengeset <i>Recording</i> dan <i>Run</i>	64
5.3.4 Menjalankan <i>Test Case</i>	64
5.4 Prosedur Praktikum	66

5.5	Hasil Praktikum	71
5.6	Analisa.....	73
5.7	Kesimpulan.....	74
BAB VI PENGUJIAN <i>STRESS</i> DENGAN APACHE JMETER		75
6.1	Tujuan.....	75
6.2	Alat dan Bahan	75
6.3	Dasar Teori	75
6.4	Prosedur Praktikum	77
6.5	Hasil Praktikum	84
6.6	Analisa.....	88
6.7	Kesimpulan.....	88
DAFTAR PUSTAKA		
LAMPIRAN		

DAFTAR GAMBAR

Gambar 1. 1 Diagram alir.....	3
Gambar 1. 2 Flowchart basis path.....	3
Gambar 1. 3 Grafik alir basis path	4
Gambar 1. 4 Grafik program	6
Gambar 1. 5 Hasil uji 1	7
Gambar 1. 6 Hasil uji 2	8
Gambar 1. 7 Hasil uji 3	8
Gambar 1. 8 Hasil uji 4	8
Gambar 1. 9 Hasil uji 5	9
Gambar 2. 1 Hasil pengujian rule 1	19
Gambar 2. 2 Hasil pengujian rule 2	19
Gambar 2. 3 Hasil pengujian rule 3	19
Gambar 2. 4 Hasil pengujian rule 4	20
Gambar 2. 5 Hasil pengujian rule 7	20
Gambar 2. 6 Hasil pengujian rule 9	20
Gambar 2. 7 Hasil pengujian rule 10	20
Gambar 2. 8 Hasil pengujian rule 11	21
Gambar 3. 1 Hasil test case 1 blackbox equivalence class	39
Gambar 3. 2 Hasil test case 2 blackbox equivalence class	39
Gambar 3. 3 Hasil test case 3 blackbox equivalence class	39
Gambar 3. 4 Hasil test case 4 blackbox equivalence class	40
Gambar 3. 5 Hasil test case 5 blackbox equivalence class	40
Gambar 3. 6 Hasil test case 6 blackbox equivalence class	40
Gambar 3. 7 Hasil test case 7 blackbox equivalence class	41
Gambar 3. 8 Hasil test case 8 blackbox equivalence class	41
Gambar 3. 9 Hasil test case 9 blackbox equivalence class	41
Gambar 3. 10 Hasil test case 10 blackbox equivalence class	42
Gambar 3. 11 Hasil test case 11 blackbox equivalence class	42
Gambar 3. 12 Hasil test case 12 blackbox equivalence class	42

Gambar 3. 13 Hasil test case 13 blackbox equivalence class	43
Gambar 3. 14 Hasil test case 14 blackbox equivalence class	43
Gambar 3. 15 Hasil test case 15 blackbox equivalence class	43
Gambar 3. 16 Hasil test case 16 blackbox equivalence class	44
Gambar 3. 17 Hasil test case 17 blackbox equivalence class	44
Gambar 4. 1 Hasil test case 1 blackbox boundary analysis value.....	54
Gambar 4. 2 Hasil test case 2 blackbox boundary analysis value.....	55
Gambar 4. 3 Hasil test case 3 blackbox boundary analysis value.....	55
Gambar 4. 4 Hasil test case 4 blackbox boundary analysis value.....	55
Gambar 4. 5 Hasil test case 5 blackbox boundary analysis value.....	56
Gambar 4. 6 Hasil test case 6 blackbox boundary analysis value.....	56
Gambar 4. 7 Hasil test case 7 blackbox boundary analysis value.....	56
Gambar 4. 8 Hasil test case 8 blackbox boundary analysis value.....	57
Gambar 5. 1 Panel Selenium IDE	62
Gambar 5. 2 Website Selenium IDE	66
Gambar 5. 3 Pop up "add extension"	66
Gambar 5. 4 Berhasil menambahkan extension Selenium IDE pada Chrome.....	67
Gambar 5. 5 Tab "extensions"	67
Gambar 5. 6 Halaman awal Selenium IDE	68
Gambar 5. 7 Hasil test case “energy efficient” Google Search.....	71
Gambar 5. 8 Hasil test case “Selenium RC” Google Search	71
Gambar 5. 9 Hasil test case “energy efficient” Google Search.....	72
Gambar 5. 10 Hasil test case “energy efficient” Yahoo Search.....	72
Gambar 6. 1 Mengecek versi JDK	77
Gambar 6. 2 Mengecek versi JRE.....	78
Gambar 6. 3 Memilih bahasa	79
Gambar 6. 4 Direktori instalasi Apache JMeter	79
Gambar 6. 5 Start menu Apache JMeter	80
Gambar 6. 6 Menambahkan shortcut Apache JMeter pada desktop	80
Gambar 6. 7 Installation resume	81
Gambar 6. 8 Proses instalasi	81

Gambar 6. 9 Instalasi Apache JMeter selesai.....	82
Gambar 6. 10 Tampilan Apache JMeter	82
Gambar 6. 11 Membuat Thread Group	84
Gambar 6. 12 Membuat HTTP request defaults	85
Gambar 6. 13 Membuat HTTP request	85
Gambar 6. 14 Menambahkan Listener Graph Results	86
Gambar 6. 15 Menambahkan Listener View Results in Table	86
Gambar 6. 16 Results in Graphic	87
Gambar 6. 17 Results in Table	87

DAFTAR TABEL

Tabel 1. 1 Uji basis path.....	6
Tabel 1. 2 Hasil uji basis path	7
Tabel 2. 1 Tabel keputusan atau kondisi	17
Tabel 2. 2 Hasil pengujian kondisi.....	18
Tabel 3. 1 Weak equivalence class testing	31
Tabel 3. 2 Weak robust equivalence class testing	32
Tabel 3. 3 Strong robust equivalence class testing	35
Tabel 4. 1 Test case boundry value anlysis.....	52
Tabel 6. 1 Elemen dasar pengujian menggunakan JMeter.....	76

BAB I

PENGUJIAN *WHITEBOX* (1): PENGUJIAN *BASIS PATH*

1.1 Tujuan

1. Praktikan dapat mempersiapkan tahapan pengujian
2. Praktikan dapat merancang kasus uji
3. Praktikan dapat melakukan pengujian perangkat lunak dengan teknik *Basis Path*

1.2 Alat dan Bahan

Alat dan bahan yang dibutuhkan dalam modul BAB I ini adalah sebuah komputer yang sudah ter-install IDE (*Integrated Development Environment*) “CodeBlocks” yang memiliki *compiler C*.

1.3 Dasar Teori

1.3.1 Teknik Pengujian Perangkat Lunak

Pengujian *software* adalah elemen kritis dari jaminan kualitas *software* dan merupakan *review* akhir dari spesifikasi, perancangan dan pengkodean. Pada tahap awal dari pengembangan *software*, *engineer* berusaha untuk membangun *software* dari sebuah konsep abstrak menjadi implementasi nyata. Pada saat pengujian, *engineer* membuat serangkaian kasus uji yang bertujuan untuk “merusak” *software* yang telah dibuat. (Sugiyono, 2018)

1.3.2 Tujuan Pengujian

Tujuan pengujian perangkat lunak menurut Glen Meyer adalah sebagai berikut :

1. Pengujian adalah proses eksekusi sebuah program untuk menemukan “*error*”.
2. Kasus uji yang baik adalah sesuatu yang bisa mengungkapkan kemungkinan yang tinggi untuk menemukan *error-error* yang tidak ditemukan sebelumnya.

3. Pengujian yang sukses adalah sesuatu yang bisa mengungkapkan *error* yang tidak ditemukan sebelumnya.

1.3.3 Prinsip Pengujian

Pada saat melakukan pengujian terhadap suatu *software* ada beberapa prinsip pengujian yang harus diperhatikan, diantaranya :

1. Semua pengujian harus terlacak kebutuhan *user*.
2. Pengujian harus direncanakan jauh sebelum pengujian dimulai.
3. Pengujian dimulai dari kecil dan mengarah kepada yang besar.
4. Pengujian yang sempurna adalah tidak mungkin.
5. Agar pengujian berjalan efektif, maka sebaiknya dilakukan oleh pihak ketiga yang netral.

1.3.4 Perancangan Kasus Uji atau *Test Case*

Setiap produk rekayasa perangkat lunak bisa diuji dalam dua acara yaitu sebagai berikut :

1. Mengetahui fungsinya, sehingga pengujian dilakukan dengan mendemonstrasikan fungsi tersebut bisa dengan sempurna atau ada *error* (*Blackbox Testing*).
2. Mengetahui cara kerja internal dari produk tersebut (*Whitebox Testing*).

1.3.5 Pengujian *Whitebox* atau *Glassbox*

Adalah sebuah metoda perancangan kasus uji yang menggunakan struktur kontrol dari perancangan prosedur. Pengguna metoda pengujian akan membuat *software engineer* dapat : (Rusfandi, 2018)

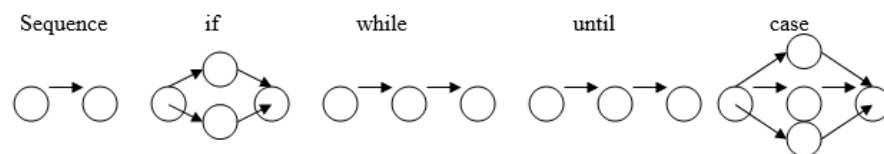
1. Menjamin bahwa semua jalur independen dalam sebuah modul telah dilewati paling tidak satu kali.
2. Memeriksa semua keputusan logika baik pada sisi sebenarnya maupun pada sisi salahnya.
3. Mengeksekusi semua *loop* pada nilai batasnya dan dalam nilai dimana dia harus berjalan.

1.3.6 Basis Path Testing

Metoda *basis path testing* membuat perancangan kasus uji bisa menurunkan sebuah ukuran kompleksitas logika dari sebuah perancangan prosedural dan ukuran ini selanjutnya digunakan untuk menentukan sekelompok data dasar (*basis set*) dari jalur eksekusi.

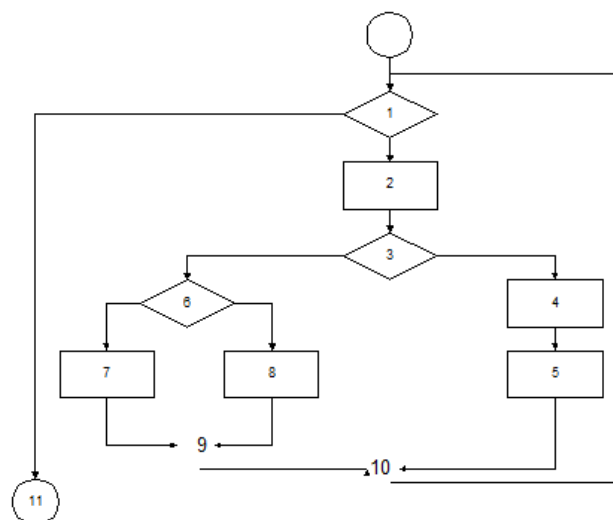
Pengujian *basis path* adalah teknik pengujian *whitebox* yang diusulkan Tom McCabe. Metode ini memungkinkan perancang *test case* mendapatkan ukuran kekompleksitasan logika dari perancangan prosedural dan menggunakan ukuran ini sebagai petunjuk untuk mendefinisikan *basis set* dari jalur pengerjaan. *Test case* yang didapat digunakan untuk mengerjakan *basis set* yang menjamin pengerjaan setiap perintah minimal satu kali selama pengujian.

1. Notasi Diagram Alir



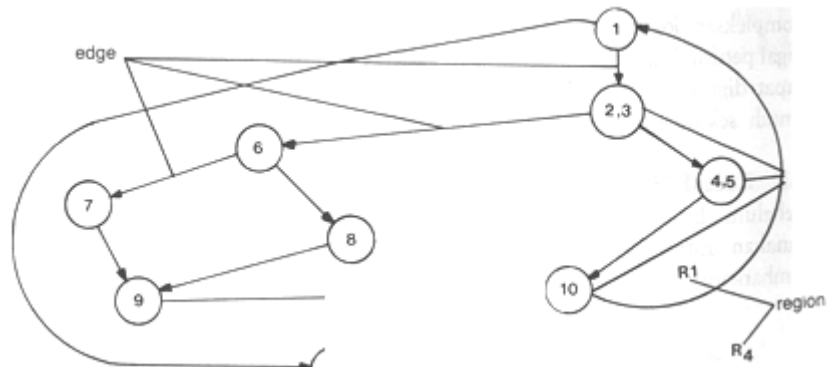
Gambar 1. 1 Diagram alir

Untuk menggambarkan pemakaian diagram alir diberikan contoh perancangan dalam bentuk *flowchart*.



Gambar 1. 2 Flowchart basis path

Selanjutnya diagram alir di atas dipetakan ke grafik alir.



Gambar 1. 3 Grafik alir *basis path*

Lingkaran atau *node* menggambarkan satu atau lebih perintah prosedural. Urutan proses dan keputusan dapat dipetakan dalam satu *node*.

Tanda panah atau *edge* menggambarkan aliran kontrol. Setiap *node* harus mempunyai tujuan *node*.

Region adalah daerah yang dibatasi oleh *edge* dan *node*. Termasuk daerah diluar grafik alir.

1.4 Prosedur Praktikum

Berikut ini merupakan prosedur praktikum program pencarian biner dalam Bahasa C.

1. Tentukan *basis path*
2. Turunkan *test case*-nya
3. Jalankan *test case* tersebut
4. Analisis hasil pengujian tersebut

Program pencarian biner :

```
#include <stdio.h>
#include <stdlib.h>

int binSrc(int x[], int low, int high, int key){
    int mid;
    while(low <= high){
        mid = (low + high)/2;
```

```

if(x[mid] == key){          3
return mid;                 8
}else if(x[mid] < key){     4
low = mid + 1;             5
}else{
high = mid - 1;           6
}
}                           7
return -1;                 8
}                           9

int main(){

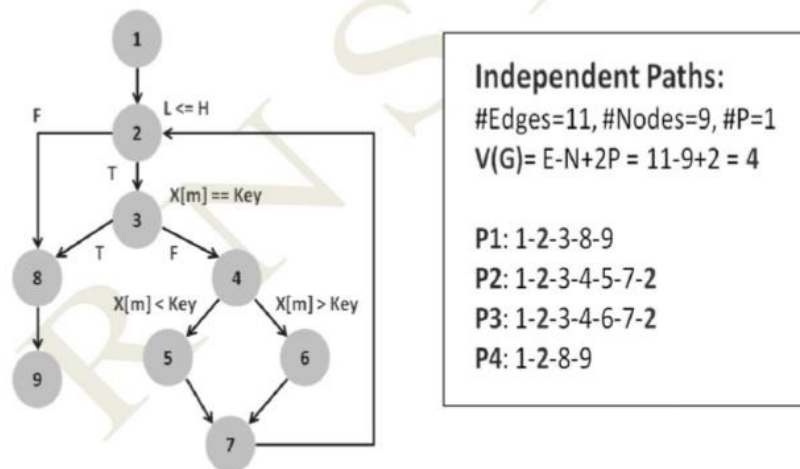
printf("Nama\t: Nur Muhammad Syaifuddin\n");
printf("NIM\t: 32601900026\n\n");

int a['k'], key, n, succ;
printf("Enter the n value : ");
scanf("%d", &n);

if(n > 0){
printf("Enter the elements in ascending order\n");
for(int i = 0; i < n; i++){
scanf("%d", &a[i]);
}
printf("Enter the key element to be searched\n");
scanf("%d", &key);
succ = binSrc(a, 0, n-1, key);
if(succ >= 0){
printf("Element Found in position = %d\n", succ+1);
}else{
printf("Element not found in position = %d\n", succ);
}
}else{
printf("Number of element should be greater than zero \n");
}
return 0;
}

```

Grafik program :



Gambar 1. 4 Grafik program

Test case

Pre-kondisi :

Array mempunyai elemen – elemen yang urutannya naik **T/F**

Elemen *key* ada di dalam array **T/F**

Array mempunyai jumlah elemen ganjil **T/F**

Tabel 1. 1 Uji *basis path*

No	Path	Input		Expected Output	Remarks
		X[]	Key		
1	P1:1-2-3-8-9	(10,20,30,40,50)	30	3	3 (Pass)
2	P2:1-2-3-4-6-7-2	(10,20,30,40,50)	20		
3	P3:1-2-3-4-5-7-2	(10,20,30,40,50)	40		
4	P4:1-2-3-4-5-7-2--8-9	(10,20,30,40,50)	60		
5		kosong	Any key		

1.5 Hasil Praktikum

Tabel 1. 2 Hasil uji *basis path*

No	Path	Input		Expected Output	Remarks
		X[]	Key		
1	P1:1-2-3-8-9	(10,20,30,40,50)	30	3	3 (Pass)
2	P2:1-2-3-4-6-7-2	(10,20,30,40,50)	20	2	2 (pass)
3	P3:1-2-3-4-5-7-2	(10,20,30,40,50)	40	4	4 (pass)
4	P4:1-2-3-4-5-7-2-8-9	(10,20,30,40,50)	60	-1	-1 (pass)
5		kosong	Any key	Number of element should be greater than zero	Number of element should be greater than zero (pass)

Nama : Nur Muhammad Syaifuddin
NIM : 32601900026

```
Enter the n value : 5
Enter the elements in ascending order
10
20
30
40
50
Enter the key element to be searched
30
Element Found in position = 3

Process returned 0 (0x0)   execution time : 12.374 s
Press any key to continue.
```

Gambar 1. 5 Hasil uji 1

```

Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

Enter the n value : 5
Enter the elements in ascending order
10
20
30
40
50
Enter the key element to be searched
20
Element Found in position = 2

Process returned 0 (0x0)   execution time : 8.870 s
Press any key to continue.

```

Gambar 1. 6 Hasil uji 2

```

Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

Enter the n value : 5
Enter the elements in ascending order
10
20
30
40
50
Enter the key element to be searched
40
Element Found in position = 4

Process returned 0 (0x0)   execution time : 12.106 s
Press any key to continue.

```

Gambar 1. 7 Hasil uji 3

```

Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

Enter the n value : 5
Enter the elements in ascending order
10
20
30
40
50
Enter the key element to be searched
60
Element not found in position = -1

```

Gambar 1. 8 Hasil uji 4

```

Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

Enter the n value : 0
Number of element should be greater than zero

Process returned 0 (0x0)   execution time : 2.182 s
Press any key to continue.

