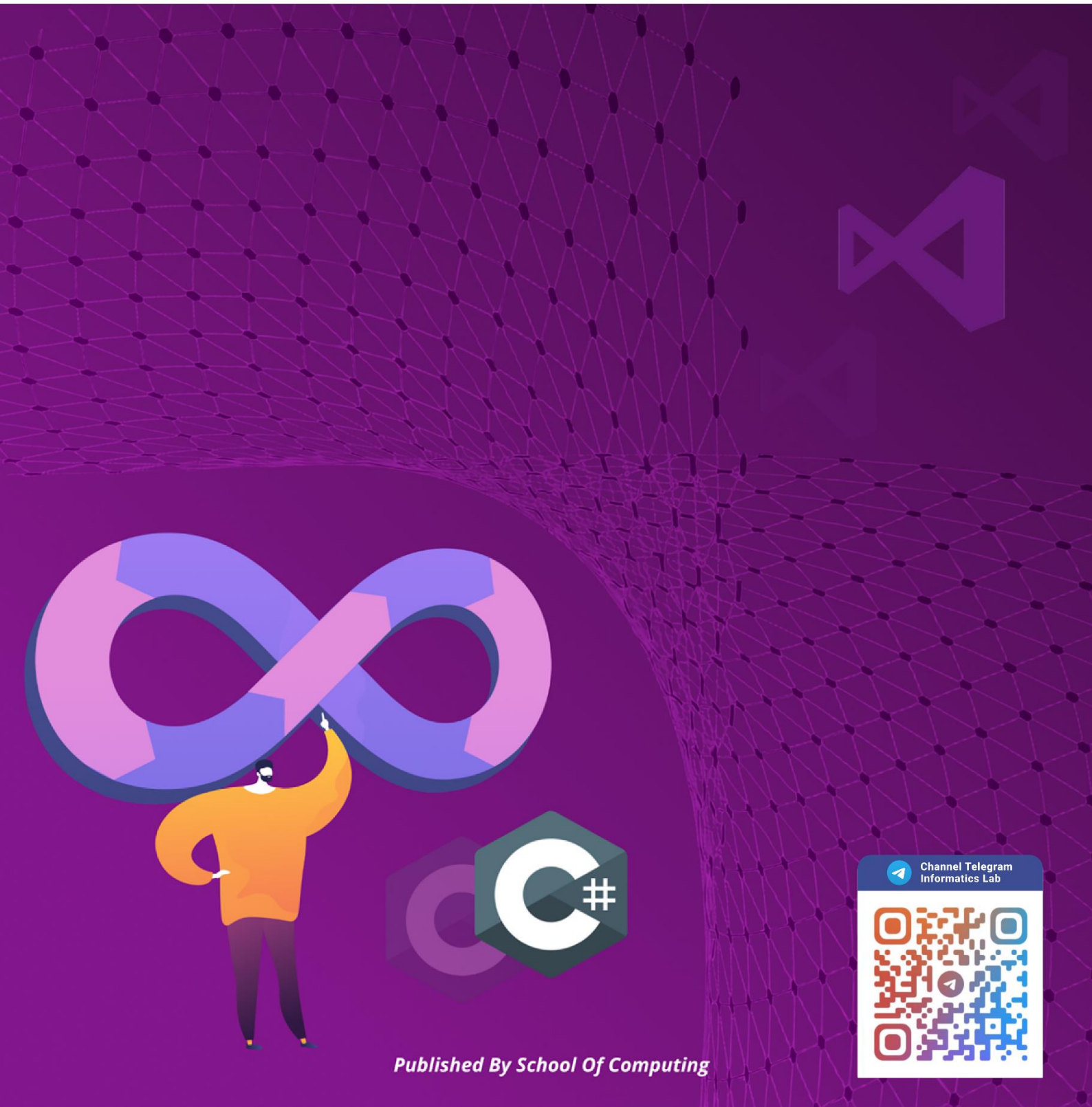


MODUL PRAKTIKUM

KONSTRUKSI PERANGKAT LUNAK

S1 REKAYASA PERANGKAT LUNAK



Published By School Of Computing

LEMBAR PENGESAHAN

Saya yang bertanda tangan di bawah ini:

Nama : Muhammad Johan Alibasa, S.T., M.T., Ph.D
NIK : 21900001
Koordinator Mata Kuliah : Konstruksi Perangkat Lunak
Prodi : S1 Rekayasa Perangkat Lunak

Menerangkan dengan sesungguhnya bahwa modul ini digunakan untuk pelaksanaan praktikum di Semester Genap Tahun Ajaran 2022/2023 di Laboratorium Informatika Fakultas Informatika Universitas Telkom.



Bandung, 19 Februari 2023

Mengesahkan,

Koordinator MK Konstruksi Perangkat Lunak

ACC WA_19-02-2023

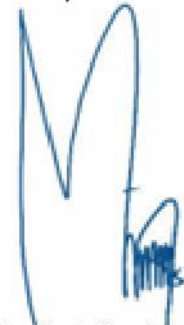


Muhammad Johan Alibasa, S.T., M.T., Ph.D



Mengetahui,

Kaprodi S1 Rekayasa Perangkat Lunak



Dr. Mira Kania Sabariah, S.T., M.T.

Peraturan Praktikum

Laboratorium Informatika 2022/2023

1. Praktikum diampu oleh dosen kelas dan dibantu oleh asisten laboratorium dan asisten praktikum.
2. Praktikan wajib memakai pakaian berkerah dan sesuai aturan institusi selama kegiatan praktikum.
3. Praktikan wajib hadir 10 menit sebelum jadwal praktikum yang ditentukan.
4. Praktikan wajib hadir minimal 75% dari seluruh pertemuan praktikum (sesuai ketentuan institusi).
5. Praktikum dilaksanakan di Gedung Telkom University Lanmark Tower (TULT) Lantai 6 dan Lantai 7 sesuai jadwal yang ditentukan.
6. Durasi kegiatan praktikum S-1 = 2 jam (100 menit).
7. Jumlah pertemuan praktikum sebanyak 16 kali.
8. Dosen berhak melarang praktikan masuk ataupun mengeluarkan praktikan yang tidak mematuhi aturan praktikum.
9. Praktikan yang datang terlambat :
 - ≤ 30 menit : diperbolehkan mengikuti praktikum tanpa tambahan waktu pengerjaan praktikum.
 - >30 menit : tidak diperbolehkan mengikuti praktikum.
10. Saat praktikum berlangsung, asisten praktikum dan praktikan:
 - Wajib menggunakan seragam sesuai aturan institusi.
 - Wajib mematikan/ mengkondisikan semua alat komunikasi.
 - Dilarang membuka aplikasi yang tidak berhubungan dengan praktikum yang berlangsung.
 - Dilarang mengubah pengaturan *software* maupun *hardware* komputer tanpa ijin.
 - Dilarang meninggalkan tempat selama tidak diizinkan oleh asisten.
 - Dilarang bekerjasama atau kecurangan lainnya selama praktikum.
 - Dilarang membawa makanan maupun minuman di ruang praktikum.
 - Dilarang memberikan jawaban ke praktikan lain.
 - Dilarang menyebarkan soal praktikum.
 - Dilarang membuang sampah di ruangan praktikum.
 - Wajib meletakkan alas kaki dengan rapi pada tempat yang telah disediakan.
 - Wajib menjaga sopan santun kepada sesama praktikan maupun asisten.
11. Setiap praktikan dapat mengikuti praktikum susulan maksimal dua modul untuk satu mata kuliah praktikum.
 - Praktikan yang dapat mengikuti praktikum susulan hanyalah praktikan yang memenuhi syarat sesuai ketentuan institusi, yaitu: sakit (dibuktikan dengan surat keterangan medis), tugas dari institusi (dibuktikan dengan surat dinas atau dispensasi dari institusi), atau mendapat musibah atau kedukaan (menunjukkan surat keterangan dari orangtua/wali mahasiswa.)
 - Persyaratan untuk praktikum susulan diserahkan sesegera mungkin kepada asisten laboratorium untuk keperluan administrasi.
 - Praktikan yang diijinkan menjadi peserta praktikum susulan ditetapkan oleh Asman Lab dan Bengkel Informatika dan tidak dapat diganggu gugat.