```

Gambar 1. 9 Hasil uji 5

1.6 Analisa

Dalam percobaan praktikum bab 1 ini, menggunakan pengujian *whitebox* dengan teknik *basis path* untuk mengukur kompleksitas logis dari desain prosedural dan menggunakannya sebagai pedoman untuk menerapkan himpunan basis dari semua jalur eksekusi. penerapan *basis path* terlebih dahulu dari *flowchart/ flowgraph*.

1. Analisa Pengujian Nomor 1

Pada pengujian nomer 1, diperlihatkan mengenai pengujian melalui jalur independen yaitu 1-2-3-8-9. Sebagai contoh dimasukan data array $X = [10,20,30,40,50]$ dan kunci yang akan dicari memiliki kunci 30. Proses mencari *key* dimulai dari posisi array tengah, maka *function* `binsrc` langsung melakukan pengembalian nilai posisi array di angka 2 ke *function* `main`. Pada *function* `main` ini nilai balik ditambah satu, sehingga akan memberikan *output* ke layar yaitu nilai 3.

2. Analisa Pengujian Nomor 2

Pada pengujian nomer 2 diperlihatkan mengenai pengujian melalui jalur independen yaitu 1-2-3-4-6-7-2. Sebagi conntoh dimasukan data array $X = [10,20,30,40,50]$ dan *key* yang akan dicari memiliki nilai 20. Pada *function* `binsrc` mula - mula mendeklarasikan variabel `mid` (*node* 1), setelah itu masuk pada proses perulangan (*node* 2) karena memenuhi syarat perulangan yakni $low \leq high$ dengan nilai `low` adalah 0 dan `high` adalah 4, kemudian mendapatkan nilai `mid` dari operasi $low + high / 2$ (*node* 3), kemudian akan memeriksa apakah nilai tengah array X sama dengan nilai *key* (*node* 3), karena nilai tidak benar maka lanjut

ke *node* 4 untuk memeriksa apakah nilai X urutan mid lebih kecil daripada nilai *key*, karena pada kondisi tersebut bernilai salah maka proses berlanjut ke *statement* *else* yaitu variabel *low* diisi nilai variabel $mid - 1$ (*node* 6). Setelah itu program akan membaca pada *node* 7 dan mengulang kembali ke *node* 2. Ketika kembali ke *node* 2, maka terjadi proses perulangan hingga menemukan posisi array yang memiliki nilai sama dengan nilai *key*, yaitu pada posisi array indeks ke-1. Ketika nilai posisi array dikembalikan pada program utama akan ditambahkan 1, sehingga pada layar akan muncul pada posisi 2.

3. Analisa Pengujian Nomor 3

Pada pengujian nomer 3 diperlihatkan mengenai pengujian melalui jalur independen yaitu 1-2-3-4-5-7-2. Sebagai contoh dimasukan data array $X = [10,20,30,40,50]$ dan *key* yang akan dicari memiliki nilai 40. Pada *function* *binsrc* mula - mula mendeklarasikan variabel *mid* (*node* 1), setelah itu masuk pada proses perulangan (*node* 2) karena memenuhi syarat perulangan yaitu $low \leq high$ dengan nilai *low* adalah 0 dan *high* adalah 4, kemudian mendapatkan nilai *mid* dari operasi $low + high / 2$ (*node* 3), kemudian akan memeriksa apakah nilai tengah array X sama dengan nilai *key* (*node* 3), karena nilai tidak benar maka lanjut ke *node* 4 untuk memeriksa apakah nilai X urutan mid lebih kecil daripada nilai *key*, karena pada kondisi tersebut bernilai benar, maka variabel *low* diisi nilai variabel $mid + 1$ (*node* 5). Setelah itu program akan membaca pada *node* 7 dan mengulang kembali ke *node* 2. Ketika kembali ke *node* 2, maka terjadi proses perulangan hingga menemukan posisi array yang memiliki nilai sama dengan nilai *key*, yaitu pada posisi array indeks ke-3. Ketika nilai posisi array dikembalikan pada program utama akan ditambahkan 1, sehingga pada layar akan muncul pada posisi 4.

4. Analisa Pengujian Nomor 4

Pada pengujian nomer 4 diperlihatkan mengenai pengujian melalui jalur *another path* yaitu 1-2-3-4-5-7-2-8-9. Sebagai contoh dimasukan data array $X = [10, 20, 30, 40, 50]$ dan *key* yang akan dicari memiliki nilai 60. Pada *function binsrc* mula - mula mendeklarasikan variabel *mid* (*node 1*), setelah itu masuk pada proses perulangan (*node 2*) karena memenuhi syarat perulangan yaitu $low \leq high$ dengan nilai *low* adalah 0 dan *high* adalah 4, kemudian mendapatkan nilai *mid* dari operasi $low + high / 2$ (*node 3*), kemudian akan memeriksa apakah nilai tengah array *X* sama dengan nilai *key* (*node 3*), karena nilai tidak benar maka lanjut ke *node 4* untuk memeriksa apakah nilai *X* urutan *mid* lebih kecil daripada nilai *key*, karena pada kondisi tersebut bernilai benar, maka variabel *low* diisi nilai variabel $mid + 1$ (*node 5*). Setelah itu program akan membaca pada *node 7* dan mengulang kembali ke *node 2*. Ketika kembali ke *node 2*, maka terjadi proses perulangan hingga suatu kondisi dimana perulangan berhenti, karena $low > high$, dan program akan mengembalikan nilai -1 (*node 8*) dan proses eksekusi pada *function binsrc* berakhir (*node 9*). Ketika nilainya dikembalikan pada program utama akan di cek apakah nilainya lebih dari sama dengan 0, karena lebih kecil dari 0, maka pada layar akan muncul pada posisi -1.

5. Analisa Pengujian Nomor 5

Pada pengujian nomer 5, jumlah array yang dimasukan adalah 0. Kemudian dilakukan pengecekan apakah jumlah array yang dimasukan lebih dari 0. Karena jumlah array yang dimasukan sama dengan 0, maka program akan mengeksekusi *statement else* yaitu menampilkan *output* pada *console* “*Number of element should be greater than zero*”, yang artinya elemen pada *array* tidak boleh sama dengan 0 (kosong). Sehingga pada pengujian Nomor 5 *function binsrc* tidak dieksekusi.

Dari keseluruhan analisa di atas, kita berhasil melakukan kelima pengujian dengan sukses. Karena semua ekspektasi kita harapkan sesuai dengan hasil yang ditampilkan

1.7 Kesimpulan

Dari hasil praktikum pada BAB I di atas, dapat disimpulkan bahwa pengujian *basis path* berfungsi untuk memeriksa seluruh kemungkinan yang mungkin akan terjadi dengan cara menginputkan berbagai macam kombinasi dan memastikan program berjalan sesuai dengan ketentuan. Dengan menggunakan metode ini memungkinkan untuk menemukan *error* pada program dengan lebih mudah.

BAB II

PENGUJIAN *WHITEBOX* (2): PENGUJIAN KONDISI

2.1 Tujuan

1. Praktikum dapat mempersiapkan tahapan pengujian
2. Praktikan dapat merancang kasus uji
3. Praktikan dapat melakukan pengujian – pwnujian perangkat lunak dengan teknik pengujian kondisi (menggunakan tabel keputusan)

2.2 Alat dan Bahan

Alat dan bahan yang dibutuhkan dalam modul BAB II ini adalah sebuah komputer yang sudah ter-*install* IDE (*Integrated Development Environtmen*) “CodeBlocks” yang memiliki *compiler* C.

2.3 Dasar Teori

2.3.1 Teknik Pengujian Perangkat Lunak

Pengujian kondisi merupakan metode desain *test case* yang menguji kondisi logika dalam suatu perangkat lunak. Pengujian ini dimaksudkan untuk mendeteksi *error* kondisi pada suatu kondisi perangkat lunak dan mendeteksi *error* lainnya dan mempunyai keuntungan sebagai berikut :

1. Pengukuran jangkauan pengujian kondisi adalah sederhana.
 2. Jangkauan pengujian kondisi pada perangkat lunak menyediakan petunjuk pembangkitan pengujian tambahan bagi perangkat lunak.
- Contoh: kondisi sederhana dari persamaan relasional

$E_1 \text{ (Operator relasional) } E_2$

E_1 dan E_2 merupakan persamaan matematika

Operator relasional adalah salah satu dari operator berikut ini:

$<, ?, =, ? (- =), >, ?$

Operator boolean: OR ('?'), AND ('&'), NOT ('-')

Yang termasuk strategi pengujian kondisi adalah :

1. Pengujian Cabang

Merupakan strategi pengujian kondisi yang paling sederhana. Untuk mencapai suatu kondisi gabungan C, cabang-cabang *true* dan *false* dari C dan setiap kondisi pada C perlu dieksekusi paling tidak satu kali.

2. Pengujian Domain

Membutuhkan tiga atau empat pengujian yang dilakukan untuk sebuah persamaan relasional.

$$E_1 > E_2, E_1 = E_2, E_1 < E_2$$

3. Pengujian BRO (*Branch and Relational Operator*)

Menggunakan batasan kondisi C. batasan kondisi C dengan n kondisi sederhana (D_1, D_2, \dots, D_n) dimana $D_i (0 < i \leq n)$ merupakan batasan akhir dari kondisi C.

$$C_2 : B_1 \& (E_3 = E_4)$$

B_1 adalah variabel boolean.

E_3 dan E_4 adalah persamaan matematika atau aritmatika.

Batasan kondisi C_2 adalah bentuk (D_1, D_2) dengan D_1 adalah 't' atau 'f', dan D_2 adalah '>', '=', '<'

C_2 adalah persamaan relasional dengan memodifikasi himpunan pembatas $\{(t, t), (f, t), (t, f)\}$.

't' untuk ($E_3=E_4$) mengimplikasikan '='

'f' untuk ($E_3=E_4$) mengimplikasikan '<' atau '>'

Dengan menggantikan :

$$\begin{aligned} (t,t) &\rightarrow (t, =) \\ (f,t) &\rightarrow (f, =) \\ (t,f) &\rightarrow (t, <) \text{ atau } (t, < >) \text{ atau } (t, >) \end{aligned}$$

Maka hasil himpunan batasan untuk C

$$\{ (t, =), (f, =), (t, >), (t, <) \}$$

2.4 Prosedur Praktikum

Berikut ini adalah kode program dalam bahasa C untuk masalah segitiga yaitu menerima tiga *integer* yang dianggap sebagai tiga sisi segitiga dan menentukan apakah ketiga nilai tersebut merepresentasikan segitiga sama sisi, segitiga sama kaki, segitiga yang tidak sama semua sisinya atau tidak membentuk segitiga.

1. Turunkan *test case* berdasarkan pengujian kondisi!
2. Jalankan *test case* tersebut!
3. Analisis hasil pengujian tersebut!

Program segitiga

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("Nama\t: Nur Muhammad Syaifuddin\n");
    printf("NIM\t: 32601900026\n");

    int a,b,c,c1,c2,c3;
    char istriangle;
    do
    {
        printf("\n\nenter 3 integers which are sides of triangle\n");
        scanf("%d%d%d",&a,&b,&c);
        printf("\na=%d\tb=%d\tc=%d\n\n",a,b,c);
        c1 = a>=1 && a<=10;
        c2= b>=1 && b<=10;
        c3= c>=1 && c<=10;
        if (!c1)
            printf("\nthe value of a=%d is not the range of permitted value",a);
        if (!c2)
            printf("\nthe value of b=%d is not the range of permitted value",b);
```

```

if (!c3)
printf("\nthe value of c=%d is not the range of permitted
value",c);
} while(!(c1 && c2 && c3));
// Mengecek segitiga atau tidak
if( a<b+c && b<a+c && c<a+b )
istriangle='y';
else
istriangle ='n';
if (istriangle=='y')
if ((a==b) && (b==c))
printf("equilateral triangle\n");
else if ((a!=b) && (a!=c) && (b!=c))
printf("scalene triangle\n");
else
printf("isosceles triangle\n");
else
printf("Not a triangle\n");

printf("\n");
return 0;
}

```

Test Case

Test data: Masukkan 3 nilai integer (a, b, dan c)

Pre-condition: $a < b + c$, $b < a + c$, $c < a + b$

Deskripsi singkat: cek apakah nilai merupakan segitiga sama sisi, segitiga sama kaki, segitiga yang tidak sama semua sisinya, atau tidak membentuk segitiga.

Tabel 2. 1 Tabel keputusan atau kondisi

<i>RULES</i>		R 1	R 2	R 3	R 4	R 5	R 6	R 7	R 8	R 9	R 1 0	R 1 1
<i>Conditio ns</i>	C1: $a < b + c$	F	T	T	T	T	T	T	T	T	T	T
	C2: $b < a + c$	-	F	T	T	T	T	T	T	T	T	T
	C3: $c < a + b$	-	-	F	T	T	T	T	T	T	T	T
	C4: $a = b$	-	-	-	T	T	T	T	F	F	F	F
	C5: $a = c$	-	-	-	T	T	F	F	T	T	F	F
	C6: $b = c$	-	-	-	T	F	T	F	T	F	T	F
<i>Actions</i>	A1: <i>Not a triangle</i>	X	X	X								
	A2: <i>Scalene triangle</i>											X
	A3: <i>Isosceles triangle</i>							X		X	X	
	A4: <i>Equilateral triangle</i>				X							
	A5: <i>Impossible</i>					X	X		X			

2.5 Hasil Praktikum

Tabel 2. 2 Hasil pengujian kondisi

Rules			C1	C2	C3	C4	C5	C6	Output yang diharapkan
			$a < b + c$	$b < a + c$	$c < a + b$	$a = b$	$a = c$	$b = c$	
R1	a	8	F	-	-	-	-	-	Not a triangle
	b	3							
	c	3							
R2	a	3	T	F	-	-	-	-	Not a triangle
	b	8							
	c	3							
R3	a	3	T	T	F	-	-	-	Not a triangle
	b	3							
	c	8							
R4	a	8	T	T	T	T	T	T	Equilateral triangle
	b	8							
	c	8							
R5	a		T	T	T	T	T	F	Impossible
	b								
	c								
R6	a		T	T	T	T	F	T	Impossible
	b								
	c								
R7	a	8	T	T	T	T	F	F	Isosceles triangle
	b	8							
	c	3							
R8	a		T	T	T	F	T	T	Impossible
	b								
	c								
R9	a	8	T	T	T	F	T	F	Isosceles triangle
	b	3							
	c	8							
R10	a	3	T	T	T	F	F	T	Isosceles triangle
	b	8							
	c	8							
R11	a	3	T	T	T	F	F	F	Scalene Triangle
	b	4							
	c	5							

```

Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
8
3
3

a=8      b=3      c=3

Not a triangle

Process returned 0 (0x0)   execution time : 3.891 s
Press any key to continue.

```

Gambar 2. 1 Hasil pengujian rule 1

```

Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
3
8
3

a=3      b=8      c=3

Not a triangle

Process returned 0 (0x0)   execution time : 6.847 s
Press any key to continue.

```

Gambar 2. 2 Hasil pengujian rule 2

```

Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
3
3
8

a=3      b=3      c=8

Not a triangle

Process returned 0 (0x0)   execution time : 3.663 s
Press any key to continue.

```

Gambar 2. 3 Hasil pengujian rule 3

```

Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
8
8
8

a=8      b=8      c=8

equilateral triangle

Process returned 0 (0x0)   execution time : 3.055 s
Press any key to continue.

```

Gambar 2. 4 Hasil pengujian rule 4