Konsekuensi Pelanggaran Praktikum

Laboratorium Informatika 2022/2023

1. Keterlambatan menghadiri kelas praktikum (max 30 menit)
 - Tidak diperkenankan mengikuti kegiatan praktikum
2. Ketidakhadiran pada kelas praktikum
 - Absensi dibawah 75% = nilai '0' pada assessment akhir/tubes
3. Meminta, mendapatkan, dan menyebarluaskan soal atau kunci jawaban praktikum
 - Penyebar soal dan kunci jawaban : Pengajuan sanksi kepada Komisi Disiplin Fakultas
 - Penerima soal dan kunci jawaban : Nilai '0' pada (seluruh assessment) praktikum
4. Lupa menghapus file praktikum
 - Pengurangan nilai assessment

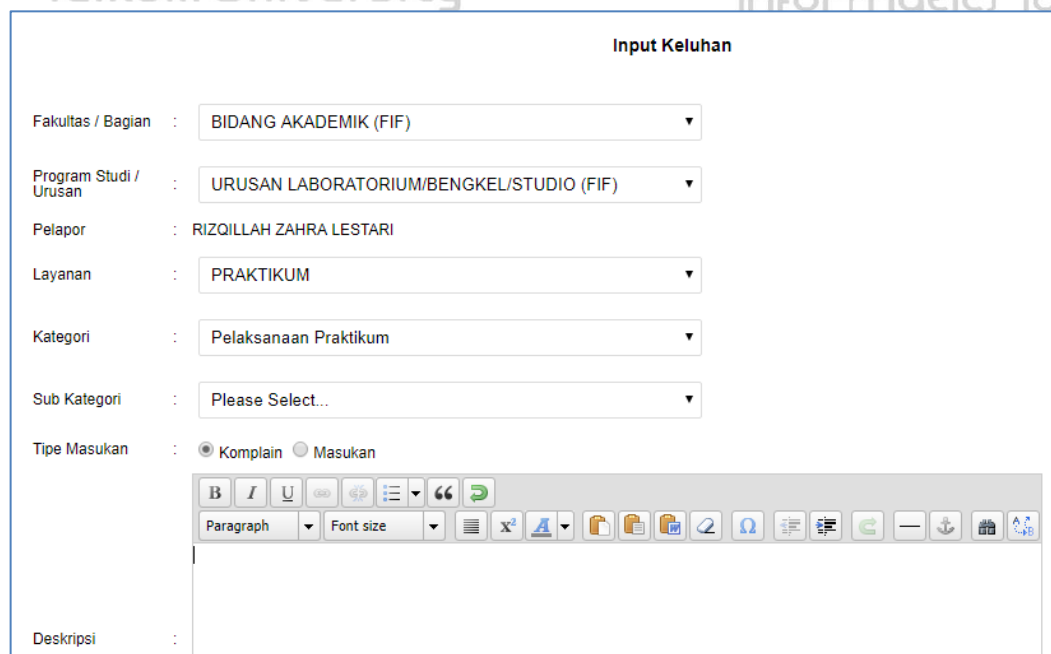


Tata Cara Komplain Praktikum IFLab Melalui IGracias

1. Login IGracias
2. Pilih Menu **Masukan dan Komplain**, pilih **Input Tiket**



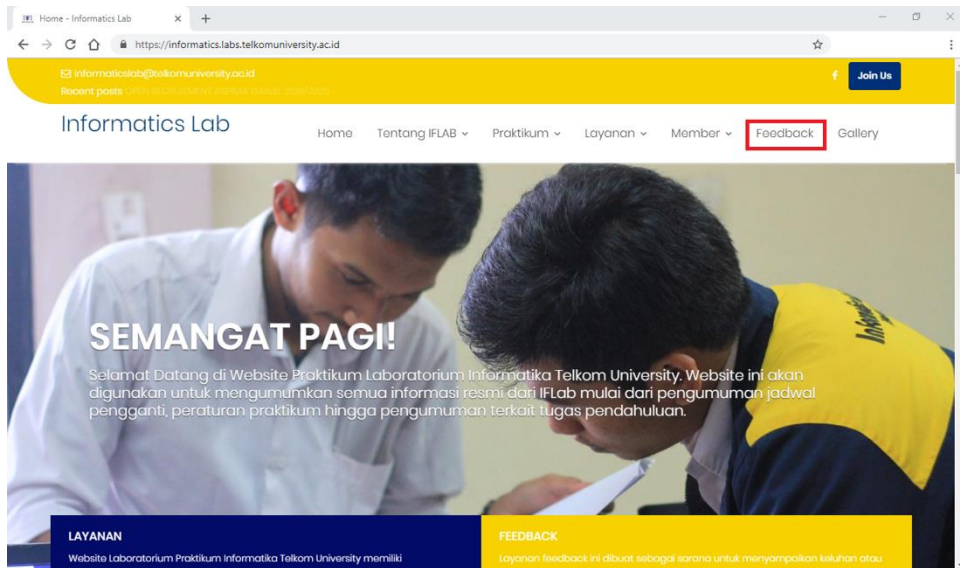
3. Pilih Fakultas/Bagian: **Bidang Akademik (FIF)**
4. Pilih Program Studi/Urusan: **Urusan Laboratorium/Bengkel/Studio (FIF)**
5. Pilih Layanan: **Praktikum**
6. Pilih Kategori: **Pelaksanaan Praktikum**, lalu pilih **Sub Kategori**.
7. Isi **Deskripsi** sesuai komplain yang ingin disampaikan.

A screenshot of a web form titled 'Input Keluhan'. The form contains several dropdown menus and a text area. The fields are: 'Fakultas / Bagian' (set to 'BIDANG AKADEMIK (FIF)'), 'Program Studi / Urusan' (set to 'URUSAN LABORATORIUM/BENGKEL/STUDIO (FIF)'), 'Pelapor' (set to 'RIZQILLAH ZAHRA LESTARI'), 'Layanan' (set to 'PRAKTIKUM'), 'Kategori' (set to 'Pelaksanaan Praktikum'), 'Sub Kategori' (set to 'Please Select...'), and 'Tipe Masukan' (with radio buttons for 'Komplain' and 'Masukan', where 'Komplain' is selected). Below these is a rich text editor with a toolbar containing icons for bold, italic, underline, link, unlink, list, quote, and other formatting options. The 'Deskripsi' field is a large text area at the bottom.

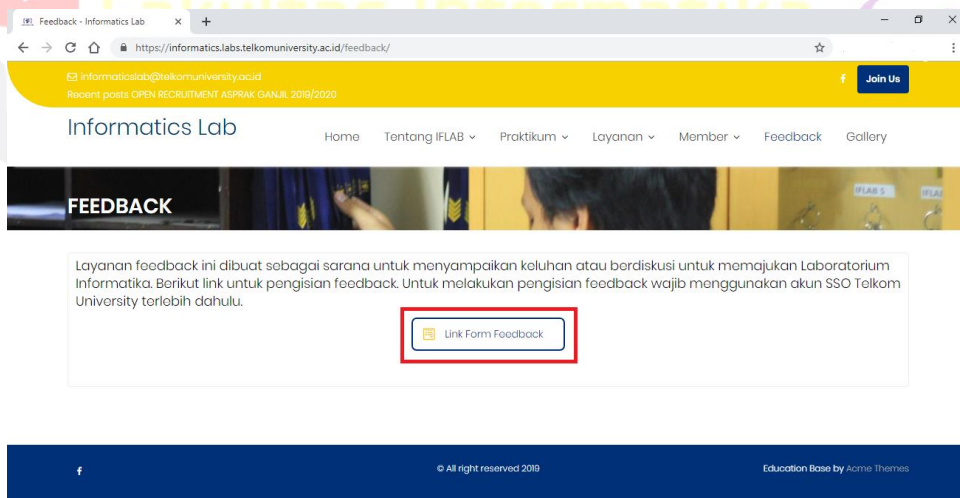
8. Lampirkan *file* jika perlu. Lalu klik Kirim.

Tata Cara Komplain Praktikum IFLAB Melalui Website

1. Buka website <https://informatics.labs.telkomuniversity.ac.id/> melalui browser.
2. Pilih menu **Feedback** pada *navigation bar* website.



3. Pilih tombol **Link Form Feedback**.



4. Lakukan *login* menggunakan akun **SSO Telkom University** untuk mengakses *form feedback*.
5. Isi *form* sesuai dengan *feedback* yang ingin diberikan.

DAFTAR ISI

LEMBAR PENGESAHAN	2
Peraturan Praktikum Laboratorium Informatika 2022/2023	3
Konsekuensi Pelanggaran Praktikum Laboratorium Informatika 2022/2023	4
DAFTAR ISI	7
DAFTAR GAMBAR.....	10
MODUL 1 PENGANTAR PRAKTIKUM KONSTRUKSI PERANGKAT LUNAK	11
1.1 Konstruksi Perangkat Lunak.....	11
1.2 Ekspektasi Capaian.....	11
1.3 Garis Besar Praktikum	12
MODUL 2 PENGENALAN IDE DAN PEMROGRAMAN C#	13
2.1 IDE Visual Studio	13
2.1.1 Instalasi Aplikasi Visual Studio 2022	13
2.1.2 Membuat Project Console Baru	15
2.2 Bahasa Pemrograman C#.....	15
2.2.1 Menerima Input dan Print Output.....	15
2.2.2 Variabel dan Operator	16
2.2.3 Branching dengan IF.....	16
2.2.4 Contoh Array dan For Loop.....	17
MODUL 3 GUI BUILDER DAN GITHUB.....	18
3.1 GUI Builder.....	18
3.1.1 Implementasi C#	19
3.2 Github	22
3.2.1 Instalasi Github	22
3.2.2 Git Commands.....	24
MODUL 4 AUTOMATA DAN TABLE-DRIVEN CONSTRUCTION	25
4.1 Automata Construction.....	25
4.1.1 Implementasi	25
4.1.2 Table-driven Construction	27
4.1.2.1 Direct Access	27
4.1.2.2 Indexed Access.....	27
4.1.2.3 Stair-step Access	27
MODUL 5 GENERICS	29
5.1 Generics	29
5.1.1 Panduan Penamaan Type Paramater	29
5.2 Generic Classes	30
5.3 Generic Methods	31
5.4 Generic Delegate.....	31
MODUL 6 DESIGN BY CONTRACT DAN DEFENSIVE PROGRAMMING	33
6.1 Design by Contract.....	33
6.1.1 Preconditions	34
6.1.2 Postconditions.....	34
6.1.3 Invariants.....	35
6.2 Defensive Programming.....	35