```

Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
8
8
3

a=8      b=8      c=3

isosceles triangle

```

Gambar 2. 5 Hasil pengujian rule 7

```

Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
8
3
8

a=8      b=3      c=8

isosceles triangle

```

Gambar 2. 6 Hasil pengujian rule 9

```

Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
3
8
8

a=3      b=8      c=8

isosceles triangle

```

Gambar 2. 7 Hasil pengujian rule 10

```

Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
3
4
5

a=3      b=4      c=5

scalene triangle

```

Gambar 2. 8 Hasil pengujian rule 11

2.6 Analisa

1. Pengujian Rule 1

Pada gambar pengujian rule 1, sebagai contoh pengujian rule 1 diberikan nilai $a=8$, $b=3$, $c=3$. Selanjutnya dijalankan sesuai kondisi C1 yaitu $a < b+c$ yang berfungsi untuk mengetahui apakah nilai a , b , c termasuk segitiga atau tidak. $8 < 3+3$ dinyatakan salah karena $8 < 6$. Nilai C1, C2, dan C3 jika dioperasikan dengan operator logika AND maka akan menghasilkan nilai *false*, sehingga variabel *istriangle* akan diisi karakter 'n'. Kemudian program akan mengeksekusi kondisi *else* dan mengeluarkan *output* berupa *Not triangle*.

2. Pengujian Rule 2

Pada gambar pengujian rule 2, sebagai contoh pengujian rule 2 diberikan nilai $a=3$, $b=8$, $c=3$. Selanjutnya dijalankan sesuai kondisi C1 yaitu $a < b+c$ dan C2 $b < a+c$. Pada pengujian ini $3 < 8+3$ atau $3 < 12$ hal ini memenuhi kondisi C1 dan benar. Kemudian melanjutkan ke kondisi C2 yaitu $b < a+c$. Pada pengujian ini $8 < 3+3$ atau $8 < 6$ hal ini tidak memenuhi kondisi C2 atau salah. Nilai C1, C2, dan C3 jika dioperasikan dengan operator logika AND maka akan menghasilkan nilai *false*, sehingga variabel *istriangle* akan diisi karakter 'n'. Kemudian program akan mengeksekusi kondisi *else* dan mengeluarkan *output* berupa *Not triangle*.

3. Pengujian Rule 3

Pada gambar pengujian rule 3, sebagai contoh pengujian rule 3 diberikan nilai $a=3$, $b=3$, $c=8$. Pada pengujian rule 3 harus memenuhi tiga kondisi yaitu C1 $a < b+c$, C2 $b < a+c$, dan C3 $c < a+b$. Selanjutnya dijalankan sesuai kondisi C1 $3 < 3+8$ dimana $3 < 11$ menghasilkan nilai *true* atau terpenuhi, kemudian untuk C2 $3 < 3+8$ dimana $3 < 11$ menghasilkan nilai *true* dan kemudian untuk C3 $8 < 3+3$ dimana $8 < 6$ dan menghasilkan nilai *false* atau tidak terpenuhi. Nilai C1, C2, dan C3 jika dioperasikan dengan operator logika AND maka akan menghasilkan nilai *false*, sehingga variabel *istriangle* akan diisi karakter 'n'. Kemudian program akan mengeksekusi kondisi *else* dan mengeluarkan *output* berupa *Not triangle*.

4. Pengujian Rule 4

Pada pengujian rule 4 ditampilkan nilai variabel $a=8$, $b=8$, $c=8$. Pada C1 $8 < 8+8$ menghasilkan nilai *true* (memenuhi), sedangkan C2 $8 < 8+8$ menghasilkan nilai *true* (memenuhi), C3 $8 < 8+8$ menghasilkan nilai *true* (memenuhi). Nilai C1, C2, dan C3 terpenuhi semua (*true*) jika dioperasikan dengan operator logika AND maka akan menghasilkan nilai benar, sehingga variabel *istriangle* akan diisi karakter 'y'. Sedangkan C4 $8=8$ menghasilkan nilai *true* (memenuhi), C5 $8=8$ menghasilkan nilai *true* (memenuhi), dan C6 $8=8$ menghasilkan nilai *true* (memenuhi). maka kondisi percabangan tersebut terpenuhi semua, dan program akan menampilkan *output* berupa *Equilateral Triangle* (segitiga sama sisi).

5. Pengujian Rule 5

Pada pengujian rule 5 harus memenuhi enam kondisi yaitu C1 $a < b + c$ memenuhi (*true*), C2 $b < a + c$ memenuhi (*true*), C3 $c < a + b$ memenuhi (*true*), C4 $a = b$ memenuhi (*true*), C5 $a = c$ memenuhi (*true*), dan C6 $b = c$ tidak memenuhi (*false*). Karena tidak ada nilai yang memenuhi untuk kondisi C4, C5, C6, dimana nilai b tidak boleh sama dengan nilai c, namun nilai a harus sama dengan b dan nilai a sama dengan nilai c, maka jenis segitiga tersebut adalah *impossible*.

6. Pengujian Rule 6

Pada pengujian rule 6 harus memenuhi enam kondisi yaitu C1 $a < b + c$ memenuhi (*true*), C2 $b < a + c$ memenuhi (*true*), C3 $c < a + b$ memenuhi (*true*), C4 $a = b$ memenuhi (*true*), C5 $a = c$ tidak terpenuhi (*false*), dan C6 $b = c$ memenuhi (*true*). Karena tidak ada nilai yang memenuhi untuk kondisi C4, C5, C6, dimana nilai a tidak boleh sama dengan nilai c, namun nilai a harus sama dengan b dan nilai b sama dengan nilai c, maka jenis segitiga tersebut adalah *impossible*.

7. Pengujian Rule 7

Pada pengujian rule 7 harus memenuhi enam kondisi yaitu C1 $a < b + c$ memenuhi (*true*), C2 $b < a + c$ memenuhi (*true*), C3 $c < a + b$ memenuhi (*true*), C4 $a = b$ memenuhi (*true*), dan untuk C5 $a = c$ tidak terpenuhi (*false*), dan C6 $b = c$ tidak terpenuhi (*false*). maka diberikan nilai $a = 8$, $b = 8$, $c = 3$. Dari kondisi C1 $8 < 8 + 3$ maka bernilai *true*, kondisi C2 $8 < 8 + 3$ maka bernilai *true*, kondisi C3 $3 < 8 + 8$ maka bernilai *true*. Nilai C1, C2, dan C3 terpenuhi semua (*true*) jika dioperasikan dengan operator logika AND maka akan menghasilkan nilai benar, sehingga variabel *istriangle* akan diisi karakter 'y'. Sedangkan kondisi C4 $8 = 8$ maka bernilai *true*, kondisi C5 $8 = 3$ maka bernilai *false* karena $8 \neq 3$, begitu juga dengan C6 $8 = 3$ maka bernilai *false* karena $8 \neq 3$. sehingga program akan menampilkan *output* berupa *Isosceles Triangle* (segitiga sama kaki).

8. Pengujian Rule 8

Pada pengujian rule 8 harus memenuhi enam kondisi C1 $a < b + c$ memenuhi (*true*), C2 $b < a + c$ memenuhi (*true*), C3 $c < a + b$ memenuhi (*true*), C4 $a = b$ tidak terpenuhi (*false*), C5 $a = c$ memenuhi (*true*), dan C6 $b = c$ memenuhi (*true*). Karena tidak ada nilai yang memenuhi untuk kondisi C4, C5, C6, dimana nilai a tidak boleh sama dengan nilai b, namun nilai b harus sama dengan c dan nilai a sama dengan nilai c, maka jenis segitiga tersebut adalah *impossible*.

9. Pengujian Rule 9

Pada pengujian rule 9 harus memenuhi enam kondisi yaitu C1 $a < b + c$ memenuhi (*true*), C2 $b < a + c$ memenuhi (*true*), C3 $c < a + b$ memenuhi (*true*), C4 $a = b$ tidak terpenuhi (*false*), C5 $a = c$ memenuhi (*true*), C6 $b = c$ tidak terpenuhi (*false*). maka nilai yang dapat memenuhi adalah $a = 8$, $b = 3$, $c = 8$. Pada percabangan pertama kondisi C1 $8 < 3 + 8$ adalah *true*, kondisi C2 $3 < 8 + 8$ adalah *true*, dan kondisi C3 $8 < 8 + 3$ adalah *true*. Nilai C1, C2, dan C3 terpenuhi semua (*true*) jika dioperasikan dengan operator logika AND maka akan menghasilkan nilai benar, sehingga variabel *istriangle* akan diisi karakter 'y'. sedangkan untuk C4 $8 = 3$ adalah *false* karena $8 \neq 3$, C5 $8 = 8$ adalah *true*, dan untuk C6 $8 = 3$ adalah bernilai *false* karena $8 \neq 3$. sehingga program akan menampilkan *output* berupa *Isosceles Triangle* (segitiga sama kaki).

10. Pengujian Rule 10

Pada pengujian rule 10 harus memenuhi enam kondisi yaitu C1 $a < b + c$ memenuhi (*true*), C2 $b < a + c$ memenuhi (*true*), C3 $c < a + b$ memenuhi (*true*), C4 $a = b$ tidak terpenuhi (*false*), C5 $a = c$ tidak memenuhi (*false*), C6 $b = c$ memenuhi (*true*). maka nilai yang dapat memenuhi adalah $a=3$, $b=8$, $c=8$. Pada percabangan pertama kondisi C1 $3 < 8+8$ adalah *true*, kondisi C2 $8 < 3+8$ adalah *true*, dan kondisi C3 $8 < 3+8$ adalah *true*. Nilai C1, C2, dan C3 terpenuhi semua *true*, jika dioperasikan dengan operator logika AND maka akan menghasilkan nilai benar, sehingga variabel *istriangle* akan diisi karakter 'y', sedangkan untuk C4 $3=8$ adalah *false* karena $3 \neq 8$, C5 $3=8$ adalah *false* karena $3 \neq 8$, dan untuk C6 $8=8$ terpenuhi atau *true*. sehingga program akan menampilkan *output* berupa *Isosceles Triangle* (segitiga sama kaki).

11. Pengujian Rule 11

Pada pengujian rule 11 harus memenuhi enam kondisi yaitu C1 $a < b + c$ memenuhi (*true*), C2 $b < a + c$ memenuhi (*true*), C3 $c < a + b$ memenuhi (*true*), C4 $a = b$ tidak terpenuhi (*false*), C5 $a = c$ tidak terpenuhi (*false*), dan C6 $b = c$ tidak terpenuhi (*false*). Diberikan nilai variabel $a=3$, $b=4$, dan $c=5$. Pada percabangan pertama diberikan kondisi C1 $3 < 4+5$ adalah *true*, C2 $4 < 3+5$ adalah *true*, dan C3 $5 < 3+4$ adalah *true*, Sehingga nilai C1, C2, dan C3 *true*, *true*, *true* yang dioperasikan dengan operator AND akan menghasilkan nilai benar dan variabel *istriangle* akan diisi karakter 'y'. sedangkan untuk C4 $3=4$ adalah *false* karena $3 \neq 4$, C5 $3=5$ adalah *false* karena $3 \neq 5$, dan untuk C6 $4=5$ tidak terpenuhi atau *false* karena $4 \neq 5$. sehingga program akan menampilkan *output* berupa *Scalene Triangle* (segitiga sembarang).

2.7 Kesimpulan

Dari hasil praktikum pada BAB II di atas, dapat diambil kesimpulan bahwa dalam melakukan pengujian *whitebox* memiliki teknik-teknik untuk pengujian kondisi untuk mengetahui cara kerja internal suatu perangkat lunak untuk menjamin operasi-operasi internal sesuai dengan spesifikasi yang telah ditetapkan dengan menggunakan struktur yang telah dirancang.

Dan pada BAB II ini menggunakan kondisi kombinasi dari pengujian cabang dan pengujian domain. yaitu menguji perhitungan aritmatika dan nilai *boolean* untuk mendapatkan *output* jenis segitiga.

BAB III

PENGUJIAN *BLACKBOX* (1): PENGUJIAN ANALISIS KELAS EKUIVALENSI

3.1 Tujuan

1. Praktikan dapat mempersiapkan tahapan pengujian
2. Praktikan dapat merancang kasus uji
3. Praktikan dapat melakukan pengujian perangkat lunak dengan Teknik Analisis Kelas Ekuivalensi.

3.2 Alat dan Bahan

Alat dan bahan yang dibutuhkan dalam modul BAB III ini adalah sebuah komputer yang sudah ter-install IDE (*Integrated Development Environment*) “CodeBlocks” yang memiliki compiler C.

3.3 Dasar Teori

3.3.1 *Blackbox Testing*

Blackbox testing memfokuskan pada kebutuhan fungsional dari *software*. Hal ini berarti bahwa pengujian ini memperbolehkan *software engineer* menurunkan sejumlah *input* yang ditujukan untuk menguji kebutuhan fungsional dari program tersebut. Pengujian ini berusaha menemukan *error* dengan kategori sebagai berikut : (Lutfi, 2018)

- Fungsi yang salah atau hilang.
- Kesalahan antarmuka, struktur data atau pengaksesan data eksternal, unjuk kerja, inisialisasi dan penghentian.

Tidak seperti pengujian *whitebox* yang dilakukan pada awal proses, pengujian *blackbox* diterapkan pada akhir tahapan proses pengujian. Hal ini dikarenakan pengujian ini tidak mementingkan struktur kontrol tapi lebih memfokuskan pada *domain* informasi.

3.3.2 Analisis Partisi Kelas Ekuivalensi

Adalah metoda pengujian *blackbox* yang membagi *input domain* dari program ke dalam kelas-kelas data dimana kasus uji bisa diturunkan. *Equivalence Partitioning* didasarkan pada evaluasi persamaan kelas dari *input condition*. Sebuah persamaan kelas menunjukkan sekumpulan keadaan valid atau tidak valid untuk syarat/ kondisi masukan yang umumnya adalah nilai numerik tertentu, sebuah jangkauan nilai (*range value*), sebuah himpunan nilai-nilai yang berkaitan, atau kondisi *boolean*. Persamaan kelas bisa didefinisikan menurut panduan sebagai berikut :

1. Jika kondisi *input* adalah sebuah *range*, satu kelas persamaan dan dua kelas persamaan didefinisikan.
2. Jika sebuah kondisi *input* memerlukan sebuah nilai khusus, satu kelas persamaan dan dua kelas persamaan didefinisikan.
3. Jika sebuah kondisi *input* adalah sebuah anggota himpunan, satu kelas persamaan yang valid dan satu kelas persamaan yang tidak valid didefinisikan.
4. Jika kondisi *input* adalah *boolean*, satu kelas yang valid dan satu kelas yang tidak valid didefinisikan.

Contoh:

Pemeliharaan data untuk aplikasi bank yang sudah diotomatiskan. Pemakai dapat memutar nomor telepon bank dengan menggunakan mikro komputer yang terhubung dengan *password* yang telah ditentukan dan diikuti dengan perintah-perintah. Data yang diterima adalah:

Kode area	: kosong atau tiga digit
<i>Prefix</i>	: 3 digit atau tidak diawali 0 atau 1
<i>Suffix</i>	: 4 digit
<i>Password</i>	: 6 digit alfanumerik
Perintah	: <i>check</i> , deposit, dan lain-lain

Selanjutnya kondisi input digabungkan dengan masing-masing data elemen dapat ditentukan sebagai berikut:

Kode area	:	Kondisi <i>input, boolean</i> – kode area mungkin ada atau tidak. Kondisi <i>input, range</i> – nilai ditentukan antara 200 dan 999.
<i>Prefix</i>	:	Kondisi <i>input range</i> > 200 atau tidak diawali 0 atau 1.
<i>Suffix</i>	:	Kondisi <i>input</i> 4 digit.
<i>Password</i>	:	Kondisi <i>input boolean</i> – <i>password</i> mungkin dibutuhkan atau tidak. Kondisi <i>input</i> nilai dengan 6 karakter string.
Perintah	:	Kondisi <i>input set</i> berisi perintah-perintah yang telah didefinisikan.

3.4 Prosedur Praktikum

Berikut ini adalah kode program dalam bahasa C untuk masalah segitiga yaitu menerima tiga *integer* yang dianggap sebagai tiga sisi segitiga dan menentukan apakah ketiga nilai tersebut merepresentasikan segitiga sama sisi, segitiga sama kaki, segitiga yang tidak sama semua sisinya atau tidak membentuk segitiga.

1. Turunkan *test case* berdasarkan partisi kelas ekuivalensi!
2. Jalankan *test case* tersebut!
3. Analisis hasil pengujian tersebut!

Program segitiga

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("Nama\t: Nur Muhammad Syaifuddin\n");
```

```

printf("NIM\t: 32601900026\n");
int a,b,c,c1,c2,c3;
char istriangle;
do
{
printf("\n\enter 3 integers which are sides of triangle\n");
scanf("%d%d%d",&a,&b,&c);
printf("\na=%d\tb=%d\tc=%d\n\n",a,b,c);
c1 = a>=1 && a<=10;
c2= b>=1 && b<=10;
c3= c>=1 && c<=10;
if (!c1)
printf("\nthe value of a=%d is not the range of permitted
value",a);
if (!c2)
printf("\nthe value of b=%d is not the range of permitted
value",b);
if (!c3)
printf("\nthe value of c=%d is not the range of permitted
value",c);
} while(!(c1 && c2 && c3));
// Mengecek segitiga atau tidak
if( a<b+c && b<a+c && c<a+b )
istriangle='y';
else
istriangle ='n';
if (istriangle=='y')
if ((a==b) && (b==c))
printf("equilateral triangle\n");
else if ((a!=b) && (a!=c) && (b!=c))
printf("scalene triangle\n");
else
printf("isosceles triangle\n");
else
printf("Not a triangle\n");
printf("\n");
return 0;
}

```


Test Case

<i>Test data</i>	:	Memasukan 3 nilai <i>integer</i> (a, b, dan c)
<i>Pre-condition</i>	:	$1 \leq a \leq 10$, $1 \leq b \leq 10$ dan $1 \leq c \leq 10$ dan $a < b + c$, $b < a + c$ dan $c < a + b$
<i>Deskripsi singkat</i>	:	Mengecek apakah nilai merupakan segitiga sama sisi, segitiga sama kaki, segitiga yang tidak sama semua sisinya atau tidak membentuk segitiga.

Tabel 3. 1 *Weak equivalence class testing*

<i>Weak Equivalence Class Testing</i>							
<i>Case Id</i>	<i>Description</i>	<i>Input Data</i>			<i>Expected Output</i>	<i>Actual Output</i>	<i>Status</i>
		a	b	c			
1	Enter the normal value for a, b, and c	5	5	5	Should display the message equilateral triangle	Equilateral triangle	Pass
2	Enter the normal value for a, b, and c	2	2	3	Should display the message isosceles triangle	Isosceles triangle	Pass
3	Enter the normal value for a, b, and c	3	4	5	Should display the message scalene triangle	Scalene triangle	Pass

4	Enter the normal value for a, b, and c	4	1	2	Message should be displayed can't form a triangle	not a triangle	Pass	Sesuai yang diharapkan
---	--	---	---	---	---	----------------	------	------------------------

Tabel 3. 2 Weak robust equivalence class testing

Weak Robust Equivalence Class Testing								
Case Id	Description	Input Data			Expected Output	Actual Output	Status	Comments
5	Enter one invalid input (min) value and two valid value for a, b and c	1			Should display value of a is not in the range of permitted values	Not the range of permitted value	Pass	Sesuai yang diharapkan
6	Enter one invalid input (min) value and two valid value for a, b and c		1		Should display value of b is not in the range of permitted values	Not the range of permitted value	Pass	Sesuai yang diharapkan

7	<i>Enter one invalid input (min) value and two valid value for a, b and c</i>			1	<i>Should display value of c is not in the range of permitted values</i>	<i>Not the range of permitted value</i>	<i>pass</i>	Sesuai yang diharapkan
8	<i>Enter one invalid input (max) value and two valid value for a, b and c</i>	1			<i>Should display value of a is not in the range of permitted values</i>	<i>Not the range of permitted value</i>	<i>pass</i>	Sesuai yang diharapkan
9	<i>Enter one invalid input (max) value and two valid value for a, b and c</i>			1	<i>Should display value of b is not in the range of permitted values</i>	<i>Not the range of permitted value</i>	<i>pass</i>	Sesuai yang diharapkan

01	Enter one invalid input (max) value and two valid value for a, b and c			1	Should display value of c is not in the range of permitted values	Not the range of permitted value	pass	Sesuai yang diharapkan
----	--	--	--	---	---	----------------------------------	------	------------------------

Tabel 3. 3 Strong robust equivalence class testing

Strong Robust Equivalence Class Testing								
Case Id	Description	Input Data			Expected Output	Actual Output	Status	Comments
11	Enter two invalid input (min) and two valid value for a, b and c	1			Should display value of a is not the range of permitted values	Not the range of permitted value	pass	Sesuai yang diharapkan
12	Enter two invalid input (min) and two valid value for a, b and c.		1		Should display value of b is not the range of	Not the range of permitted value	pass	Sesuai yang diharapkan

					<i>permitted values</i>			
1 3	<i>Enter two invalid input (min) and two valid value for a, b and c.</i>			1	<i>Should display value of c is not the range of permitted values</i>	<i>Not the range of permitted value</i>	<i>pass</i>	Sesuai yang diharapkan
1 4	<i>Enter all invalid inputs (min)</i>	1	1		<i>Should display value of a is not the range of permitted values</i>	<i>Not the range of permitted value</i>	<i>pass</i>	Sesuai yang diharapkan
					<i>Should display value of b is not the range of permitted values</i>	<i>Not the range of permitted value</i>	<i>pass</i>	Sesuai yang diharapkan

5	1	Enter two invalid input (max) and two valid value for a, b and c	1	-1	Should display value of b is not the range of permitted values	Not the range of permitted value	pass	Sesuai yang diharapkan
					Should display value of c is not the range of permitted values	Not the range of permitted value	pass	Sesuai yang diharapkan
6	1	Enter two invalid input (max) and two valid value for a, b and c	1	1	Should display value of a is not the range of permitted values	Not the range of permitted value	pass	Sesuai yang diharapkan
					Should display value of c is not the range of permitted values	Not the range of permitted value	pass	Sesuai yang diharapkan

					<i>permitted values</i>			
1 7	<i>Enter two invalid input (max) and two valid value for a, b and c</i>	1	1	1	<i>Should display value of a is not the range of permitted values</i>	<i>Not the range of permitted value</i>	<i>pass</i>	Sesuai yang diharapkan
					<i>Should display value of b is not the range of permitted values</i>	<i>Not the range of permitted value</i>	<i>pass</i>	Sesuai yang diharapkan
					<i>Should display value of c is not the range of permitted values</i>	<i>Not the range of permitted value</i>	<i>pass</i>	Sesuai yang diharapkan

3.5 Hasil Praktikum

1. Hasil *test case 1*

```

Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
5
5
5

a=5      b=5      c=5

equilateral triangle

Process returned 0 (0x0)   execution time : 8.159 s
Press any key to continue.

```

Gambar 3. 1 Hasil *test case 1 blackbox equivalence class*

2. Hasil *test case 2*

```

Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
2
2
3

a=2      b=2      c=3

isosceles triangle

Process returned 0 (0x0)   execution time : 7.418 s
Press any key to continue.

```

Gambar 3. 2 Hasil *test case 2 blackbox equivalence class*

3. Hasil *test case 3*

```

Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
3
4
5

a=3      b=4      c=5

scalene triangle

Process returned 0 (0x0)   execution time : 7.043 s
Press any key to continue.

```

Gambar 3. 3 Hasil *test case 3 blackbox equivalence class*

4. Hasil *test case 4*

```

Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
4
1
2

a=4      b=1      c=2

Not a triangle

Process returned 0 (0x0)   execution time : 7.419 s
Press any key to continue.

```

Gambar 3. 4 Hasil *test case 4 blackbox equivalence class*5. Hasil *test case 5*

```

Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
-1
5
5

a=-1     b=5      c=5

the value of a=-1 is not the range of permitted value

```

Gambar 3. 5 Hasil *test case 5 blackbox equivalence class*6. Hasil *test case 6*

```

Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
5
-1
5

a=5      b=-1     c=5

the value of b=-1 is not the range of permitted value

```

Gambar 3. 6 Hasil *test case 6 blackbox equivalence class*

7. Hasil *test case* 7

```
Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
5
5
-1

a=5      b=5      c=-1

the value of c=-1 is not the range of permitted value
```

Gambar 3. 7 Hasil *test case* 7 *blackbox equivalence class*8. Hasil *test case* 8

```
Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
11
5
5

a=11     b=5      c=5

the value of a=11 is not the range of permitted value
```

Gambar 3. 8 Hasil *test case* 8 *blackbox equivalence class*9. Hasil *test case* 9

```
Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
5
11
5

a=5      b=11     c=5

the value of b=11 is not the range of permitted value
```

Gambar 3. 9 Hasil *test case* 9 *blackbox equivalence class*

10. Hasil *test case* 10

```
Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
5
5
11

a=5      b=5      c=11

the value of c=11 is not the range of permitted value
```

Gambar 3. 10 Hasil *test case* 10 *blackbox equivalence class*11. Hasil *test case* 11

```
Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
-1
5
5

a=-1     b=5      c=5

the value of a=-1 is not the range of permitted value
```

Gambar 3. 11 Hasil *test case* 11 *blackbox equivalence class*12. Hasil *test case* 12

```
Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
5
-1
5

a=5      b=-1     c=5

the value of b=-1 is not the range of permitted value
```

Gambar 3. 12 Hasil *test case* 12 *blackbox equivalence class*

13. Hasil *test case* 13

```

Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
5
5
-1

a=5      b=5      c=-1

the value of c=-1 is not the range of permitted value

```

Gambar 3. 13 Hasil *test case* 13 *blackbox equivalence class*14. Hasil *test case* 14

```

Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
-1
-1
5

a=-1     b=-1     c=5

the value of a=-1 is not the range of permitted value
the value of b=-1 is not the range of permitted value

```

Gambar 3. 14 Hasil *test case* 14 *blackbox equivalence class*15. Hasil *test case* 15

```

Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
5
-1
-1

a=5      b=-1     c=-1

the value of b=-1 is not the range of permitted value
the value of c=-1 is not the range of permitted value

```

Gambar 3. 15 Hasil *test case* 15 *blackbox equivalence class*

16. Hasil *test case* 16

```

Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
-1
5
-1

a=-1      b=5      c=-1

the value of a=-1 is not the range of permitted value
the value of c=-1 is not the range of permitted value

```

Gambar 3. 16 Hasil *test case* 16 *blackbox equivalence class*

17. Hasil *test case* 17

```

Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
-1
-1
-1

a=-1      b=-1      c=-1

the value of a=-1 is not the range of permitted value
the value of b=-1 is not the range of permitted value
the value of c=-1 is not the range of permitted value

```

Gambar 3. 17 Hasil *test case* 17 *blackbox equivalence class*

3.6 Analisa

1. *Test case* 1

Pada *test case* pertama ditampilkan variabel $a = 5$, $b = 5$, $c = 5$. Nilai tersebut masih dalam *range* yang telah ditetapkan dan dikarenakan nilai tersebut sama semua, maka ditampilkan *output* yang berupa jenis segitiga *Equilateral Triangle*.

2. *Test case* 2

Pada *test case* ke-2 ditampilkan variabel $a = 2$, $b = 2$, $c = 3$. Nilai ini masih dalam *range* yang telah ditentukan dan akan menampilkan *output* *Isosceles Triangle*.

3. *Test case 3*

Pada *test case* ke-3 ditampilkan variabel $a = 3$, $b = 4$, $c = 5$. Nilai ini masih dalam *range* yang telah ditentukan pada program dan output yang ditampilkan adalah *Scalene Triangle*.

4. *Test case 4*

Pada *test case* ke-4 ditampilkan nilai variabel $a = 4$, $b = 1$, $c = 2$. Nilai ini masih dalam *range/* jangkauan yang telah ditentukan pada program. Namun karena $a > b + c$, sehingga tidak membentuk segitiga dan *output* yang keluar adalah *Not a Ttriangle*.

5. *Test case 5*

Pada *test case* ke-5 ditampilkan nilai variabel $a = -1$, $b = 5$, $c = 5$, dikarenakan salah satu nilai ada yang diluar jangkauan yaitu -1 maka tidak terbentuk segitiga dan *output* yang dikeluarkan adalah *the value of $c = -1$ is not the range of permitted value*.

6. *Test case 6*

Pada *test case* ke-6 ditampilkan nilai variabel $a = 5$, $b = -1$, $c = 5$, dikarenakan salah satu nilai ada yang diluar jangkauan yaitu -1, dan tidak terbentuk segitiga maka *output* yang dikeluarkan adalah *the value of $b = -1$ is not the range of permitted value*.

7. Test case 7

Pada *test case* ke-7 ditampilkan nilai variabel $a = 5$, $b = 5$, $c = -1$, dikarenakan salah satu nilai ada yang diluar jangkauan yaitu -1 maka tidak terbentuk segitiga dan *output* yang dikeluarkan adalah *the value of $c=-1$ is not the range of permitted value.*

8. Test case 8

Pada *test case* ke-8 ditampilkan nilai variabel $a = 11$, $b = 5$, $c = 5$, pada kasus terdapat satu variabel yang tidak memenuhi *range* yaitu 11 , karena pada program hanya menampung angka yang *range*-nya ≥ 1 dan ≤ 10 . Maka, *output* yang dikeluarkan adalah *the value of $a=11$ is not the range of permitted value.*

9. Test case 9

Pada *test case* ke- 9 ditampilkan nilai variabel $a = 5$, $b = 11$, $c = 5$, pada kasus ini terdapat satu nilai variabel yang melebihi *range* yang telah ditetapkan pada program yaitu 11 , karena pada program hanya dapat menampung nilai hingga ≥ 1 dan ≤ 10 . Maka *output* yang ditampilkan adalah *the value of $b=11$ is not the range of permitted value*

10. Test case 10

Pada *test case* ke-10 ditampilkan nilai variabel $a = 5$, $b = 5$, $c = 11$, pada kasus ini terdapat salah satu nilai variabel yang melebihi *range/* jangkauan yang telah ditetapkan pada program yaitu 11 , karena pada program hanya dapat menampung nilai hingga ≥ 1 dan ≤ 10 , maka *output* yang dihasilkan adalah *the value of $c=11$ is not the range of permitted value.*

11. Test case 11

Pada *test case* ke-11 ditampilkan nilai variabel $a = -1$, $b = 5$, $c = 5$ dalam kasus ini terdapat satu variabel yang tidak memenuhi *range/* jangkauan yang telah ditetapkan pada program yaitu -1 . Pada program nilai variabel harus ≥ 1 dan ≤ 10 , maka *output* yang ditampilkan adalah *the value of $b=-1$ is not the range of permitted value.*

12. Test case 12

Pada *test case* ke-12 ditampilkan nilai variabel $a = 5$, $b = -1$, $c = 5$, dalam kasus ini terdapat satu variabel yang tidak memenuhi *range/* jangkauan yang telah ditetapkan pada program yaitu -1. Pada program nilai variabel harus ≥ 1 dan ≤ 10 , maka *output* yang ditampilkan adalah *the value of $b = -1$ is not the range of permitted value.*

13. Test case 13

Pada *test case* ke-13 ditampilkan nilai variabel $a = 5$, $b = 5$, $c = -1$, dalam kasus ini salah satu variabel tidak memenuhi *range/* jangkauan yang telah ditetapkan pada program, yaitu -1. Pada program nilai variabel harus ≥ 1 dan ≤ 10 , maka *output* yang ditampilkan adalah *the value of $b = -1$ is not the range of permitted value.*

14. Test case 14

Pada *test case* ke-14 ditampilkan nilai variabel $a = -1$, $b = -1$, $c = 5$. Pada kasus ini, terdapat 2 variabel yang tidak memenuhi *range/* jangkauan yaitu $a = -1$ dan $b = -1$. Pada program nilai variabel harus ≥ 1 dan ≤ 10 , maka *output* yang ditampilkan adalah *the value of $b = -1$ is not the range of permitted value* dan *the value of $b = -1$ is not the range of permitted value.*

15. Test case 15

Pada *test case* ke-15 ditampilkan nilai variabel $a = 5$, $b = -1$, $c = -1$. Pada kasus ini, terdapat 2 variabel yang tidak memenuhi *range/* jangkauan yaitu $b = -1$ dan $c = -1$. Pada program nilai variabel harus ≥ 1 dan ≤ 10 , maka *output* yang ditampilkan adalah *the value of $b = -1$ is not the range of permitted value* dan *the value of $b = -1$ is not the range of permitted value.*

16. Test case 16

Pada *test case* ke-16 ditampilkan nilai variabel $a = -1$, $b = 5$, $c = -1$. Pada kasus ini, terdapat 2 variabel yang tidak memenuhi *range/*jangkauan yaitu $a = -1$ dan $c = -1$. Pada program nilai variabel harus ≥ 1 dan ≤ 10 , maka *output* yang ditampilkan adalah *the value of $b = -1$ is not the range of permitted value* dan *the value of $b = -1$ is not the range of permitted value*.

17. Test case 17

Pada *test case* ke-17 ditampilkan nilai variabel $a = -1$, $b = -1$, $c = -1$. Pada kasus ini, terdapat 3 variabel yang tidak memenuhi *range/*jangkauan. Pada program nilai variabel harus ≥ 1 dan ≤ 10 , maka *output* yang ditampilkan adalah *the value of $b = -1$ is not the range of permitted value* dan *the value of $b = -1$ is not the range of permitted value* dan *the value of $b = -1$ is not the range of permitted value*.

3.7 Kesimpulan

Dari hasil praktikum BAB III ini, dapat diambil kesimpulan bahwa pengujian *blackbox* adalah pengujian yang didasarkan pada pengecekan atau pengujian terhadap detail perancangan dari perangkat lunak. *Blackbox* sendiri mempunyai strategi sistem yaitu *equivalence class testing* yang meliputi beberapa nilai yaitu nilai minimum variabel *input*, nilai di bawah nilai minimum, nilai normal, nilai di atas nilai maksimum dan nilai maksimum. Jika nilai termasuk dalam jangkauan maka akan menampilkan jenis segitiganya, kemudian jika nilai yang dimasukan kurang dari nilai minimum dan lebih dari nilai maksimum maka program akan mengeluarkan *output* bahwa nilai tersebut berada di luar *range/*jangkauan.

BAB IV

PENGUJIAN *BLACKBOX* (2): PENGUJIAN ANALISIS NILAI BATAS

4.1 Tujuan

1. Praktikan dapat mempersiapkan tahapan pengujian
2. Praktikan dapat merancang kasus uji
3. Praktikan dapat melakukan pengujian perangkat lunak dengan teknik Analisis Nilai Batas

4.2 Alat dan Bahan

Alat dan bahan yang dibutuhkan dalam modul BAB III ini adalah sebuah komputer yang sudah ter-install IDE (Integrated Development Environment) “CodeBlocks” yang memiliki compiler C.

4.3 Dasar Teori

4.3.1 Analisis Nilai Batas (*Boundary Value Analysis* atau BVA)

Teknik Analisis Nilai Batas ini dilakukan karena adanya fenomena bahwa kesalahan sering terjadi pada daerah batas dari suatu *input*. Teknik ini tidak hanya memperhatikan batas suatu nilai *input* tapi juga memperhatikan batas nilai *output*.

Untuk permasalahan yang tidak diketahui dengan jelas, cenderung menimbulkan kesalahan pada *domain output*-nya. BVA merupakan pilihan *test case* yang mengerjakan nilai yang telah ditentukan, dengan teknik perancangan *test case* melengkapi *test case equivalence partitioning* yang fokusnya pada *domain input*. BVA fokusnya pada *domain output*.

4.3.2 Petunjuk Pengujian BVA

1. Jika kondisi *input* berupa *range* yang dibatasi nilai *a* dan *b*, *test case* harus dirancang dengan nilai *a* dan *b*.
2. Jika kondisi *input* ditentukan dengan sejumlah nilai, *test case* harus dikembangkan dengan mengerjakan sampai batas maksimal nilai tersebut.
3. Sesuai petunjuk 1 dan 2 untuk kondisi *output test case* dirancang sampai jumlah maksimal.
4. Untuk struktur data pada program harus dirancang sampai batas kemampuan.

4.4 Prosedur Praktikum

Berikut ini adalah kode program dalam bahasa C untuk masalah segitiga yaitu menerima tiga *integer* yang dianggap sebagai tiga sisi segitiga dan menentukan apakah ketiga nilai tersebut merepresentasikan segitiga sama sisi, segitiga sama kaki, segitiga yang tidak sama semua sisinya atau tidak membentuk segitiga.

1. Turunkan *test case*-nya!
2. Jalankan *test case* tersebut!
3. Analisis hasil pengujian tersebut!

Program segitiga

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("Nama\t: Nur Muhammad Syaifuddin\n");
    printf("NIM\t: 32601900026\n");
    int a,b,c,c1,c2,c3;
    char istriangle;
    do
    {
```

```

printf("\n\nenter 3 integers which are sides of triangle\n");
scanf("%d%d%d",&a,&b,&c);
printf("\na=%d\tb=%d\tc=%d\n\n",a,b,c);
c1 = a>=1 && a<=10;
c2= b>=1 && b<=10;
c3= c>=1 && c<=10;
if (!c1)
printf("\nthe value of a=%d is not the range of permitted
value",a);
if (!c2)
printf("\nthe value of b=%d is not the range of permitted
value",b);
if (!c3)
printf("\nthe value of c=%d is not the range of permitted
value",c);
} while(!(c1 && c2 && c3));
// Mengecek segitiga atau tidak
if( a<b+c && b<a+c && c<a+b )
istriangle='y';
else
istriangle ='n';
if (istriangle=='y')
if ((a==b) && (b==c))
printf("equilateral triangle\n");
else if ((a!=b) && (a!=c) && (b!=c))
printf("scalene triangle\n");
else
printf("isosceles triangle\n");
else
printf("Not a triangle\n");

printf("\n");
return 0;
}

```

Test Case

Test data : Memasukan 3 nilai *integer* (a, b, dan c)
Pre-condition : $1 \leq a \leq 10$, $1 \leq b \leq 10$ dan $1 \leq c \leq 10$ dan $a < b + c$, $b < a + c$ dan $c < a + b$
Deskripsi singkat : Mengecek apakah nilai merupakan segitiga sama sisi, segitiga sama kaki, segitiga yang tidak sama semua sisinya atau tidak membentuk segitiga.

Tabel 4. 1 *Test case boundry value anlysis*

<i>C ase Id</i>	<i>Descript ion</i>	<i>In put Data</i>			<i>Expece d Output</i>	<i>Actu al Output</i>	<i>St atus</i>	<i>Comm ents</i>
1	<i>Enter the min value for a, b, c</i>				<i>Should display the message equilateral triangle</i>	<i>Equi la-teral triangle</i>	<i>P ass</i>	Sesuai yang diharapkan
2	<i>Enter the min value for 2 items and min +1 for any one item1</i>				<i>Messag e should be displayed can't form a triangle</i>	<i>Not a triangle</i>	<i>P ass</i>	Sesuai yang diharapkan

3	<i>Enter the min value for 2 items and min +1 for any one item1</i>			<i>Message should be displayed can't form a triangle</i>	<i>Not a triangle</i>	<i>P_{ass}</i>	Sesuai yang diharapkan
4	<i>Enter the min value for 2 items and min +1 for any one item1</i>			<i>Message should be displayed can't form a triangle</i>	<i>Not a triangle</i>	<i>P_{ass}</i>	Sesuai yang diharapkan
5	<i>Enter the normal value for 2 items and 1 item is min value</i>			<i>Should display the message isosceles triangle</i>	<i>Iso-sceles triangle</i>	<i>P_{ass}</i>	Sesuai yang diharapkan
6	<i>Enter the normal value for 2 items and 1 item is min value</i>			<i>Should display the message isosceles triangle</i>	<i>Iso-sceles triangle</i>	<i>P_{ass}</i>	Sesuai yang diharapkan

7	<i>Enter the normal value for 2 items and 1 item is min value</i>			<i>Should display the message isosceles triangle</i>	<i>Iso-sceles triangle</i>	<i>P_{ass}</i>	Sesuai yang diharapkan
8	<i>Enter the normal value for a, b, and c</i>			<i>Shoud display the message equilateral triangle</i>	<i>Eq uila-teral triangle</i>	<i>P_{ass}</i>	Sesuai yang diharapkan

4.5 Hasil Praktikum

1. Hasil test case 1

```

Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
1
1
1

a=1      b=1      c=1

equilateral triangle

```

Gambar 4. 1 Hasil test case 1 blackbox boundary analysis value

2. Hasil *test case* 2