6.2.1	Assertions.....	36
6.2.2	Exceptions Handling.....	36
MODUL 7	GRAMMAR-BASED INPUT PROCESSING (PARSING)	38
7.1	Parsing.....	38
7.2	Parsing String	39
7.3	Parsing JSON	40
MODUL 8	RUNTIME CONFIGURATION DAN INTERNATIONALIZATION	44
8.1	Runtime Configuration.....	44
8.1.1	Runtimeconfig.json	44
8.1.2	MSBuild Properties	45
8.1.3	Environment variables	46
8.2	Internationalization.....	46
8.2.1	Portable Object Localization	47
MODUL 9	API DESIGN DAN CONSTRUCTION USING SWAGGER.....	50
9.1	API	50
9.2	Membuat Sample Project	51
9.3	Menjalankan Sample Project	52
MODUL 10	LIBRARY CONSTRUCTION	54
10.1	Library	54
10.1.1	Membuat Custom Library	55
10.1.2	Menambahkan Custom Library pada Solution	56
MODUL 11	Penilaian Tugas Besar CLO2	60
11.1	Tugas besar	60
11.2	Hal yang harus dipersiapkan	60
MODUL 12	PERFORMANCE ANALYSIS, UNIT TESTING, DAN DEBUGGING	61
12.1	Profiling	61
12.1.1	Profiling CPU usage	61
12.1.2	Profiling Memory usage	64
12.2	Unit Test.....	65
12.2.1	Membuat Unit Test	66
12.2.2	Run Unit Test.....	67
12.3	Debugging	67
MODUL 13	DESIGN PATTERN IMPLEMENTATION	69
13.1	Design patterns	69
13.2	Pattern	69
13.2.1	Singleton	70
13.2.2	Adapter	73
13.2.3	Command.....	75
MODUL 14	CLEAN CODE.....	79
14.1	Clean Code	79
14.1.1	Penamaan Method	80
14.1.2	Penamaan Variable	80
MODUL 15	Review Tugas Besar.....	82
15.1	Tugas besar	82

MODUL 16 Presentasi Tugas Besar.....	84
16.1 Tugas besar	84
16.2 Hal yang harus dipersiapkan	84
DAFTAR PUSTAKA	85
PENUTUP	87
DAFTAR PERUBAHAN	88



DAFTAR GAMBAR

Gambar 2. 1 Menjalankan setup visual studio installer.....	14
Gambar 2. 2 Konfigurasi instalasi visual studio	14
Gambar 2. 3 Menjalankan visual studio.....	15
Gambar 3. 1 Create a new project.	19
Gambar 3. 2 Memilih Windows Forms App (.NET Framework).....	19
Gambar 3. 3 Konfigurasi projek baru.	20
Gambar 3. 4 Menu Toolbox.	20
Gambar 3. 5 Windows properties (Text).	21
Gambar 3. 6 Windows properties (Design).....	21
Gambar 3. 7 Menu File.....	22
Gambar 3. 8 Sign-in Github.....	23
Gambar 3. 9 Masuk Github.....	23
Gambar 4. 1 State Diagram.....	25
Gambar 9. 1 Membuat project baru pada Visual Studio	51
Gambar 9. 2 Memilih versi .NET pada project	52
Gambar 9. 3 Popup SSL ISS Express	52
Gambar 9. 4 Test API dengan Swagger	53
Gambar 9. 5 Hasil Test API dengan Swagger	53
Gambar 10. 1 Ilustrasi aplikasi yang menggunakan libvorbisfile untuk memutar file Ogg Vorbis.	54
Gambar 10. 2 StringLibraryProjects window.	55
Gambar 10. 3 Add Project Reference.	58
Gambar 10. 4 Reference Manager.....	58
Gambar 10. 5 Solution Explorer.	59
Gambar 10. 6 Tombol Debug.	59
Gambar 12. 2 Usage Tools.	61
Gambar 12. 3 Diagnostic Tools.	62
Gambar 12. 4 Record CPU Profile.	62
Gambar 12. 5 Hasil Record CPU Profile.	63
Gambar 12. 6 Penggunaan CPU.	63
Gambar 12. 7 Memory Usage pada Select Tools.	64
Gambar 12. 8 Start Debugging.....	64
Gambar 12. 9 Take Snapshot untuk mengambil snapshot pada awal I sesi debungging.	65
Gambar 12. 10 Reference Manager - HelloWorldTests.....	66
Gambar 12. 11 Run untuk menjalankan program.	68
Gambar 12. 12 Continue untuk kembali menjalankan program.	68
Gambar 12. 13 Local windows.	68
Gambar 13. 1 Contoh struktur singleton.	70
Gambar 13. 2 Output program.	74

MODUL 1 PENGANTAR PRAKTIKUM KONSTRUKSI PERANGKAT LUNAK

TUJUAN PRAKTIKUM
<ol style="list-style-type: none">1. Mengetahui tujuan praktikum MK Konstruksi Perangkat Lunak2. Mengetahui isi praktikum MK Konstruksi Perangkat Lunak

1.1 Konstruksi Perangkat Lunak

Konstruksi adalah bagian penting dari rekayasa perangkat lunak. Konstruksi dilakukan untuk melakukan pembuatan perangkat lunak yang bermakna secara detail. Dalam pelaksanaannya, konstruksi terdiri dari kegiatan coding, verifikasi, unit testing, integration testing dan debugging.

Pada praktikum mata kuliah Konstruksi Perangkat Lunak, mahasiswa akan belajar bagaimana menerapkan teknologi konstruksi perangkat lunak yang sudah dijelaskan di kelas. Pada akhir pertemuan mahasiswa diharapkan dapat menunjukkan penerapan teknologi konstruksi yang benar dan sesuai dengan kebutuhan.

1.2 Ekspektasi Capaian

Setelah mengikuti praktikum ini, mahasiswa diharapkan memiliki kemampuan sebagai berikut:

1. Mampu mengoperasikan dan memanfaatkan IDE, version control, dan GUI Builder sebagai tools konstruksi.
2. Mampu menerapkan beragam teknologi konstruksi PL.
3. Mampu melakukan validasi dan verifikasi code melalui unit testing, debugging, dan performance analysis.
4. Mampu menghasilkan kode yang berkualitas melalui penerapan design pattern dan coding standard.

Kemampuan ini diturunkan dari CLO yang ditentukan untuk mata kuliah Konstruksi Perangkat Lunak.

1.3 Garis Besar Praktikum

Modul	Materi besar	Materi
1	Pengantar praktikum	Running module
2	Tools konstruksi	Pengenalan IDE dan Pemrograman C#
3		GUI Builder, dan Github
4	Teknologi konstruksi	Automata and table-based construction
5		Generics
6		Design by contract and defensive programming
7		Grammar-based input processing
8		Runtime configuration
9		API Design
10		Library construction
11	Assessment	Penilaian Tugas Besar CLO2
12	Validasi dan verifikasi code	Unit test, performance analysis, dan debugging
13	Code quality	Design pattern
14		Clean code
15	Tugas besar	Review tugas besar
16		Presentasi akhir

MODUL 2 PENGENALAN IDE DAN PEMROGRAMAN C#

TUJUAN PRAKTIKUM
<ol style="list-style-type: none">1. Menguasai penggunaan IDE yaitu Visual Studio2. Menguasai penggunaan bahasa pemrograman C#

2.1 IDE Visual Studio

Visual studio merupakan salah satu IDE (*Integrated Development Environment*) yang dapat membantu dalam mengembangkan kode perangkat lunak secara efisien. IDE bertujuan untuk meningkatkan produktivitas pengembang dengan menggabungkan kemampuan seperti pengeditan, pembuatan, pengujian, dan pengemasan perangkat lunak dalam aplikasi yang mudah digunakan.