```
Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
1
1
2

a=1      b=1      c=2

Not a triangle
```

Gambar 4. 2 Hasil *test case* 2 *blackbox boundary analysis value*3. Hasil *test case* 3

```
Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
1
2
1

a=1      b=2      c=1

Not a triangle
```

Gambar 4. 3 Hasil *test case* 3 *blackbox boundary analysis value*4. Hasil *test case* 4

```
Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
2
1
1

a=2      b=1      c=1

Not a triangle
```

Gambar 4. 4 Hasil *test case* 4 *blackbox boundary analysis value*

5. Hasil *test case* 5

```
Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
5
5
1

a=5      b=5      c=1

isosceles triangle
```

Gambar 4. 5 Hasil *test case* 5 *blackbox boundary analysis value*6. Hasil *test case* 6

```
Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
5
1
5

a=5      b=1      c=5

isosceles triangle
```

Gambar 4. 6 Hasil *test case* 6 *blackbox boundary analysis value*7. Hasil *test case* 7

```
Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
1
5
5

a=1      b=5      c=5

isosceles triangle
```

Gambar 4. 7 Hasil *test case* 7 *blackbox boundary analysis value*

8. Hasil *test case* 8

```
Nama      : Nur Muhammad Syaifuddin
NIM       : 32601900026

enter 3 integers which are sides of triangle
5
5
5

a=5      b=5      c=5

equilateral triangle
```

Gambar 4. 8 Hasil *test case* 8 *blackbox boundary analysis value*

4.6 Analisa

1. *Test case* 1

Pada *test case* ke-1 menampilkan variabel dengan nilai $a = 1$, $b = 1$, dan $c = 1$ sesuai dengan modul. Karena nilai yang ditampilkan sesuai dengan jangkauan yang terdapat pada program dan semua sisinya memiliki panjang yang sama maka *output* yang ditampilkan adalah *Equilateral Triangle*.

2. *Test case* 2

Pada *test case* ke-2 menampilkan variabel dengan nilai $a = 1$, $b = 1$, dan $c = 2$ sesuai dengan modul. Variabel tersebut sudah dapat memenuhi aturan yang terdapat pada program yaitu memenuhi jangkauan, namun nilai-nilai itu tidak memungkinkan untuk terbuat menjadi segitiga karena tidak mungkin satu sisi sama panjang dengan jumlah dua sisi lainnya (1 dan 1).

3. *Test case 3*

Pada *test case* ke-3 menampilkan variabel dengan nilai $a = 1$, $b = 2$, dan $c = 1$ sesuai dengan modul. Variabel tersebut sudah dapat memenuhi aturan yang terdapat pada program yaitu memenuhi jangkauan, namun nilai-nilai itu tidak memungkinkan untuk terbuat menjadi segitiga karena tidak mungkin satu sisi sama panjang dengan jumlah dua sisi lainnya (1 dan 1).

4. *Test case 4*

Pada *test case* ke-3 menampilkan variabel dengan nilai $a = 2$, $b = 1$, dan $c = 1$ sesuai dengan modul. Variabel tersebut sudah dapat memenuhi aturan yang terdapat pada program yaitu memenuhi jangkauan, namun nilai-nilai itu tidak memungkinkan untuk terbuat menjadi segitiga karena tidak mungkin satu sisi sama panjang dengan jumlah dua sisi lainnya (1 dan 1).

5. *Test case 5*

Pada *test case* ke-5 menampilkan variabel dengan nilai $a = 5$, $b = 5$, dan $c = 1$ sesuai dengan modul. Variabel tersebut sudah dapat memenuhi aturan yang terdapat pada program yaitu memenuhi jangkauan, dan karena 2 nilai variabel sama, dan satu sisinya lebih kecil dari penjumlahan 2 sisi lain maka *output* yang ditampilkan *Isosceles Triangle*.

6. *Test case 6*

Pada *test case* ke-6 menampilkan variabel dengan nilai $a = 5$, $b = 1$, dan $c = 5$ sesuai dengan modul. Variabel tersebut sudah dapat memenuhi aturan yang terdapat pada program yaitu memenuhi jangkauan, dan karena 2 nilai variabel sama, dan satu sisinya lebih kecil dari penjumlahan 2 sisi lain maka *output* yang ditampilkan *Isosceles Triangle*.

7. *Test case 7*

Pada *test case* ke-7 menampilkan variabel dengan nilai $a = 1$, $b = 5$, dan $c = 5$ sesuai dengan modul. Variabel tersebut sudah dapat memenuhi aturan yang terdapat pada program yaitu memenuhi jangkauan, dan karena 2 nilai variabel sama, dan satu sisinya lebih kecil dari penjumlahan 2 sisi lain maka *output* yang ditampilkan *Isosceles Triangle*.

8. *Test case 8*

Pada *test case* ke-8 menampilkan variabel dengan nilai $a = 5$, $b = 5$, dan $c = 5$ sesuai dengan modul. Karena nilai yang ditampilkan sesuai dengan jangkauan yang terdapat pada program dan semua sisinya memiliki panjang yang sama maka *output* yang ditampilkan adalah *Equilateral Triangle*.

4.7 Kesimpulan

Dari hasil praktikum BAB IV ini, dapat diambil kesimpulan bahwa pengujian Analisis Nilai batas akan menguji program dengan memasukan nilai-nilai di sekitar nilai batas, sehingga akan menguji apakah program sesuai dengan kebutuhan sistem atau tidak. Jika nilai yang dimasukan berupa nilai normal, nilai minimum, nilai maksimum, dan 1 nilai variabel lebih kecil dari jumlah 2 nilai variabel lainnya, maka akan mengeluarkan *output* berupa (segitiga).

BAB V

PENGUJIAN FUNGSIONAL DENGAN SELENIUM IDE

5.1 Tujuan

1. Praktikan dapat memahami pendekatan pengujian terotomasi
2. Praktikan dapat menggunakan Selenium IDE untuk membuat *test suite* yang berisi minimal 2 *test case*
3. Praktikan dapat membuat *test suite* untuk dua *website*

5.2 Alat dan Bahan

1. Komputer *desktop*
2. *Software* Selenium IDE
3. Aplikasi *web*

5.3 Dasar Teori

Otomatisasi dimaksudkan sebagai otomatisasi untuk menghilangkan intervensi manusia, yaitu merupakan proses yang *self-controlling* atau *self-moving*. Perangkat lunak otomasi menawarkan *wizard* dan perintah otomatis selain mempunyai kemampuan merekam tugas dan *re-play*. Menggunakan program ini dapat melakukan rekaman tugas IT ataupun bisnis. Keuntungannya adalah cepat, handal, *repeatable*, *programmable*, *reusable*, dan lain-lain. (Jaya, 2019)

5.3.1 Pengenalan Selenium

Selenium dibuat pada tahun 2004 oleh Jason R. Huggins dan timnya. Nama asalnya adalah *JavaScript Functional Tester* (JSFT). Merupakan *framework* pengujian berbasis *browser open source* yang awalnya dibangun oleh Thoughtworks, dengan fitur-fitur berikut:

1. 100% JavaScript dan HTML
2. *Web testing tool*
3. Mendukung pengujian aplikasi *web 2.0*
4. Mendukung *cross-browser testing (on multiple browsers)*

5. Mendukung berbagai sistem operasi
6. *Cross-browser* – IE 6/7, Firefox 8+, Chrome. Opera, Safari 2.0+
7. Pengujian dapat berjalan pada *browser* secara langsung
8. Selenium dapat di-*deploy* pada Windows, Linux, dan Macintosh
9. Diimplementasikan dengan teknologi *browser*
 - a. JavaScript
 - b. DHTML
 - c. Frames

Komponen Selenium terdiri atas :

1. Selenium IDE
2. Selenium Core
3. Selenium RC
4. Selenium GRID

5.3.2 Selenium IDE

Selenium IDE (*Integrated Development Environment*) merupakan *tool* yang digunakan untuk membuat Selenium *test case*, dengan fitur sebagai berikut:

1. Merupakan Chrome *plugin*
2. Ekstensi Chrome yang memungkinkan paradigma *record/play*
3. Otomasi perintah, tetapi asersi dimasukan secara manual
4. Membuat *locator* sesimpel mungkin
5. Berbasis *selences* (*set perintah Selenium*)

Selenium IDE terdiri dari :

1. Jendela *test case*
2. *Toolbar*
3. Menu bar
4. Jendela *log/reference/UI-Element/Rollup*

A. Jendela *test case* atau *test case pane* :

- *Script* ditampilkan pada *test case pane*
- Memiliki dua *tab*
- Pertama untuk menampilkan perintah (*source code*)
- Dan parameter dalam *readable “table”* format

Command	select window	//	
Target	name=null		
Value			
Description			

Gambar 5. 1 Panel Selenium IDE

B. *Toolbar*

Toolbar berisi *buttons* untuk mengontrol eksekusi *test cases*.

C. *Menu bar*

- Menu file

Menu file untuk membuat, membuka, dan menyimpan *test case* dan *test suite files*.

- Menu edit

Menu edit untuk operasi *copy*, *paste*, *delete*, *undo*, dan *select all* untuk mengedit perintah pada *test case*.

- Menu *option*

Menu *options* untuk mengubah *setting*, yaitu mengeset nilai *timeout value* untuk perintah tertentu, menambahkan ekstensi *user-defined user* untuk *setting* dasar perintah Selenium dan menspesifikasi format atau bahasa yang digunakan ketika menyimpan *test cases*.

D. *Help bar*

Menjelaskan perintah Selenium. Set perintah tersebut disebut *selenese*. Perintah Selenium terdiri dari tiga jenis: Aksi, Asessor, dan Penegasan atau Aseri.

- Aksi

Yaitu Tindakan *user* pada aplikasi atau memerintahkan *browser* untuk melakukan sesuatu. Aksi merupakan perintah yang secara umum memanipulasi keadaan aplikasi.

- 1) Klik link - klik / *Clickandwait*

- 2) Memilih item

- Asessori

Menguji keadaan aplikasi dan menyimpan hasilnya pada variabel, misal “storeTitle”.

- Asersi

Digunakan untuk memvalidasi aplikasi.

- 1) Untuk verifikasi halaman *web*

- 2) Untuk verifikasi teks

- 3) Untuk verifikasi *alerts*

Asersi dapat digunakan dalam mode :

- 1) *Assert*

- 2) *Verify*

- 3) *waitFor*

Contoh : "*assertText*", "*verifyText*" dan "*waitForText*".

Note :

- 1) Ketika “*assert*” gagal, maka tes dihentikan.

- 2) Ketika “*verify*” gagal, maka tes tetap dilanjutkan

- 3) Perintah “*waitFor*” menunggu beberapa kondisi menjadi benar

Beberapa perintah Selenium yang sering digunakan untuk pengujian adalah sebagai berikut :

- 1) *Open* – membuka halaman menggunakan URL.

- 2) *Click* atau *clickAndWait* melaksanakan operasi klik dan secara opsional menunggu halaman baru muncul.

- 3) *verifyTitle* atau *assertTitle* – memverifikasi judul halaman yang diharapkan.
- 4) *verifyTextPresent* – memverifikasi teks yang diharapkan muncul pada halaman *web*.
- 5) *verifyElementPresent* – memverifikasi elemen UI yang diharapkan, sebagaimana didefinisikan oleh HTML *tag* muncul pada halaman *web*.
- 6) *verifyText* – memverifikasi teks yang diharapkan HTML *tag* terkait pada halaman *web*.
- 7) *verifyTable* – memverifikasi konten tabel yang diharapkan.
- 8) *waitForPageToLoad* – menghentikan sementara eksekusi sampai halaman baru yang diharapkan muncul. Dipanggil secara otomatis ketika *clickAndWait* digunakan.
- 9) *waitForElementPresent* – menghentikan sementara eksekusi sampai elemen UI yang diharapkan muncul, sebagaimana didefinisikan pada HTML *tag*.

5.3.3 Mengeset *Recording* dan *Run*

Ketika Selenium IDE mula-mula dibuka, *record button* secara *default* akan *ON*. Selama perekaman, Selenium IDE secara otomatis akan menyisipkan perintah ke dalam *test case* berdasarkan tindakan yang dilakukan.

1. *Remember base URL* MODE

Menggunakan basis URL untuk menjalankan *test case* pada *domain* yang berbeda.

2. *Record absolute recording* MODE

Menjalankan *test case* pada *domain* tertentu.

5.3.4 Menjalankan *Test Case*

1. Menjalankan *test case*: Klik *Run button* untuk menjalankan *test case* yang ditampilkan saat itu.
2. Menjalankan *test suite*: Klik *Run All button* untuk menjalankan *test suite* yang di-loading saat itu.

3. *Stop and start: Pause button* dapat digunakan untuk menghentikan *test case* ketika berjalan. *Icon button* ini akan berubah ke *Resume button*. Untuk melanjutkan klik *Resume*.
4. *Stop in the middle: Breakpoint* pada *test case* dapat di-*set* yang menyebabkan berhenti pada perintah tertentu. Biasanya digunakan untuk melakukan *debugging* pada *test case*. Untuk mengeset suatu *breakpoint*, pilih perintah, klik kanan dan dari *context* menu, pilih *Toggle Breakpoint*.
5. *Start from the middle*: Untuk memulai menjalankan perintah khusus pada tengah-tengah *test case*. Biasanya digunakan untuk melakukan *debugging* pada *test case*. Untuk mengeset *startpoint*, pilih *command*, *right-click*, dan dari *context* menu pilih *Set* atau *Clear Start Point*.
6. *Run any single command: Double-click* suatu perintah tunggal untuk menjalankannya sendirian. Berguna ketika menulis perintah tunggal.

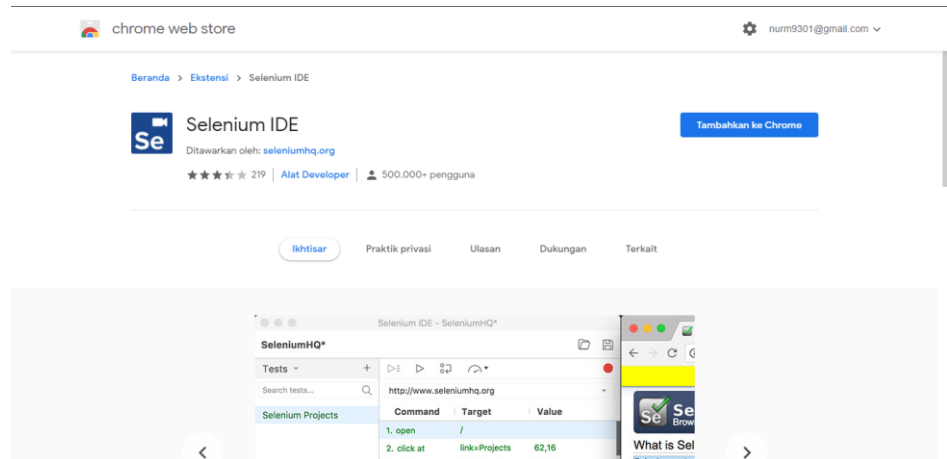
Test suite

Sebuah *test suite* adalah kumpulan tes. Seringkali kita akan menjalankan semua tes di *suite* tes sebagai salah satu *batch-job* kontinyu. Apabila menggunakan Selenium IDE, *test suite* juga dapat didefinisikan menggunakan file HTML sederhana. Sintaks juga sederhana. Tabel HTML mendefinisikan daftar tes dimana setiap baris mendefinisikan *path file system* untuk setiap tes.

5.4 Prosedur Praktikum

1. Meng-*instal* Selenium IDE

a) Men-*download* Selenium IDE dari Google Chrome

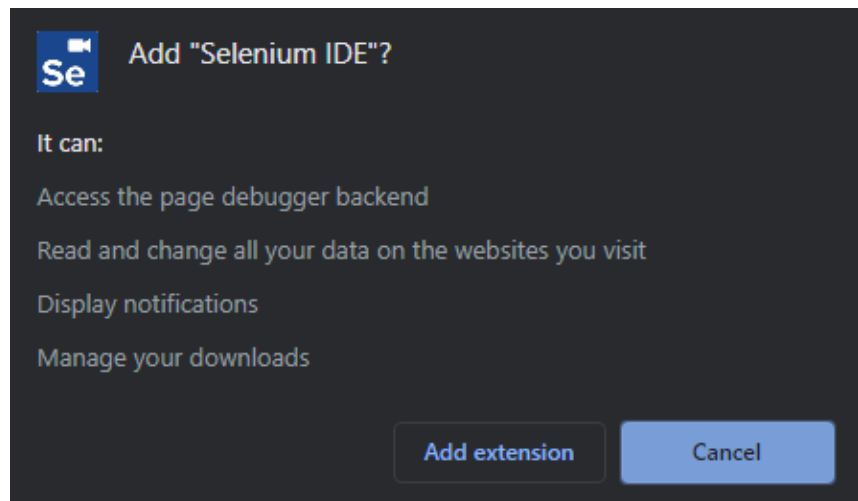


Gambar 5. 2 Website Selenium IDE

Gambar 5.2 adalah laman untuk men-*download extension* Selenium IDE. *Extension* tersebut dapat diunduh di *link* berikut:

<https://chrome.google.com/webstore/detail/selenium-ide/mooikfkahbdckldjjndioackbalphokd>

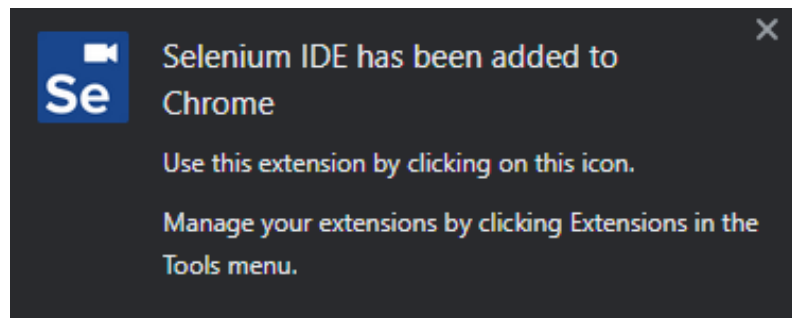
b) Pop up “add extension”



Gambar 5. 3 Pop up "add extension"

Gambar 5.3 adalah *pop up* “add extension”. Untuk melanjutkan proses *download* dan *install extension* Selenium IDE klik *button* “add extension”.

c) Instalasi selesai

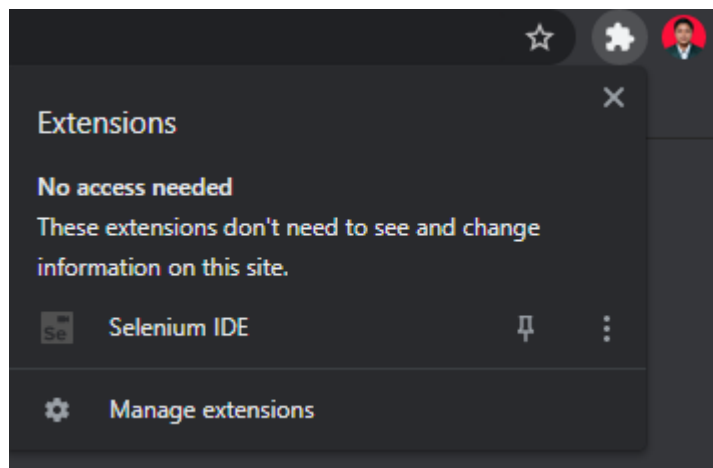


Gambar 5. 4 Berhasil menambahkan *extension* Selenium IDE pada Chrome

Gambar 5.4 adalah menandakan *extension* Selenium IDE berhasil ditambahkan pada Chrome dan siap untuk digunakan.

2. Membuka IDE

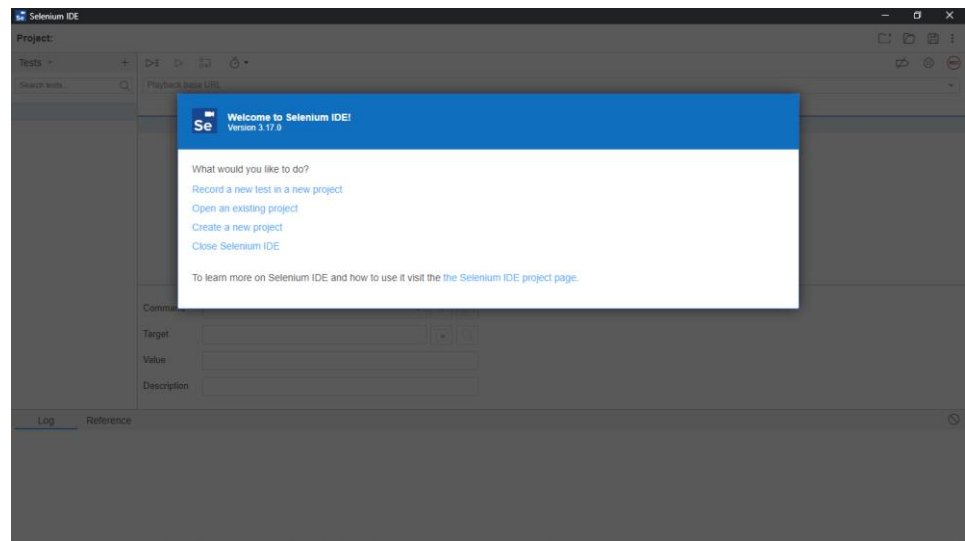
a) Klik *tab “extensions”*



Gambar 5. 5 *Tab “extensions”*

Gambar 5.5 adalah *tab “extensions”* yang ada pada Google Chrome. Posisi *tab “extensions”* berada di sebelah kiri *tab profile* email kita.

b) Klik *extension* “Selenium IDE”



Gambar 5. 6 Halaman awal Selenium IDE

Setelah mengklik *extension* “Selenium IDE”, maka kita akan dibawa ke halaman awal dari Selenium IDE, seperti pada gambar 5.6. itu artinya kita sudah siap untuk membuat menjalankan *test suite* dan *test case* di Selenium IDE.

3. Menggunakan Selenium IDE, tulis *test suite* yang berisi minimal dua kasus uji

a) *Test case 1*

- Buka Selenium IDE, buat *test suite* dan *test case*
- Klik tombol “*recording*”
- Buka halaman *web* (misal: ketikkan <https://www.google.com>)
- Ketikkan “*energy efficient*” pada *Google search input box*
- Klik *search button*
- Verifikasi munculnya teks adalah “*energy efficient*”
- *Assert title* sebagai “*energy efficient – Google Search*”
- Simpan *test case* dengan ekstensi .html

b) *Test case 2*

- Buka halaman *web* (misal: ketikan <https://www.google.com>)
- Ketikan “Selenium RC” pada *Google search input box*
- Klik *search button*
- Verifikasi munculnya teks adalah “*energy efficient*”
- *Assert title* sebagai “*energy efficient – Google Search*”
- Simpan *test case* dengan ekstensi .html

Tahap – tahap *test suite*: Buat beberapa *test case* dan simpan setiap *test case* dengan ekstensi .html

- a) Buka Google Chrome
- b) Klik *tab “extensions”*
- c) Klik *extension “Selenium IDE”*
- d) Klik “*Create a new project*”
- e) Tuliskan nama projeknya
- f) Pilih *test suite*
- g) Klik “*Add new suite*”
- h) Klik “*Add test*”
- i) Tambahkan beberapa *test cases*
- j) Simpan *suite* dengan ekstensi .html
- k) Jalankan *test suite*

4. Menjalankan *test suite* yang berisi dua *website* berbedaa) *Test case 1*

- Buka Selenium IDE, buat *test suite* dan buat *test case*
- Klik tombol *recording*
- Buka halaman *web* (misal: <https://www.google.com>)
- Ketikan “*energy efficient*” pada *Google search input box*
- Klik *search button*
- Verifikasi munculnya teks adalah “*energy efficient*”
- *Assert title* sebagai “*energy efficient – Penelusuran Google*”
- Simpan *test case* dengan ekstensi .html

b) *Test case 2*

- Buka Chrome *web browser*
- Ketikan <https://www.yahoo.com> pada *address bar*
- Ketikan “*energy efficient*” pada *Google search input box*
- Klik *search button*
- Tunggu hasil pencarian sehingga menuju ke <https://search.yahoo.com>
- Verifikasi munculnya teks adalah “*energy efficient*”
- *Assert title* sebagai “*energy efficient – Yahoo Search Results*”
- Simpan *test case* dengan ekstensi .html

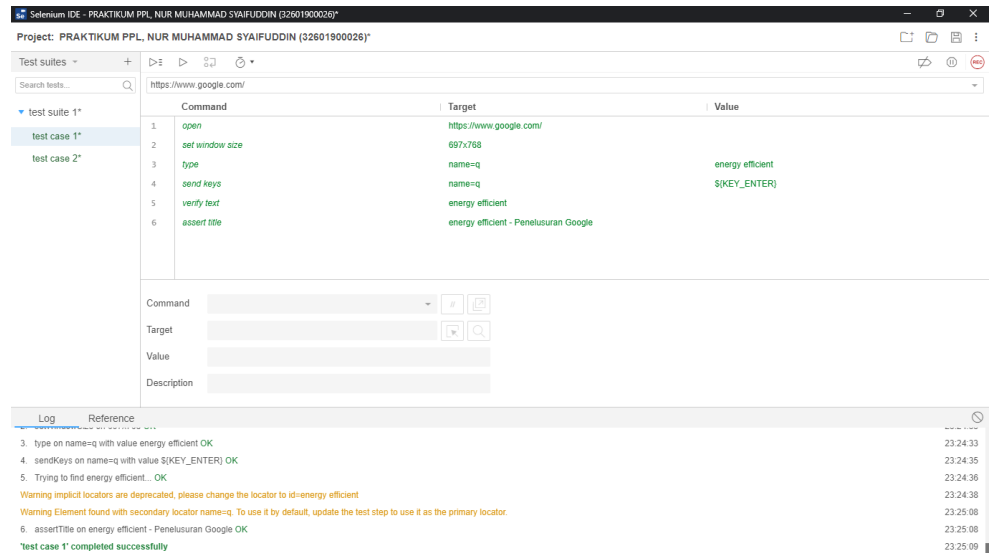
Tahap – tahap *test suite*: Buat beberapa *test case* dan simpan setiap *test case* dengan ekstensi .html.

- a) Buka Google Chrome
- b) Klik *tab “extensions”*
- c) Klik *extension “Selenium IDE”*
- d) Klik “*Create a new project*”
- e) Tuliskan nama projeknya
- f) Pilih *test suite*
- g) Klik “*Add new suite*”
- h) Klik “*Add test*”
- i) Tambahkan beberapa *test cases*
- j) Simpan *suite* dengan ekstensi .html
- k) Jalankan *test suite*

5.5 Hasil Praktikum

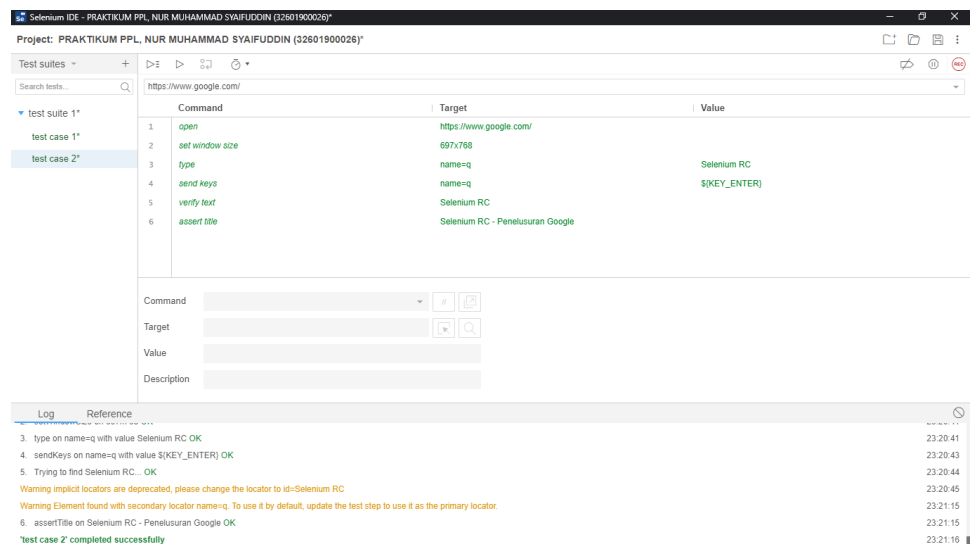
1. Test suite 1

a. Test case 1



Gambar 5. 7 Hasil test case “energy efficient” Google Search

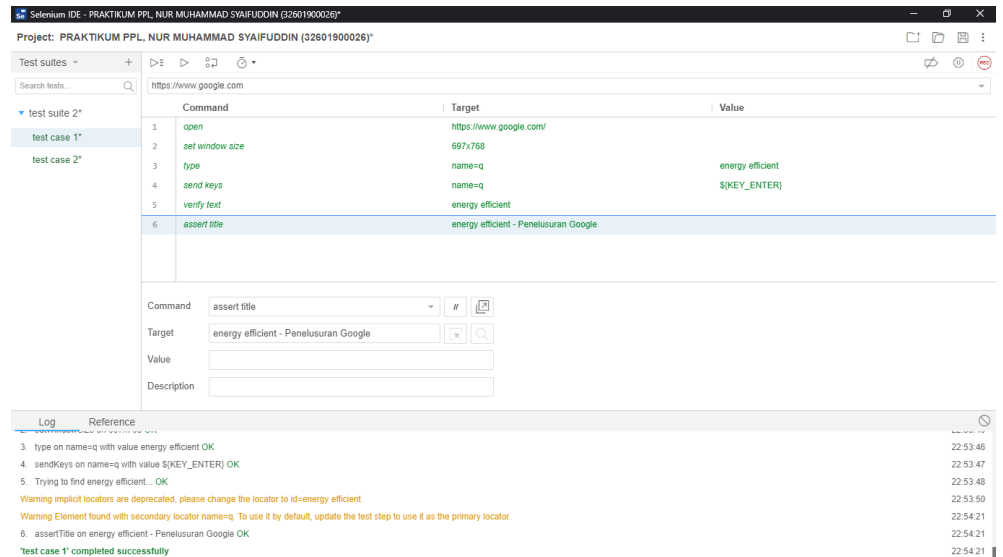
b. Test case 2



Gambar 5. 8 Hasil test case “Selenium RC” Google Search

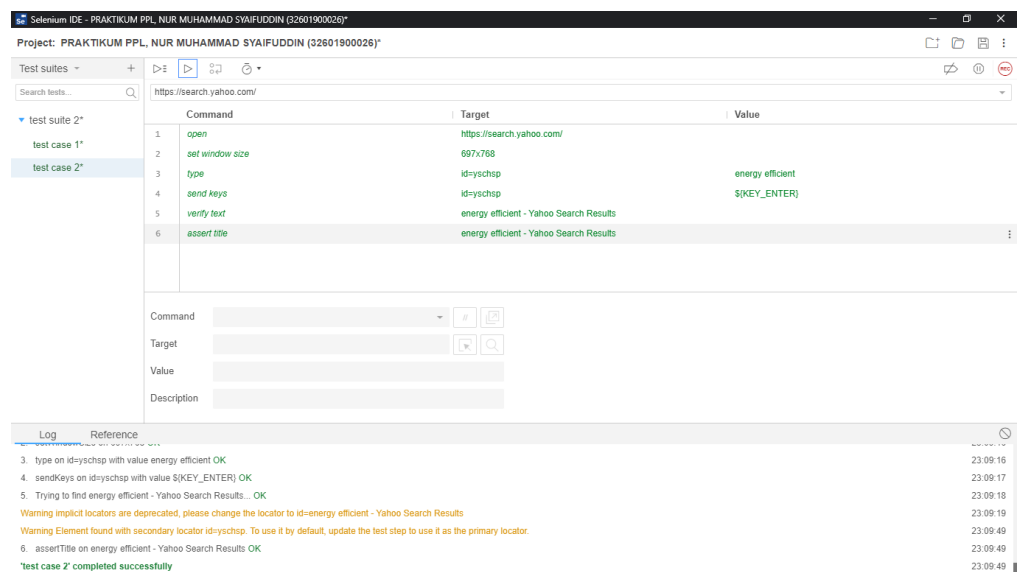
2. Test suite “energy efficient”

a. Test case 1



Gambar 5. 9 Hasil test case “energy efficient” Google Search

b. Test case 2



Gambar 5. 10 Hasil test case “energy efficient” Yahoo Search

5.6 Analisa

1. *Test suite 1*

a) *Test case 1*

Pada *test suite 1 test case* ke-1 ditampilkan *command* pertama yaitu *open* yang artinya terjadi proses membuka jendela *browser* dengan *website google*. *Command* selanjutnya adalah *set window size*, yang artinya menentukan panjang dan lebar dari *tab window* yang digunakan. *Command* selanjutnya adalah *type*, yaitu proses untuk mengetikkan kata untuk dicari pada *search engine*, pada kasus ini, kata yang diketikkan adalah “*energy efficient*”. *Command* selanjutnya adalah *verify text* dan *asserts title*, untuk memverifikasi teks “*energy efficient*” dan *title* “*energy efficient – Penelusuran Google*” yang telah diketikkan telah ditemukan atau tidak.

b) *Test case 2*

Pada *test suite 1 test case* ke-2 ditampilkan *command* pertama yaitu *open* yang artinya terjadi proses membuka jendela *browser* dengan *website google*. *Command* selanjutnya adalah *set window size*, yang artinya menentukan panjang dan lebar dari *tab window* yang digunakan. *Command* selanjutnya adalah *type*, yaitu proses untuk mengetikkan kata untuk dicari pada *search engine*, pada kasus ini, kata yang diketikkan adalah “*Selenium RC*”. *Command* selanjutnya adalah *verify text* dan *asserts title*, untuk memverifikasi teks “*Selenium RC*” dan *title* “*Selenium RC – Penelusuran Google*” yang telah diketikkan telah ditemukan atau tidak.

2. *Test suite 2*

a) *Test case 1*

Pada *test suite 2 test case* ke-1 ditampilkan *command* pertama yaitu *open* yang artinya terjadi proses membuka jendela *browser* dengan *website google*. *Command* selanjutnya adalah *set window size*, yang artinya menentukan panjang dan lebar dari *tab window* yang digunakan. *Command* selanjutnya adalah *type*, yaitu proses

untuk mengetikkan kata untuk dicari pada *search engine*, pada kasus ini, kata yang diketikkan adalah “*energy efficient*”. *Command* selanjutnya adalah *verify text* dan *asserts title*, untuk memverifikasi teks “*energy efficient*” dan *title* “*energy efficient – Penelusuran Google*” yang telah diketikkan telah ditemukan atau tidak.

b) *Test case 2*

Pada *test suite 2 test case* ke-2 ditampilkan *command* pertama yaitu *open* yang artinya terjadi proses membuka jendela *browser* dengan *website yahoo*. *Command* selanjutnya adalah *set window size*, yang artinya menentukan panjang dan lebar dari *tab window* yang digunakan. *Command* selanjutnya adalah *type*, yaitu proses untuk mengetikkan kata untuk dicari pada *search engine*, pada kasus ini, kata yang diketikkan adalah “*energy efficient*”. *Command* selanjutnya adalah *verify text* dan *asserts title*, untuk memverifikasi teks “*energy efficient*” dan *title* “*energy efficient – Yahoo Search Results*” yang telah diketikkan telah ditemukan atau tidak.

5.7 Kesimpulan

Pada pengujian BAB V ini menggunakan fungsional dengan menggunakan *tools* Selenium IDE. Fungsional *test* adalah sebuah proses *testing* dimana *software tester* berperilaku sebagai *end-user* dan memeriksa apakah fungsi dari sistem dapat berjalan dengan baik dari *user*. *Tools* Selenium IDE merupakan *integrated tool* untuk *agile testing*. Cara kerja Selenium IDE adalah dengan merekam semua aktifitas yang dilakukan *user* saat mengakses aplikasi berbasis web.

BAB VI

PENGUJIAN *STRESS* DENGAN APACHE JMeter

6.1 Tujuan

1. Praktikan dapat memahami pendekatan pengujian terotomasi
2. Praktikan dapat menggunakan Apache JMeter untuk menguji kinerja *stress* dari suatu *web server*

6.2 Alat dan Bahan

1. Komputer *desktop*
2. *Software* JMeter
3. JVM 8 atau lebih
4. *Web server*

6.3 Dasar Teori

Stress test adalah jenis pengujian yang unik. Pengujian ini dilakukan dengan memberikan beban pada sebuah aplikasi untuk mengetahui titik performansi aplikasi tersebut. Dirancang untuk menghadapi situasi yang tidak normal pada saat program mengalami uji coba. *Stress testing* dilakukan oleh sistem untuk kondisi-kondisi seperti volume data yang tidak normal (melebihi atau kurang dari batasan) frekuensinya. *Stress test* sering dilakukan pada aplikasi yang membutuhkan konkurasi maupun akses acak bersamaan dalam jumlah yang sangat banyak. Aplikasi dengan berbasis *web* dengan *request* yang sangat banyak menjadi contoh yang sangat menarik dalam hal ini. Dalam pengujian *stress test* akan menggunakan sebuah *tools* Apache JMeter. Apache JMeter adalah sebuah *tool* yang digunakan untuk melakukan *performace test* pada sebuah *software*. Apache JMeter dapat memberikan *request* dalam jumlah yang sangat banyak secara bersamaan dalam satu waktu pada server Apache JMeter. (Aji, 2006)

JMeter atau Apache JMeter merupakan *tool* pengujian kinerja yang bersifat *open source* berbasis Java, sangat mudah digunakan dan mendukung beberapa protokol, termasuk HTTP/ HTTPS, SOAP, JDBC, LDAP dan JMS. Bagi seorang *QA Engineer*, JMeter bisa digunakan untuk melakukan *load* atau *stress testing web application*, *FTP application* dan *database server test*.

JMeter bisa dijalankan dengan 2 cara, dengan GUI atau non-GUI (*Command line*). Untuk *beginner* lebih baik menggunakan cara yang pertama. Mudah dan tanpa melakukan *scripting* tertentu. Tinggal membuat *Test Plan*, mengisikan berapa *thread* dan sampel yang akan diujicobakan, *running* dan menganalisa hasil atau *report*.

JMeter dibangun untuk mendukung rencana uji atau *Test Plan*. Dalam rencana uji terdapat *Thread Group*, *Samplers* atau pengendali, *Listeners* atau *pendengar*, *timer*, *Assertions* atau pernyataan, dan elemen lainnya. Setiap rencana uji adalah skenario pengujian kinerja dimana serangkaian langkah JMeter akan dieksekusi ketika rencana tersebut dijalankan. Tabel berikut memberikan gambaran singkat tentang elemen dasar yang dapat dimasukkan dalam rencana uji.

Tabel 6. 1 Elemen dasar pengujian menggunakan JMeter

Elemen	Deskripsi
<i>Thread Group</i>	Seperti halnya uji beban, eksekusi <i>multi-threaded</i> . <i>Thread Group</i> adalah apa yang mengontrol koneksi bersamaan untuk aplikasi Anda.
<i>Samplers</i>	Sebuah <i>sampler</i> adalah jenis dasar kontroler. Sangat sederhana, <i>samplers</i> memberitahu JMeter untuk mengirim permintaan (HTTP, SOAP, dll ...) ke <i>server</i> .
<i>Listeners</i>	<i>Listeners</i> adalah apa yang Anda gunakan untuk mengakses informasi yang dikumpulkan JMeter saat dijalankan.

<i>Timers</i>	<i>Timers</i> adalah bagaimana mengatur penundaan di JMeter. <i>timer</i> bekerja sebelum setiap permintaan atau <i>request</i> yang dibuat oleh <i>thread</i>
<i>Assertions</i>	<i>Assertions</i> , seperti di alat uji, memungkinkan untuk memeriksa perilaku tertentu ketika mengeksekusi tes. <i>Assertion</i> menyediakan hasil standar lulus / gagal.

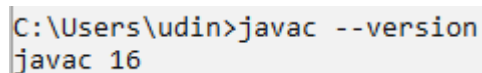
6.4 Prosedur Praktikum

1. Menginstal Apache JMeter

a. Instalasi Java

Karena JMeter adalah aplikasi berbasis Java, maka terlebih dahulu kita harus instal Java dan memastikan berjalan di sistem operasi kita. Untuk itu download dan instal terlebih dahulu Java SE Development Kit di <https://jdk.java.net/>. Saat kita mengunduh JDK, maka kita juga sudah mengunduh JRE, karena JRE secara otomatis sudah ada di dalam JDK. Saya mengunduh JDK dari “OpenJDK” karena saat ini sangat populer dikalangan programmer Java dibanding Oracle. Untuk mengecek apakah java sudah terinstal dengan baik, cek melalui *command prompt* (Windows + R > cmd), ketikkan perintah :