Text/code editor, seperti Notepad++, Sublime, Visual Studio Code (berbeda dengan Visual Studio, walaupun namanya mirip), pada umumnya digunakan untuk membantu menuliskan kode program saja. Dengan menggunakan IDE Visual Studio, berikut keuntungan yang dapat diperoleh:

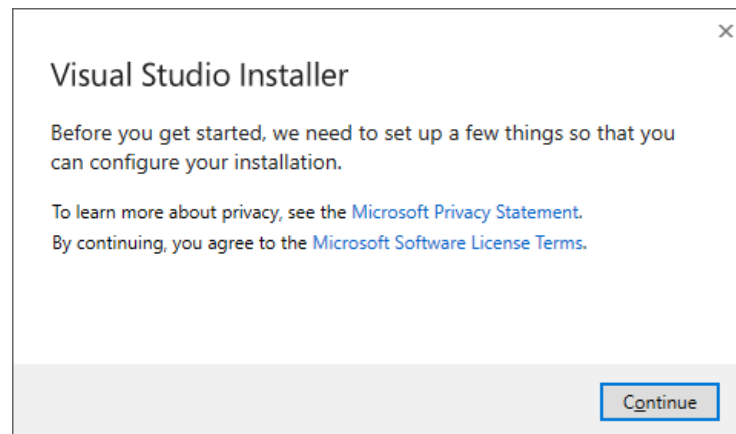
1. Mendeteksi kesalahan *syntax* tanpa harus melakukan proses *compile*
2. Memberikan rekomendasi perbaikan jika terdapat *syntax* yang bermasalah
3. Menyederhanakan proses pengembangan aplikasi GUI dengan tampilan “drag and drop”
4. Membantu proses pengujian termasuk unit testing dan performance testing
5. Mempermudah proses penambahan library eksternal yang lebih mudah
6. Melakukan instalasi semua framework wajib yang diperlukan untuk mengembangkan aplikasi

2.1.1 Instalasi Aplikasi Visual Studio 2022

Berikut langkah-langkah yang perlu dilakukan untuk melakukan instalasi aplikasi Visual Studio 2022 berdasarkan informasi yang diberikan pada halaman web Microsoft :

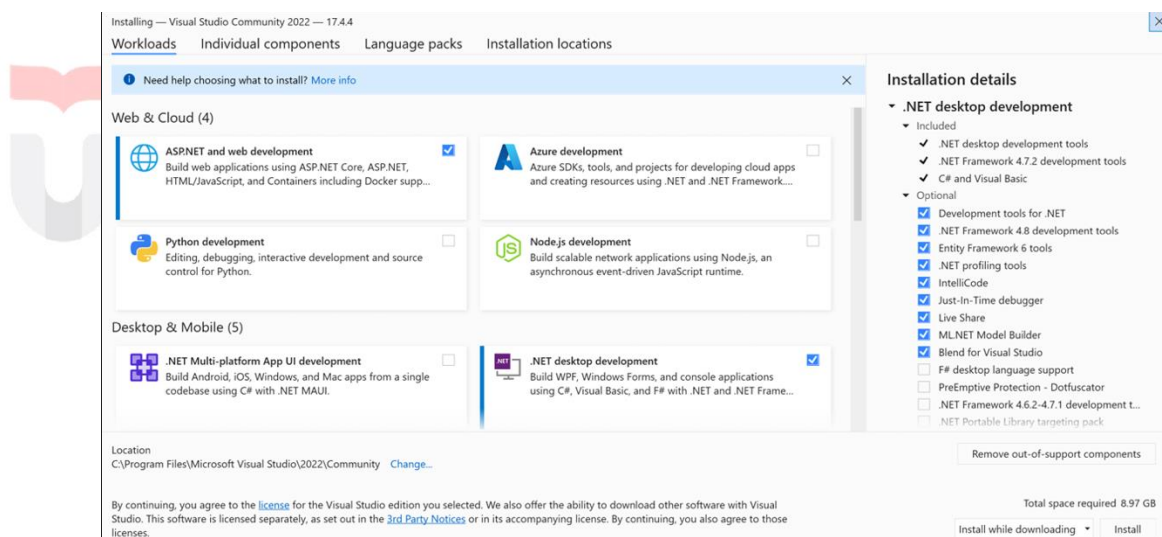
1. Memastikan laptop/komputer siap untuk Visual Studio (cek syarat instalasi pada halaman berikut:
<https://learn.microsoft.com/en-us/visualstudio/releases/2022/system-requirements>
2. Mengunduh aplikasi Visual Studio pada halaman berikut (misal Community Edition):
<https://visualstudio.microsoft.com/downloads>

3. Menjalankan aplikasi setup yang telah diunduh sebelumnya:



Gambar 2. 1 Menjalankan setup visual studio installer

4. Melakukan konfigurasi instalasi, pastikan untuk mencentang “.NET Desktop Development” dan “web development” (untuk modul API). Opsi yang lain merupakan opsional jadi tidak wajib dipilih dan tergantung pada kebutuhan.

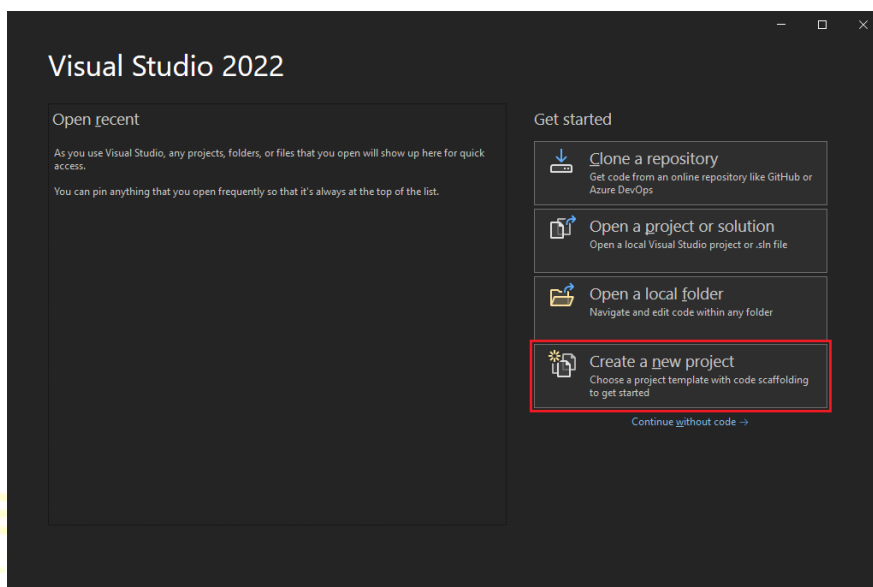


Gambar 2. 2 Konfigurasi instalasi visual studio

2.1.2 Membuat Project Console Baru

Pada praktikum modul ini, akan diperlukan project dengan console (tampilan teks). Misalnya ingin dilakukan hasil print “Hello World” pada console (misal pada terminal atau command prompt). Berikut langkah-langkah yang dapat dilakukan untuk membuat project console baru pada Visual Studio:

1. Menjalankan aplikasi Visual Studio kemudian memilih “Create a new project”



Gambar 2. 3 Menjalankan visual studio

2. Pada halaman berikutnya, pilih Bahasa C# pada drop down menu dan pilih Console pada pilihan “All project types”.
3. Masukkan nama project dan direktori penyimpanan, tekan tombol Next. Pilih framework manapun yang ter-install pada komputer/laptop.
4. Tekan tombol Run (symbol segitiga berwarna hijau) untuk menampilkan “Hello World”

2.2 Bahasa Pemrograman C#

Pada kuliah dan praktikum Konstruksi Perangkat Lunak akan digunakan Bahasa Pemrograman C#. Pada dasarnya, Bahasa pemrograman ini sangat mirip dengan Bahasa Pemrograman Java, dan Java mirip dengan Bahasa C/C++.

2.2.1 Menerima Input dan Print Output

Pada subbab sebelumnya, diberikan contoh pembuatan project console. Ubah kode pada project tersebut sehingga dapat menerima input dan melakukan print output.

```
// Ini adalah komentar untuk melakukan print  
Console.WriteLine("Enter username:");
```

```
// Membuat variabel username untuk menerima input user
string userName = Console.ReadLine();

// Melakukan print panjang string input
Console.WriteLine("The length of the input string is: " + userName.Length);

// Melakukan print dengan menggabungkan string dengan input user
Console.WriteLine("Username is: " + userName);
```

2.2.2 Variabel dan Operator

Di C#, ada berbagai jenis variabel (didefinisikan dengan kata kunci yang berbeda), misalnya "int", "double", "char", "string" dan "bool". Buka halaman web berikut untuk mengetahui semua jenis tipe data pada Bahasa C#: <https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/built-in-types>. Perlu dicatat bahwa operator pada Bahasa C# mirip dengan Java dan C, misalnya +, -, / dan seterusnya.

```
int myNum = 5;
myNum = myNum + 3;
int myNum2 = 3 * 4;
double myDoubleNum = 5.99D;
char myLetter = 'D';
bool myBool = true;
string myText = "Hello";
```

2.2.3 Branching dengan IF

Proses percabangan dari beberapa kondisi dapat menggunakan IF:

```
int time = 22;
if (time < 10)
{
    Console.WriteLine("Good morning.");
}
else if (time < 20)
{
    Console.WriteLine("Good day.");
}
else
{
    Console.WriteLine("Good evening.");
}
// Outputs "Good evening."
```

2.2.4 Contoh Array dan For Loop

Contoh iterasi elemen array dengan for loop, dan menggunakan properti "Length" untuk menentukan berapa kali loop harus dijalankan.

```
string[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
for (int i = 0; i < cars.Length; i++)
{
    Console.WriteLine(cars[i]);
}
```



MODUL 3 GUI BUILDER DAN GITHUB

TUJUAN PRAKTIKUM
3. Menguasai penggunaan github untuk version control
4. Menguasai penggunaan GUI Builder yang tersedia pada MS Visual Studio

3.1 GUI Builder

Graphical User Interface Builder (GUI builder) yang juga dikenal sebagai GUI designer adalah alat pengembangan perangkat lunak yang menyederhanakan pembuatan GUI dengan memungkinkan perancang untuk mengatur elemen kontrol grafis (widget) menggunakan drag-and-drop WYSIWYG editor. Tanpa adanya GUI Builder, GUI harus dibangun dengan secara manual menentukan parameter setiap widget dalam source-code, tanpa umpan balik visual sampai program dijalankan.