```
- javac -version
```



```
C:\Users\udin>javac --version
javac 16
```

Gambar 6. 1 Mengecek versi JDK

Gambar 6.1 adalah perintah untuk mengecek versi JDK (Java Development Kit).

- `java -version`

```
C:\Users\udin>java --version
openjdk 16 2021-03-16
OpenJDK Runtime Environment (build 16+36-2231)
OpenJDK 64-Bit Server VM (build 16+36-2231, mixed mode, sharing)
```

Gambar 6. 2 Mengecek versi JRE

Gambar 6.2 adalah perintah untuk mengecek versi JRE (*Java Runtime Environment*).

b. Instalasi JMeter

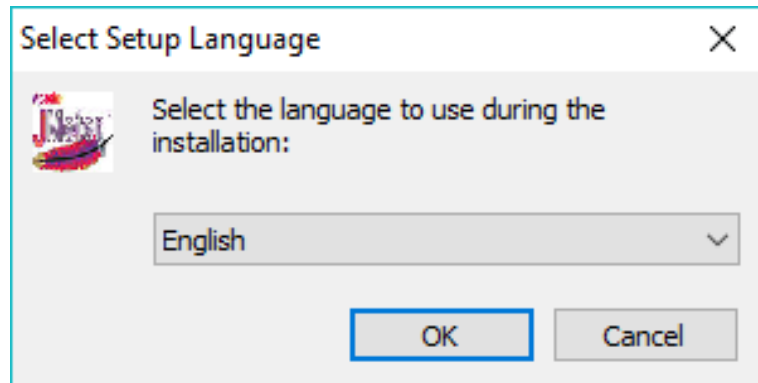
1) Instalasi dari file binary

Paket - paket JMeter bisa langsung di-*download* di situs http://jmeter.apache.org/download_jmeter.cgi. Pilih *Binaries, download* yang berekstensi .zip (misal: apache-jmeter-3.0.zip). Tidak ada tahapan khusus dalam instalasi JMeter, file zip yang telah berhasil di-*download*, tinggal ekstrak di folder yang diinginkan. Untuk keperluan *load testing*, kita masih membutuhkan beberapa *plugin* untuk ditambahkan ke JMeter. *Plugin* JMeter dapat di-*download* di <https://jmeter-plugins.org/downloads/all/>. Download :

- **JMeterPlugins-Standard.** Cara menambahkan: ekstrak > kopikan seluruh isi lib\ext ke dalam folder lib\ext dari apache-jmeter.
- **JMeterPlugins-Extras.** Cara menambahkan: ekstrak > kopikan seluruh isi bin ke dalam folder bin apache-jmeter, kopikan seluruh isi lib\ext\ ke dalam folder lib\ext\ dari apache-jmeter. Taruh file atau folder lain ke dalam *root* apache-jmeter.
- **ServerAgent.** Ekstrak dan pisahkan dari direktori JMeter atau taruh di luar folder JMeter.
- Lakukan hal yang sama untuk *plugin* lain.

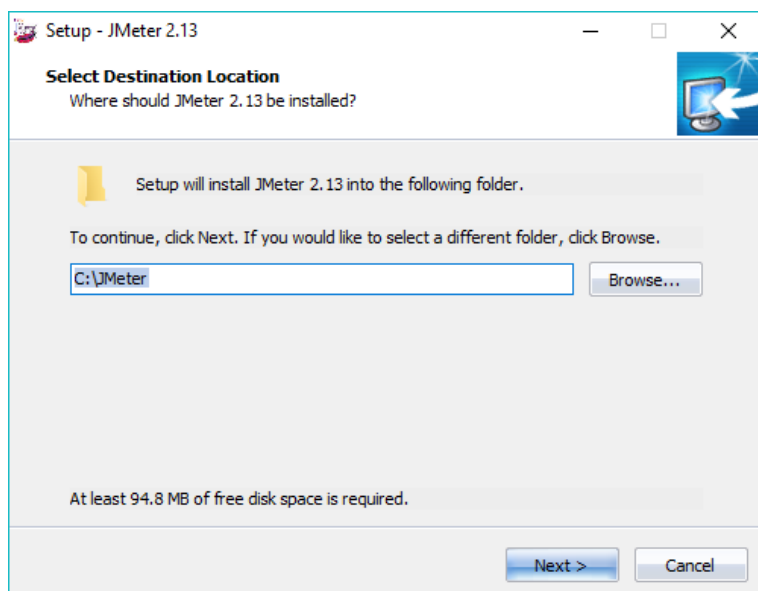
2) Instalasi Apache JMeter *for* Windows

- Download file Apache JMeter *for* Windows, buka lokasi penyimpanan file JMeter, jika sudah diklik akan muncul *form* dialog pemilihan bahasa disini kita pilih saja Bahasa Inggris kemudian klik OK.



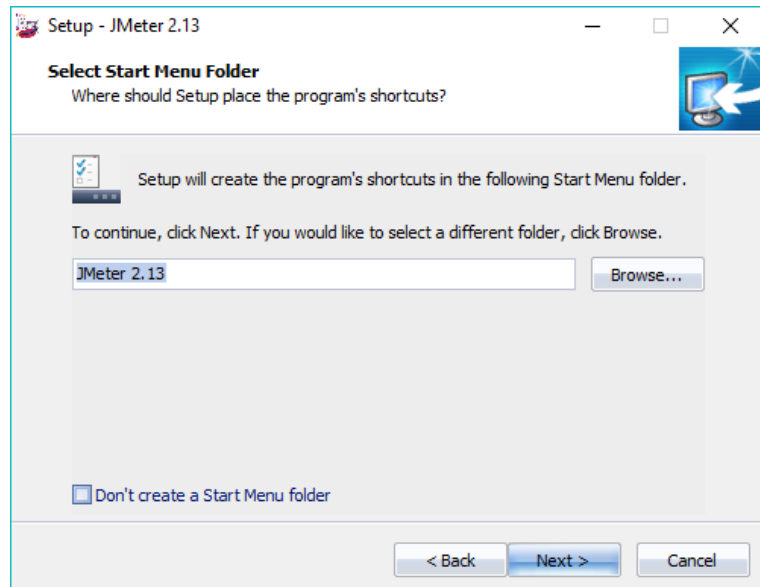
Gambar 6. 3 Memilih bahasa

- Klik *next* untuk melanjutkan.
- Pilih direktori untuk lokasi penyimpanan JMeter saat diinstal kemudian klik *next*.



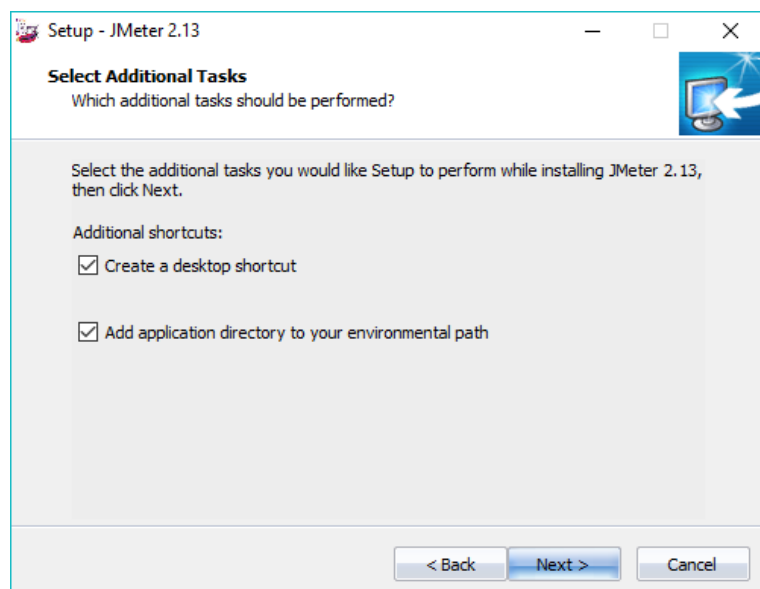
Gambar 6. 4 Direktori instalasi Apache JMeter

- Jendela *start menu* Apache JMeter. Klik *next* untuk melanjutkan.



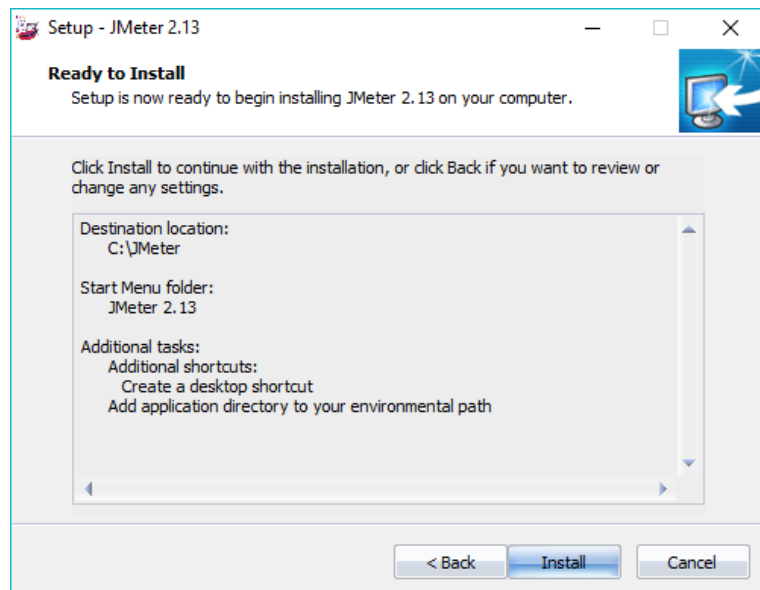
Gambar 6. 5 *Start menu* Apache JMeter

- Untuk menambahkan *shortcut* pada *desktop*, *checkboxlist* bagian *create a desktop shortcut*.



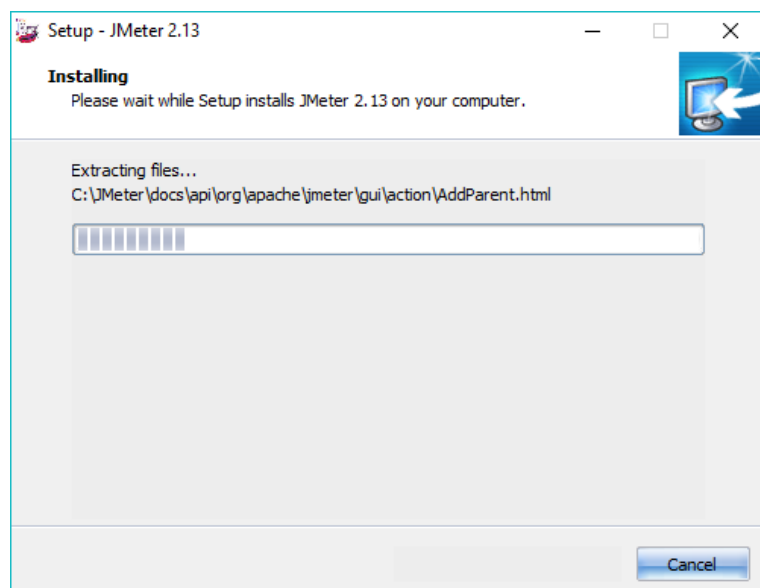
Gambar 6. 6 Menambahkan *shortcut* Apache JMeter pada *desktop*

- Klik *install* untuk melakukan proses instalasi.



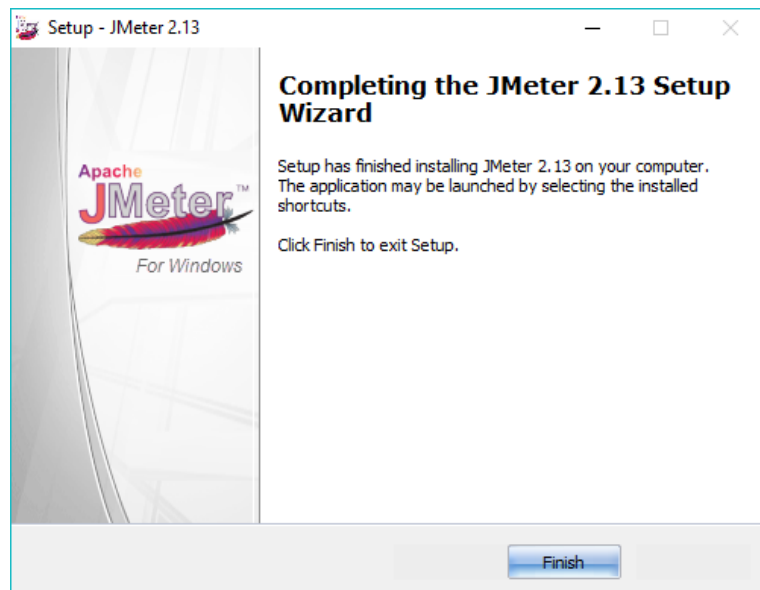
Gambar 6. 7 *Installation resume*

- Tunggu hingga proses instalasi selesai



Gambar 6. 8 Proses instalasi

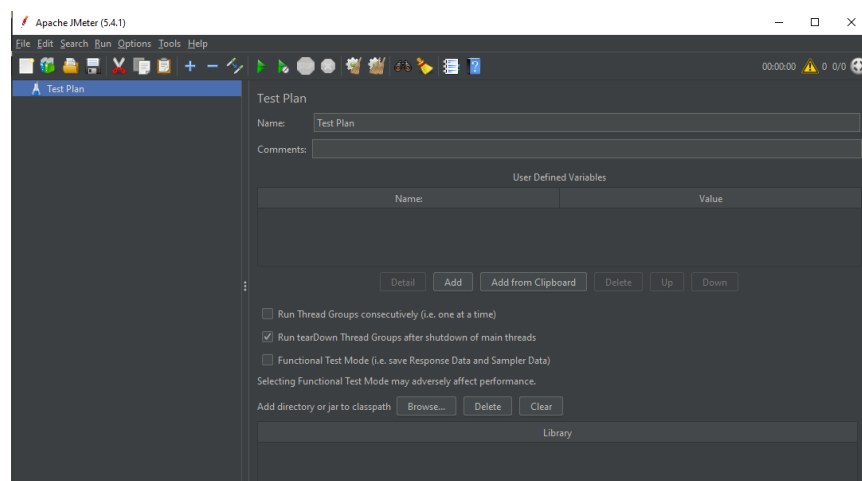
- Instalasi selesai



Gambar 6. 9 Instalasi Apache JMeter selesai

2. Menjalankan Apache JMeter

- Cara menjalankan Apache JMeter, cukup masuk ke folder bin kemudian mengklik dua kali jmeter.bat.



Gambar 6. 10 Tampilan Apache JMeter

- Untuk beberapa kasus tertentu, misalkan menjalankan perform server, diharuskan untuk menjalankan *ServerAgent* terlebih dahulu dengan cara klik dua kali *StartAgent.bat*. Alternatif kedua adalah dengan mengklik ikon JMeter.

3. Membuat *performance Test Plan*

Setelah JMeter sudah berhasil dibuka, selanjutnya kita bisa menyiapkan *test plan*.

a. *Add Thread Group*

Menambahkan trafik atau *user visitor* ke dalam komponen yang ingin di uji. Langkahnya sebagai berikut:

- 1) Mengklik kanan *test plan*
- 2) *Add > Threads (users) > Thread Group*
- 3) Dalam kontrol panel *Thread Group*, *Entri* pada *Thread Properties* :
 - *Number of threads (users)*: isi berapa *user/ visitor* yang akan mengakses *web*.
 - *Ramp-up Period (in seconds)*: isi berapa waktu *delay* antara *user* satu dengan yang lainnya dalam mengakses *web*.
 - *Loop Count*: waktu eksekusi, bertahap atau seterusnya.

b. *Add JMeter Element*

Menambahkan *web server* atau *IP address* yang akan diuji. Caranya:

- 1) Klik kanan *Threads Group*
- 2) *Add > Config Element > HTTP request defaults*
- 3) Pada *web server* isi *server name* atau IP dan *port*-nya, atau lebih mudahnya isi *website* atau URL yang akan di *test*. URL diisi dengan format <http://www>.
- 4) Jika tidak hanya halaman utama yang di tes, kita bisa menambahkan *path* atau foldernya. Caranya :
 - Klik kanan *Threads Group*
 - *Add > Sampler > HTTP request*
 - Isi *web server*, *port*, dan *path*

c. *Add Listener*

Menampilkan proses dan hasil *test* secara grafis atau berbentuk tabel. Caranya:

- 1) Klik kanan *Test Plan*
- 2) *Add > Listener > Graph Result*
- 3) *Add > Listener > View Results in Table*

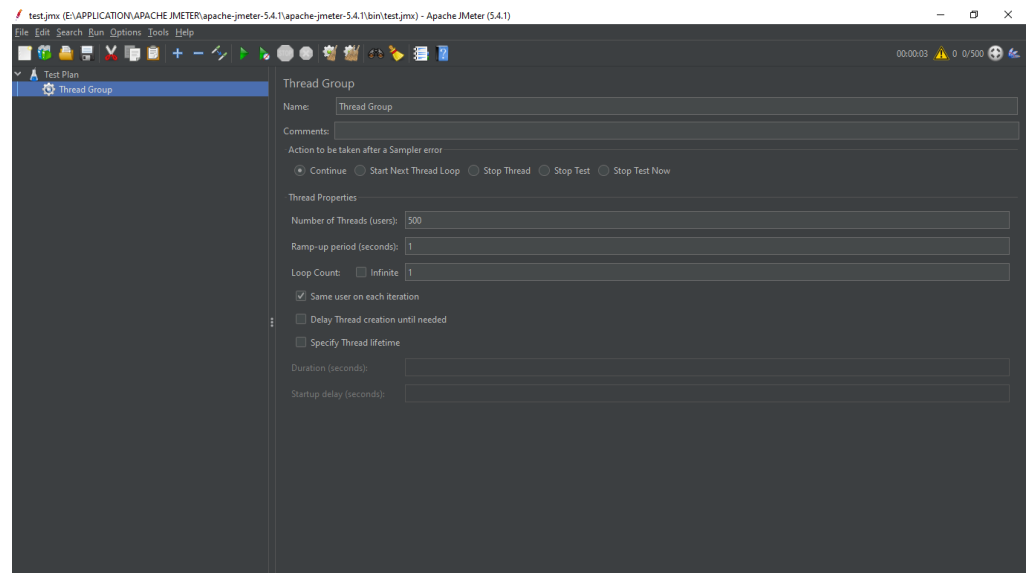
d. *Run Test*

Menjalankan test secara otomatis. Caranya:

- 1) Simpan terlebih dahulu *test plan* yang telah kita buat, *file > Save* (Ctrl + S).
- 2) Klik *run* atau Ctrl + R, JMeter akan mulai mensimulasi sejumlah *user* dalam mengakses *web server* yang telah ditentukan.

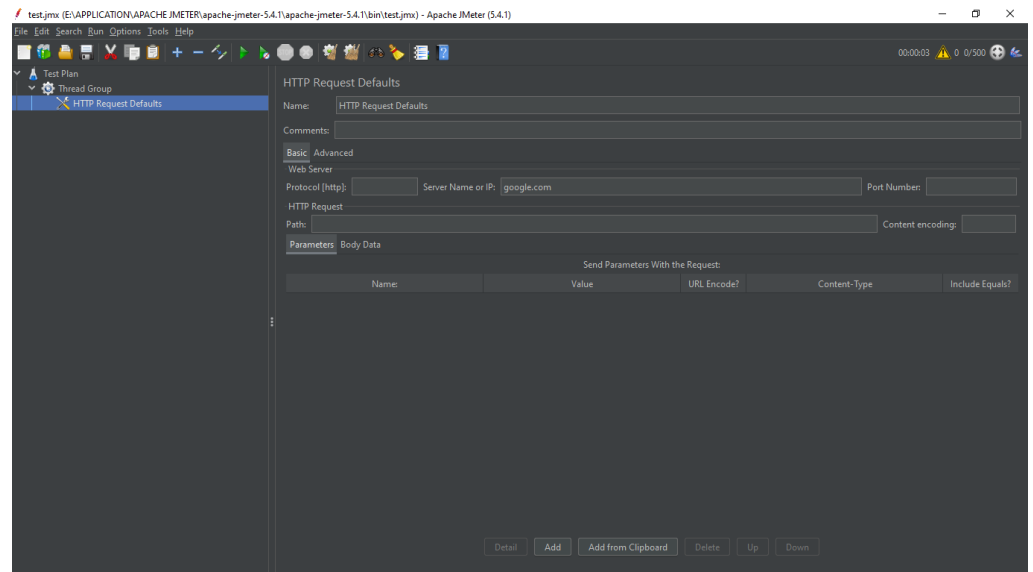
6.5 Hasil Praktikum

1. Membuat *Thread Group*

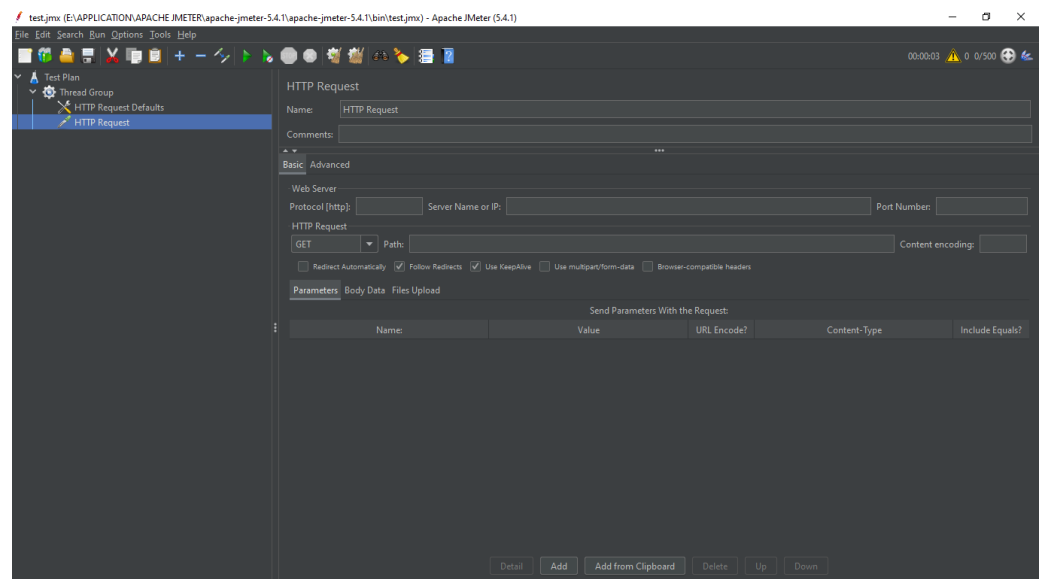


Gambar 6. 11 Membuat *Thread Group*

2. Membuat HTTP *request*

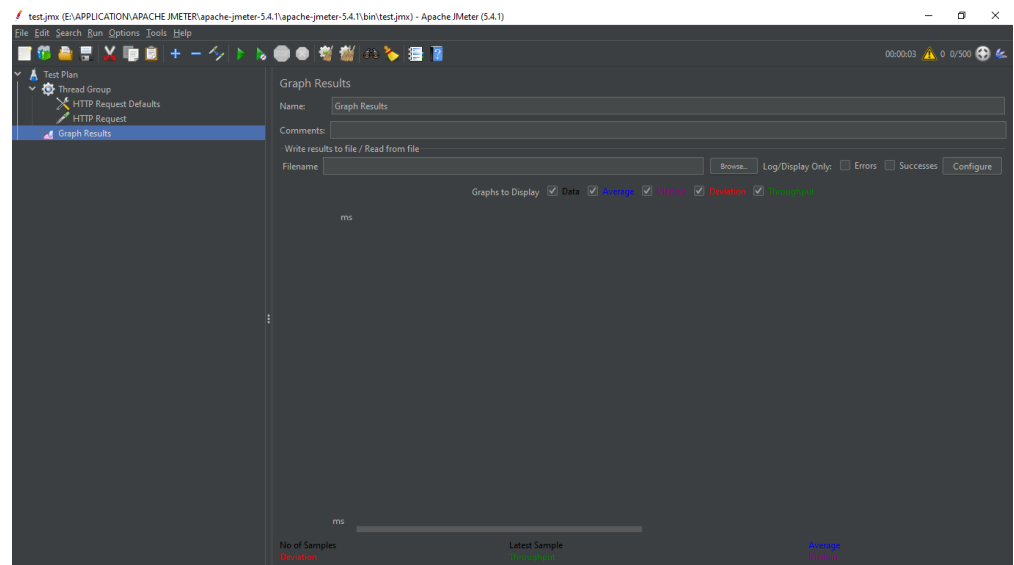


Gambar 6. 12 Membuat HTTP *request defaults*

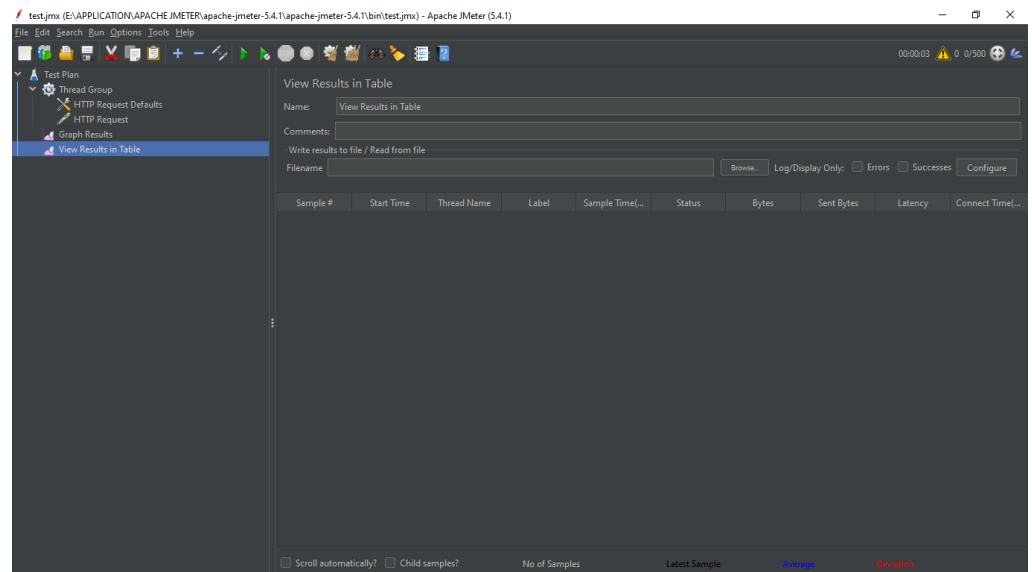


Gambar 6. 13 Membuat HTTP *request*

3. Menambahkan *Listener*

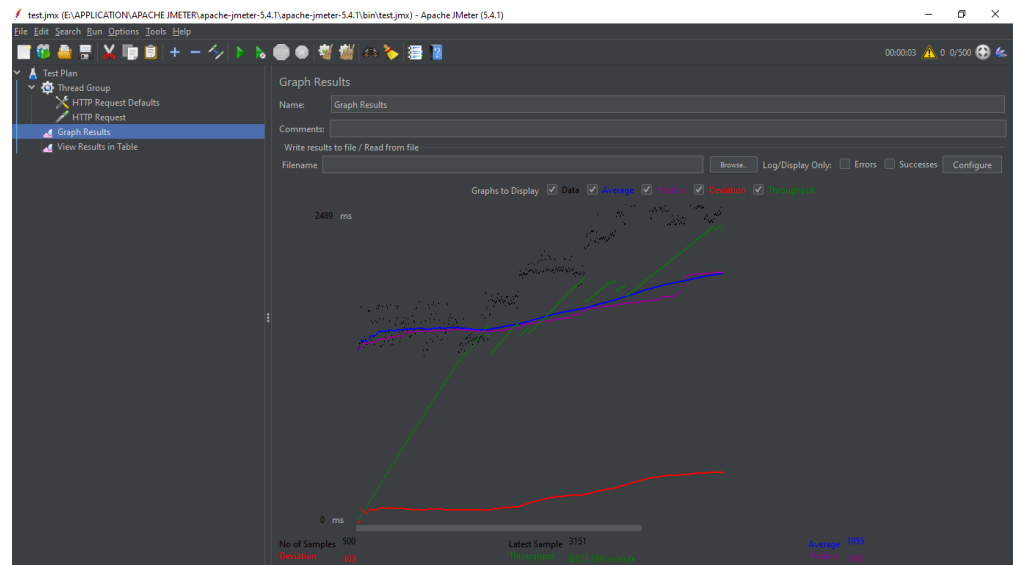


Gambar 6. 14 Menambahkan *Listener Graph Results*



Gambar 6. 15 Menambahkan *Listener View Results in Table*

4. Menjalankan *test*



Gambar 6. 16 Results in Graphic

Sample #	Start Time	Thread Name	Label	Sample Time	Status	Bytes	Sent Bytes	Latency	Connect Time
479	08:13:24.445	Thread Group ...	HTTP Request	2370	✓	15476	234	1222	727
480	08:13:24.454	Thread Group ...	HTTP Request	2361	✓	15496	234	1214	719
481	08:13:24.466	Thread Group ...	HTTP Request	2349	✓	16375	234	1203	730
482	08:13:24.462	Thread Group ...	HTTP Request	2353	✓	15497	234	1210	721
483	08:13:24.455	Thread Group ...	HTTP Request	2360	✓	15532	234	1201	718
484	08:13:24.450	Thread Group ...	HTTP Request	2366	✓	15562	234	1212	723
485	08:13:24.466	Thread Group ...	HTTP Request	2350	✓	15493	234	1193	717
486	08:13:24.463	Thread Group ...	HTTP Request	2353	✓	15630	234	1208	719
487	08:13:24.441	Thread Group ...	HTTP Request	2376	✓	15468	234	1225	729
488	08:13:24.452	Thread Group ...	HTTP Request	2411	✓	15560	234	1214	720
489	08:13:24.454	Thread Group ...	HTTP Request	2412	✓	15475	234	1214	719
490	08:13:24.461	Thread Group ...	HTTP Request	2406	✓	15574	234	1209	735
491	08:13:24.462	Thread Group ...	HTTP Request	2405	✓	15521	234	1205	720
492	08:13:24.466	Thread Group ...	HTTP Request	2422	✓	15527	234	1222	726
493	08:13:24.460	Thread Group ...	HTTP Request	2408	✓	15550	234	1206	718
494	08:13:24.426	Thread Group ...	HTTP Request	2443	✓	15601	234	1241	744
495	08:13:24.465	Thread Group ...	HTTP Request	2411	✓	15510	234	1202	713
496	08:13:24.459	Thread Group ...	HTTP Request	2425	✓	15458	234	1208	720
497	08:13:24.430	Thread Group ...	HTTP Request	2468	✓	15513	234	1231	739
498	08:13:24.437	Thread Group ...	HTTP Request	2509	✓	15496	234	1217	733
499	08:13:24.112	Thread Group ...	HTTP Request	2855	✓	15547	234	943	260
500	08:13:24.049	Thread Group ...	HTTP Request	3151	✓	15492	234	317	84

Gambar 6. 17 Results in Table

6.6 Analisa

Pada *test* diatas ditampilkan beberapa pengecekan sebuah *website* dengan menggunakan *tool* Apache JMeter. Dalam hal ini *website* yang diuji adalah google.com. Hal yang diuji adalah data berupa ip *server* yang diuji, *thread group*, lama waktu respon, status. Dari data diatas juga ditampilkan hasil dimana keseluruhan 500 *user* sukses mendapat respon.

6.7 Kesimpulan

Pada praktikum BAB VI pengujian *stress* dengan menggunakan *tool* Apache JMeter. JMeter adalah *tool* yang digunakan untuk melakukan *performace test* pada sebuah *software*. JMeter dapat memberikan analisa dan laporan dari hasil pengujian dengan menguji sebuah *website*. JMeter dapat mengetahui bagaimana *website* yang dibangun dapat menangani respon dari *request* yang dilakukan oleh *client* atau *user*. Semakin banyak *user* yang mengakses *website* tersebut dan semakin banyak respon sukses yang diberikan, maka *website* tersebut tahan ketika sudah di-*upload* dan digunakan secara global.

DAFTAR PUSTAKA

- Aji, H. S. (2006) 'Menganalisis Web dan Keamananya Menggunakan Jmeter'.
- Jaya, T. S. (2019) 'Selenium IDE', *Tutorials Point (I) Pvt. Ltd.*, pp. 1–13.
- Lutfi, M. (2018) 'Pengujian Aplikasi Dengan Metode Blackbox Testing Boundary Value Analysis (Studi Kasus: Kantor Digital Politeknik Negeri Lampung)', *Jurnal Informatika: Jurnal Pengembangan IT (JPIT)*, 3(2), pp. 45–48. doi: 10.30591/jpit.v3i1.647.
- Rusfandi (2018) 'White Box Testing', (November), pp. 7–11.
- Sugiyono, P. D. (2018) 'Teknik Pengujian Perangkat Lunak'.

LEMBAR ASISTENSI



PRAKTIKUM : Pengujian Perangkat Lunak

NAMA : Nur Muhammad Syaifuddin

NIM : 32601900026

KELOMPOK : -

No		KETERANGAN	TANDA TANGAN	
1			
2			
PRETEST				
1			
2			
ASISTENSI LAPORAN				
1	7 Mei 2021	BAB 1 dan 2 ACC	
2	11 Mei 2021	BAB 3 ACC	
3	16 Mei 2021	BAB 4 kesimpulan masih salah	
4	17 Mei 2021	BAB 4 ACC	
5	19 Mei 2021	BAB 5 ACC	
6	20 Mei 2021	BAB 6 ACC	
7			
8			
9			
10			

11			
12			
13			
14			
15			
16			
17			
18			
19			
20			