User interfaces biasanya diprogram menggunakan arsitektur event-driven architecture, jadi GUI builders juga menyederhanakan pembuatan kode berbasis event-driven. Kode pendukung ini menghubungkan widget dengan input dan output yang memicu fungsi yang menyediakan logika aplikasi.

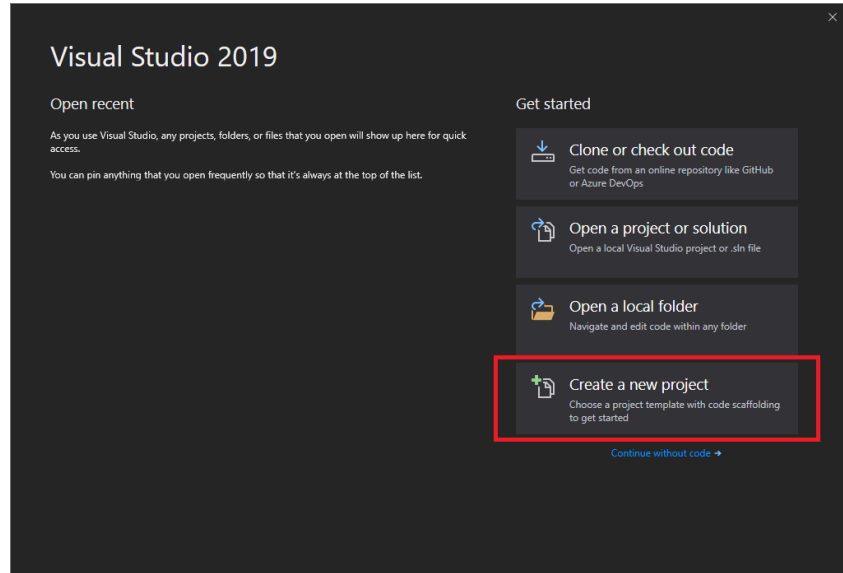
Beberapa graphical user interface builders, seperti Glade Interface Designer, secara otomatis menghasilkan semua source-code untuk elemen kontrol grafis. Lainnya, seperti Interface Builder, menghasilkan instance serialized object yang kemudian dimuat oleh aplikasi. Berikut ini beberapa contoh GUI Builders:

1. GTK+ / Glade Interface Designer
2. XForms (toolkit)
3. UWP / Windows Presentation Foundation / WinForms
4. Microsoft Visual Studio XAML Editor, XAML based GUI layout
5. SharpDevelop
6. Xamarin Studio
7. C++Builder / VCL (Visual Component Library)
8. Qt Creator / Qt
9. FLTK
10. wxWidgets
11. wxGlade
12. wxFormBuilder
13. wxCrafter (plugin for CodeLite)
14. Projucer
15. Android Studio, XML based GUI layout
16. NetBeans GUI design tool
17. 1Apache Cordova / PhoneGap

3.1.1 Implementasi C#

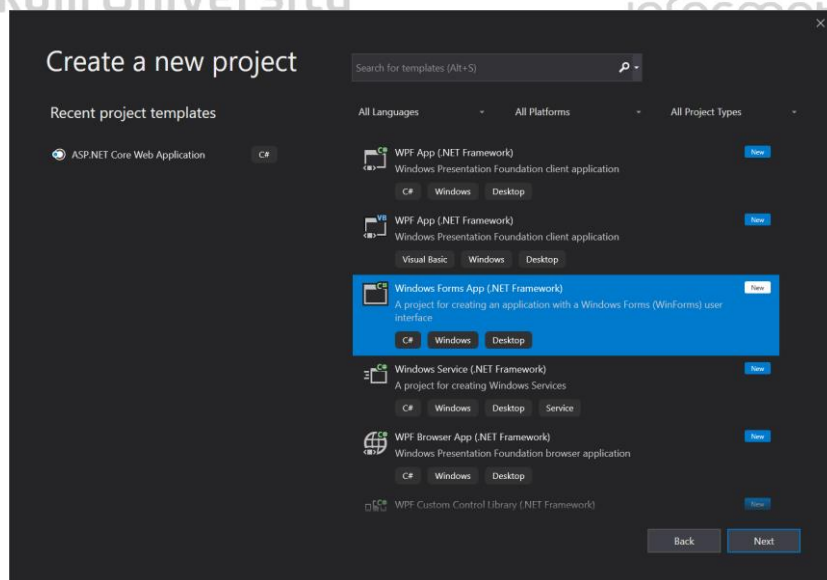
Membuat Project Baru :

1. Buka Visual Studio.
2. Pada start window, Pilih **Create a new project**.



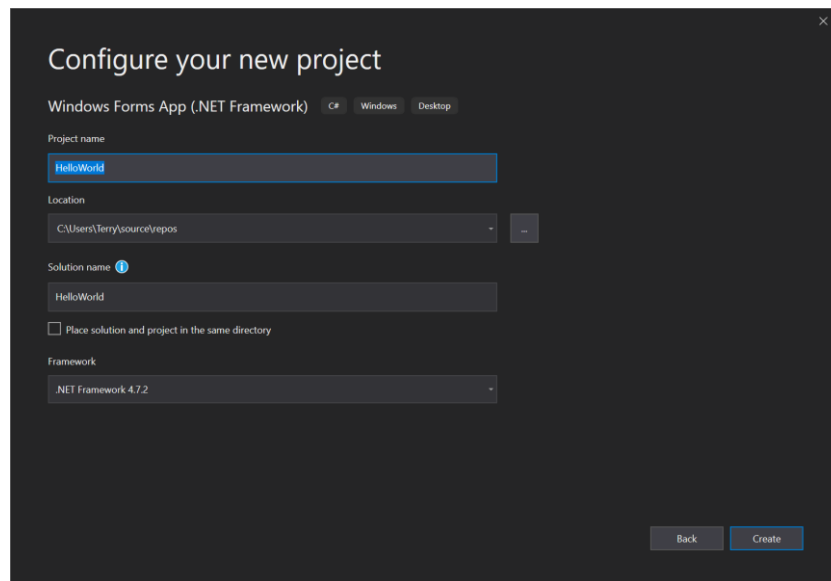
Gambar 3. 1 Create a new project.

3. Pada **Create a new project** window, Pilih **Windows Forms App (.NET Framework)** template for C#.



Gambar 3. 2 Memilih Windows Forms App (.NET Framework).

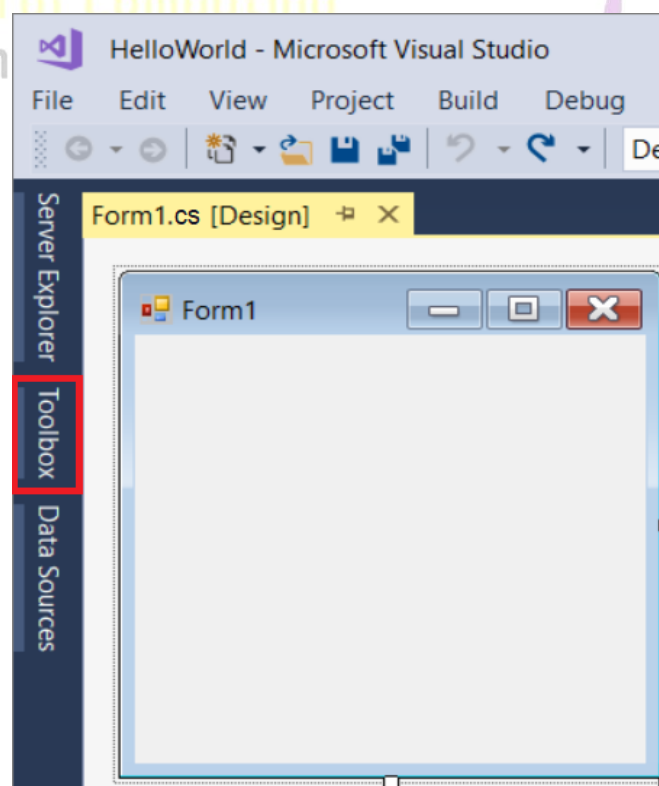
4. Pada **Configure your new project window**, Tuliskan *HelloWorld* sebagai nama project. Lalu click create.



Gambar 3. 3 Konfigurasi proyek baru.

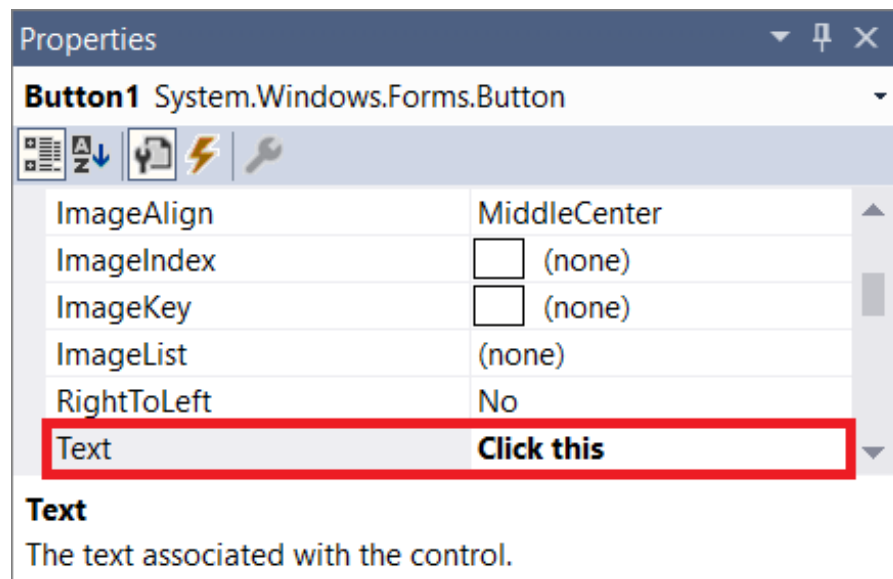
Menambahkan button / UI Control pada form

1. Klik **Toolbox** untuk membuka Toolbox fly-out window.



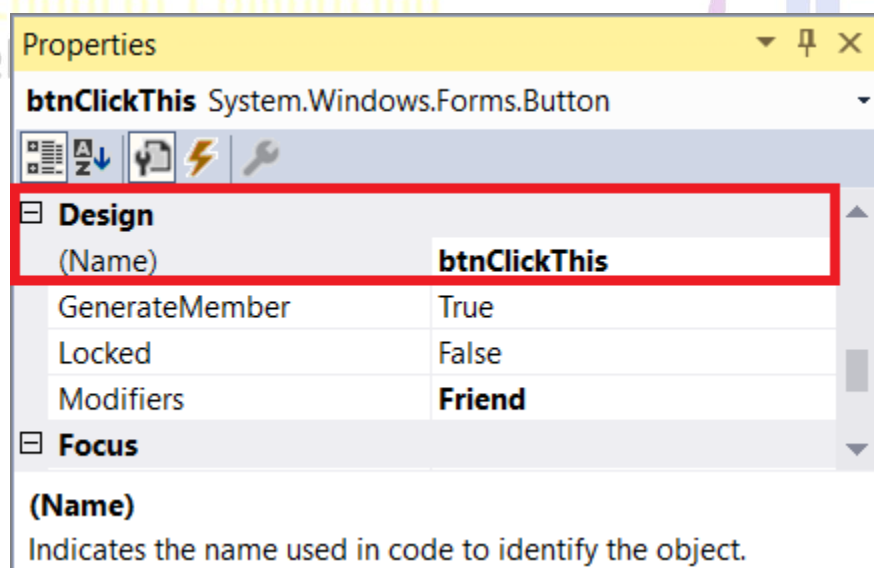
Gambar 3. 4 Menu Toolbox.

2. Pilih **Button / UI Control** yang ingin ditambahkan dan drag pada form.
3. Untuk Mengganti Text yang ditampilkan pada Forms, dapat dilakukan dengan mengklik UI Control dan pada menu **Properties** window, ganti tulisan pada bagian **Text**.



Gambar 3. 5 Windows properties (Text).

4. Untuk Mengganti nama UI Control pada Forms, dapat dilakukan dengan mengklik UI Control dan pada menu **Properties** window, Change name pada section **Design**.



Gambar 3. 6 Windows properties (Design).

Menambahkan Code pada Form

5. Pada **Form1.cs [Design]** window, double-click **Click this** button untuk membuka **Form1.cs** window.
6. Pada **Form1.cs** window (Form yang dipilih terbuka), Silahkan tambahkan code yang diperlukan pada file C# tersebut.

3.2 Github

GitHub adalah layanan hos web bersama untuk proyek pengembangan perangkat lunak yang menggunakan sistem kendali versi Git dan layanan hosting internet. Hal ini banyak digunakan untuk kode komputer. Ini memberikan kontrol akses dan beberapa fitur kolaborasi seperti pelacakan bug, permintaan fitur, manajemen tugas, dan wiki untuk setiap proyek.

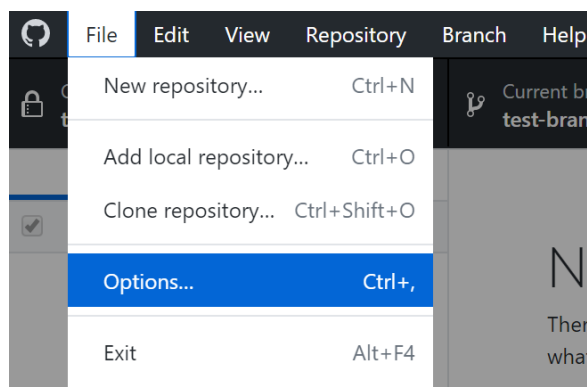
GitHub menawarkan paket repositori pribadi dan gratis pada akun yang sama dan digunakan untuk proyek perangkat lunak sumber terbuka. Pada bulan April 2017, GitHub melaporkan bahwa mereka mempunyai lebih dari 20 juta pengguna dan lebih dari 57 juta repositori, menjadikannya layanan terbesar dari kode sumber di dunia.

GitHub mempunyai sebuah maskot yang bernama Octocat, seekor kucing dengan lima tentakel dan wajah seperti manusia.

3.2.1 Instalasi Github

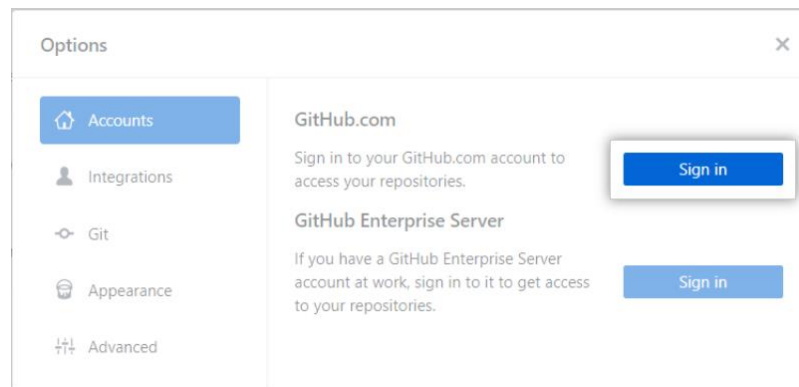
Berikut ini cara menginstall github:

1. Download git pada laman <https://git-scm.com/downloads>
2. Lakukan instalasi git dengan mengikuti default installation.
3. Download github desktop pada laman <https://desktop.github.com/>
4. Lakukan instalasi github desktop dengan membuka file installer yang tadi di download.
5. Buka Github desktop, lalu pilih file kemudian option.



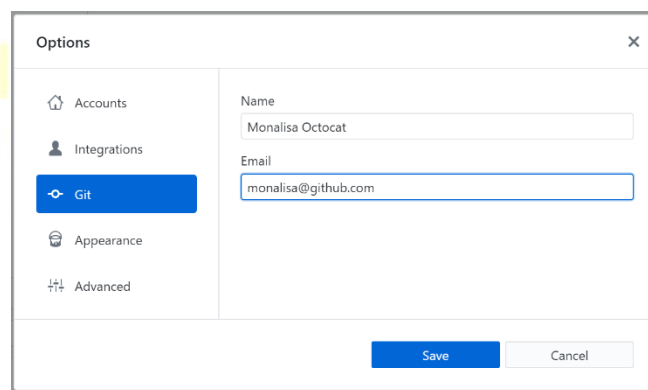
Gambar 3. 7 Menu File.

6. Pada windows options pilih accounts.
7. Klik button sign in pada kanan "GitHub.com".



Gambar 3. 8 Sign-in Github.

8. Lakukan Sign in akun github
9. Pada windows options pilih Git
10. Masukkan nama dan email yang digunakan pada akun github



Gambar 3. 9 Masuk Github.

11. Klik button save.

3.2.2 Git Commands

Berikut ini beberapa command yang biasa digunakan:

Command	Fungsi
<code>git init</code>	membuat repository baru.
<code>git clone</code>	melakukan cloning repository berdasarkan url githubnya.
<code>git add</code>	menambahkan file ke dalam repository lokal
<code>git commit</code>	menyimpan perubahan pada repository lokal
<code>git push</code>	menyimpan commit pada online repository
<code>git pull</code>	mengecek dan menggabungkan update pada online repository dengan lokal repository
<code>git fetch</code>	mengecek update pada online repository



Fakultas Informatika
School of Computing
Telkom University



informatics lab

MODUL 4 AUTOMATA DAN TABLE-DRIVEN CONSTRUCTION

TUJUAN PRAKTIKUM

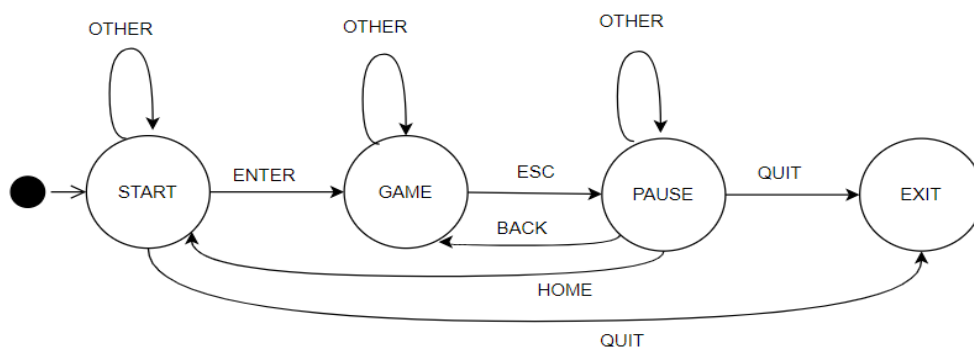
1. Menguasai penerapan automata-based construction pada C#.Net
2. Menguasai penerapan table-based construction pada C#.Net

4.1 Automata Construction

Automata Construction atau Automata-based programming adalah salah satu paradigma pemrograman dimana program dianggap seperti finite-state machine(FSM) atau formal automaton lainnya yang memiliki berbagai state-state yang saling berkaitan dan memiliki aturan tertentu yang jelas. Berikut ini indikator utama dalam Automata-based programming:

1. Jangka waktu eksekusi program dipisahkan dengan jelas pada state yang ada dan tidak terjadinya eksekusi yang overlapping pada state state yang ada.
2. Semua komunikasi antara state-state yang ada (perpindahan antar state) hanya dapat dilakukan secara eksplisit yang disimpan pada suatu global variable.

4.1.1 Implementasi



Gambar 4. 1 State Diagram.

Gambar diatas merupakan state diagram dari contoh source code Automata-based programming dibawah ini.

```
using system;
public class Program
{
    enum State {START, GAME, PAUSE, EXIT};
    public static void Main()
    {
```

```

State state = State.START;
string[] screenName = {"START", "GAME", "PAUSE", "EXIT"};
while (state != State.EXIT)
{
    Console.WriteLine(screenName[(int)state] + " SCREEN");
    Console.Write("Enter Command : ");

    string command = Console.ReadLine();
    switch (state)
    {
        case State.START:
            if(command == "ENTER")
                state = State.GAME;
            else if (command == "QUIT")
                state = State.EXIT;
            else
                state = State.START;
            break;
        case State.GAME:
            if(command == "ESC")
                state = State.PAUSE;
            else
                state = State.GAME;
            break;
        case State.PAUSE:
            if(command == "BACK")
                state = State.GAME;
            else if (command == "HOME")
                state = State.START;
            else if (command == "QUIT")
                state = State.EXIT;
            else
                state = State.PAUSE;
            break;
    }
}
Console.WriteLine("EXIT SCREEN");
}
}

```


4.1.2 Table-driven Construction

Table-driven Construction adalah skema yang memungkinkan mencari informasi menggunakan table dibandingkan menggunakan logic statements (if dan case) untuk mengetahuinya. Hampir semua hal yang dapat ditangani oleh if dan case, dapat diubah menjadi tabel-driven sebagai gantinya. Dalam kasus sederhana, pernyataan logika lebih mudah dan lebih langsung. Saat logic statements sudah menjadi begitu kompleks, penggunaan table-driven akan menjadi semakin menarik.

Table-driven memiliki dua hal yang perlu diperhatikan sebelum diimplementasikan. Pertama kita perlu menentukan bagaimana mencari entri pada tabel. Kedua beberapa tipe data tidak bisa digunakan untuk mencari entri pada table secara langsung. Secara umum cara pencarian entri pada table-driven dibagi menjadi tiga yaitu :

1. Direct Access.
2. Indexed Access.
3. Stair-step Access.

4.1.2.1 Direct Access

Secara sederhana direct access adalah metode table-driven yang secara langsung yang mengubah kondisi pada logic statement menjadi key dari table yang digunakan. Untuk lebih jelasnya perhatikan potongan kode berikut.

```
public static int GetDaysPerMonth(int month){  
    int[] daysPerMonth = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30,  
31 };  
    return daysPerMonth[month - 1];  
}
```

4.1.2.2 Indexed Access

Indexed Acces adalah metode table-driven yang menggunakan table lain untuk menyimpan index dari table utama, hal ini bertujuan untuk mengurangi penggunaan memori penyimpanan dari program apabila ukuran satu dari entri table utama besar dan memiliki nilai entri yang berulang.

4.1.2.3 Stair-step Access

Salah satu metode dalam table-driven yang berguna untuk mempermudah pencarian entri apabila terdapat kasus dimana nilai yang didapatkan berdasarkan range yang telah ada seperti contoh index nilai mahasiswa. Untuk lebih jelasnya perhatikan potongan kode berikut.

```
public static string GetGradeByScore(double studentScore){
    string[] grade = { "A", "AB", "B", "BC", "C", "D", "E" };
    double[] rangeLimit = { 80.0, 70.0, 65.0, 60.0, 50.0, 40.0, 0.0};
    int maxGradeLevel = grade.Length - 1;

    string studentGrade = "E";
    int gradeLevel = 0;
    while((studentGrade == "E") && (gradeLevel < maxGradeLevel)){
        if( studentScore > rangeLimit[gradeLevel])
            studentGrade = grade[gradeLevel];
        gradeLevel = gradeLevel + 1;
    }

    return studentGrade;
}
```



Fakultas Informatika
School of Computing
Telkom University



MODUL 5 GENERICS

TUJUAN PRAKTIKUM

1. Menguasai penggunaan generics untuk parameterisasi pada C#.Net

5.1 Generics

Generics adalah konsep dalam pemrograman dimana algoritma atau code di tulis dalam suatu istilah tertentu (biasanya T) yang akan ditentukan kemudian, code tersebut kemudian dipakai saat diperlukan untuk jenis tertentu yang disediakan sebagai parameter. Berdasarkan hal tersebut, Generic memungkinkan untuk mendesain class dan method yang dapat digunakan oleh berbagai jenis type data bergantung pada saat class dan method tersebut digunakan atau dideklarasikan oleh programmer sebagai objek. Sebagai contoh pada saat kita akan mendeklarasikan list, kita akan memberikan type data pada list sebelum digunakan. Hal itu terjadi karena list sendiri dibangun secara generic.

```
List<int> listOne = new List <int>();  
List <float> listTwo = new List <float>();
```

Dengan menggunakan generic dalam suatu code, hal itu membuat code tadi menjadi lebih reusability, aman dan efisien dibandingkan yang tidak menggunakan generic. Generics biasanya dipakai pada collection dan method yang berkaitan dengan collection. Generic dapat digunakan juga pada interface, classes, method, events dan delegate. Pada Generic class, memungkinkan kita untuk mengakses beberapa method pada type T yang digunakan pada class tersebut. Sebagai informasi tambahan, data type yang digunakan oleh generic akan ditentukan pada saat runtime program menggunakan metode reflection.

Pada Generic terdapat Type Paramater, Type Paramater adalah placeholder untuk tipe tertentu yang ditentukan klien saat mereka membuat instance tipe generik. Seperti contoh berikut:

```
GenericList<float> list1 = new GenericList<float>();  
GenericList<ExampleClass> list2 = new GenericList<ExampleClass>();  
GenericList<ExampleStruct> list3 = new GenericList<ExampleStruct>();
```

5.1.1 Panduan Penamaan Type Paramater

- Beri nama parameter tipe generik dengan nama deskriptif, kecuali jika satu nama huruf sudah cukup jelas dan nama deskriptif tidak akan menambah nilai.

```
public interface ISessionChannel<TSession> { /*...*/ }
public delegate TOutput Converter<TInput, TOutput>(TInput from);
public class List<T> { /*...*/ }
```

- Pertimbangkan untuk menggunakan T sebagai nama parameter tipe untuk tipe dengan satu parameter tipe huruf.

```
public int IComparer<T>() { return 0; }
public delegate bool Predicate<T>(T item);
public struct Nullable<T> where T : struct { /*...*/ }
```

- Lakukan awalan nama parameter tipe deskriptif dengan "T".

```
public interface ISessionChannel<TSession>
{
    TSession Session { get; }
}
```

- Pertimbangkan untuk menunjukkan batasan yang ditempatkan pada parameter tipe dalam nama parameter. Misalnya, parameter yang dibatasi untuk **ISession** dapat disebut **TSession**.

5.2 Generic Classes

Generic class encapsulate operasi yang tidak spesifik untuk tipe data tertentu. Penggunaan paling umum untuk Generic class dengan collections like linked lists, hash tables, stacks, queues, trees, dan sebagainya. Operasi seperti menambah dan menghapus item dari collections pada dasarnya dilakukan dengan cara yang sama terlepas dari jenis data yang disimpan. Berikut ini beberapa pertimbangan penting dalam membuat class generic:

1. Tipe mana yang akan digeneralisasi menjadi type parameters.
Biasanya, semakin banyak type parameters pada class, semakin fleksibel dan dapat digunakan kembali class tersebut. Namun, terlalu banyak generalisasi dapat membuat kode yang sulit dibaca atau dipahami oleh orang lain.
2. Tentukan constraints, jika ada, untuk diterapkan pada parameter tipe.
Constraints yang baik adalah menerapkan batasan maksimum yang memungkinkan tetap menangani tipe yang seharusnya bisa ditangani.
3. Apakah akan menerapkan perilaku generik ke dalam base classes and subclasses.
Karena kelas generik dapat berfungsi sebagai kelas dasar, pertimbangan desain yang sama berlaku di sini seperti pada kelas non-generik

4. Apakah akan mengimplementasikan satu atau lebih generic interfaces.

Berikut ini adalah contoh code dari generic class.

```
class BaseNode { }  
class BaseNodeGeneric<T> { }  
  
// concrete type  
class NodeConcrete<T> : BaseNode { }  
  
//closed constructed type  
class NodeClosed<T> : BaseNodeGeneric<int> { }  
  
//open constructed type  
class NodeOpen<T> : BaseNodeGeneric<T> { }
```

5.3 Generic Methods

Generic Methods adalah method yang dideklarasikan dengan parameter tipe, sebagai berikut:

```
static void Swap<T>(ref T lhs, ref T rhs)  
{  
    T temp;  
    temp = lhs;  
    lhs = rhs;  
    rhs = temp;  
}
```

5.4 Generic Delegate

Delegate dapat menentukan parameter tipenya sendiri atau delegate juga dapat menggunakan generic. code yang mereferensikan generik delegate dapat menentukan argumen type untuk membuat closed constructed type, seperti saat membuat instance kelas generik atau memanggil metode generik, seperti yang ditunjukkan pada contoh berikut:

```
public delegate void Del<T>(T item);  
public static void Notify(int i) { }  
Del<int> m1 = new Del<int>(Notify);
```

Berikut ini adalah source code implentasi generic pada C# :

```
using System;
using System.Collections;
using System.Collections.Generic;

public class Program
{
    public static void Main()
    {
        int[] arr = { 0, 1, 2, 3, 4 };
        List<int> list = new List<int>();
        for (int x = 5; x < 10; x++)
        {
            list.Add(x);
        }
        ProcessItems<int>(arr);
        ProcessItems<int>(list);
    }
    static void ProcessItems<T>(ICollection coll)
    {
        // IsReadOnly returns True for the array and False for the
        List.
        System.Console.WriteLine
            ("IsReadOnly returns {0} for this collection.",
            coll.IsReadOnly);
        // The following statement causes a run-time exception for the
        // array, but not for the List.
        //coll.RemoveAt(4);

        foreach (T item in coll)
        {
            System.Console.Write(item.ToString() + " ");
        }
        System.Console.WriteLine();
    }
}
```


MODUL 6 DESIGN BY CONTRACT DAN DEFENSIVE PROGRAMMING

TUJUAN PRAKTIKUM
1. Menguasai penggunaan CodeContract pada C#.Net
2. Menguasai penerapan defensive programming pada C#.Net

6.1 Design by Contract

Design by Contract (DbC), yang juga dikenal sebagai contract programming, programming by contract and design-by-contract programming, adalah suatu pendekatan dalam merancang perangkat lunak. DbC mengatur bahwa perancang perangkat lunak harus menentukan spesifikasi antarmuka yang formal, tepat dan dapat diverifikasi untuk komponen perangkat lunak, yang memperluas definisi biasa dari tipe data abstrak dengan preconditions, postconditions and invariants. Spesifikasi ini disebut sebagai "kontrak", sesuai dengan metafora konseptual dengan kondisi dan kewajiban kontrak bisnis.

Pendekatan DbC mengasumsikan semua komponen klien yang menjalankan operasi pada komponen server akan memenuhi prasyarat yang ditentukan seperti yang diperlukan untuk operasi itu. Jika asumsi ini dianggap terlalu berisiko (seperti dalam kasus multi-channel or distributed computing), pendekatan terbalik diambil, yang berarti bahwa komponen server menguji bahwa semua prasyarat yang relevan dipegang benar (sebelum, atau saat, memproses permintaan komponen klien) dan membalas dengan pesan kesalahan yang sesuai jika tidak.

Berikut ini keuntungan menggunakan Code contracts (C# DbC tools):

1. Meningkatkan testing: Code contracts menyediakan verifikasi kontrak statis, pemeriksaan waktu proses, dan pembuatan dokumentasi.
2. Automatic testing tools: Anda dapat menggunakan kontrak kode untuk menghasilkan pengujian unit yang lebih bermakna dengan memfilter argumen pengujian yang tidak berarti yang tidak memenuhi preconditions.
3. Static verification: Pemeriksa statis dapat memutuskan apakah ada pelanggaran kontrak tanpa menjalankan program. Ia memeriksa kontrak implisit, seperti dereferensi null dan batas array, dan kontrak eksplisit.
4. Reference documentation: Generator dokumentasi menambah file dokumentasi XML yang ada dengan informasi kontrak. Ada juga lembar gaya yang dapat digunakan dengan Sandcastle sehingga halaman dokumentasi yang dihasilkan memiliki bagian kontrak.

6.1.1 Preconditions

Pada code contract cara mengekspresikan preconditions adalah dengan memanggil method `Contract.Requires`. preconditions menentukan status saat method dipanggil. preconditions umumnya digunakan untuk menentukan nilai parameter yang valid. Semua anggota yang disebutkan dalam preconditions harus setidaknya dapat diakses oleh method itu sendiri; jika tidak, preconditions mungkin tidak dipahami oleh semua yang memanggil method. Kondisi tersebut harus tidak memiliki efek samping. Perilaku run-time dari prasyarat yang gagal ditentukan oleh runtime analyzer. Sebagai contoh, perhatikan potongan source code dibawah dimana terdapat preconditions paramter x tidak bernilai null.

```
Contract.Requires(x != null);
```

Jika kode harus menampilkan pengecualian tertentu pada kegagalan preconditions, code dapat menggunakan overload generik dari Requirement sebagai berikut.

```
Contract.Requires<ArgumentNullException>(x != null, "x");
```

6.1.2 Postconditions

Postconditions adalah status suatu method saat berakhir pada saat melakukan contract. Postconditions diperiksa sebelum keluar dari method. Behaviour run-time kegagalan postconditions ditentukan oleh runtime analyzer. Tidak seperti preconditions, postconditions mungkin merujuk anggota dengan visibilitas yang lebih rendah. Klien mungkin tidak dapat memahami atau menggunakan beberapa informasi yang diungkapkan oleh postconditions menggunakan private state, tetapi ini tidak mempengaruhi kemampuan klien untuk menggunakan method dengan benar.

Cara standard untuk menggunakan postconditions adalah dengan memanggil Method `Ensures`. Postconditions menyatakan kondisi yang harus bernilai benar setelah penghentian normal metode.

```
Contract.Ensures(this.F > 0);
```

Terdapat cara lain untuk menggunakan postcontions yaitu exceptional postconditions dan special postcondition. Exceptional postconditions adalah postconditions yang seharusnya bernilai benar saat exception tertentu dilemparkan oleh suatu method. Kita dapat menentukan postconditions ini dengan menggunakan metode `Contract.EnsuresOnThrow`, seperti yang diperlihatkan dalam contoh berikut.

```
Contract.EnsuresOnThrow<T>(this.F > 0);
```

6.1.3 Invariants

Objek invariants adalah kondisi yang harus benar untuk setiap instance kelas setiap kali objek tersebut terlihat oleh klien. Mereka mengungkapkan kondisi di mana objek tersebut dianggap benar.

Metode invarian diidentifikasi dengan ditandai dengan atribut `ContractInvariantMethodAttribute`. Method invarian tidak boleh berisi kode kecuali untuk urutan panggilan ke metode Invarian, yang masing-masing menetapkan invarian individual, seperti yang ditunjukkan pada contoh berikut.

```
[ContractInvariantMethod]
protected void ObjectInvariant ()
{
    Contract.Invariant(this.y >= 0);
    Contract.Invariant(this.x > this.y);
    ...
}
```

6.2 Defensive Programming

Defensive programming adalah bentuk desain defensif yang dimaksudkan untuk memastikan kelanjutan fungsi perangkat lunak dalam keadaan yang tidak terduga. Praktik pemrograman defensif sering digunakan saat availability, safety, or security tinggi diperlukan.

Defensive programming adalah pendekatan untuk meningkatkan perangkat lunak dan kode sumber, dalam hal:

- Kualitas secara umum - mengurangi jumlah bug dan masalah perangkat lunak.
- Membuat source code dapat dipahami (source code harus dapat dibaca dan dipahami sehingga disetujui dalam audit kode).
- Membuat perangkat lunak berperilaku dengan cara yang dapat diprediksi meskipun ada input atau tindakan pengguna yang tidak terduga.

Pemrograman yang terlalu defensif, bagaimanapun, dapat melindungi dari kesalahan yang tidak akan pernah ditemui, sehingga menimbulkan biaya waktu pengoperasian dan pemeliharaan. Ada juga risiko bahwa code traps mencegah terlalu banyak exceptions, yang berpotensi mengakibatkan hasil yang tidak diketahui dan tidak benar.

6.2.1 Assertions

Assertion, atau Assert statement, menguji kondisi yang Anda tentukan sebagai argumen untuk pernyataan Assert. Jika kondisi bernilai true, tidak ada tindakan yang terjadi. Jika kondisi bernilai false, pernyataan gagal. Jika Anda menjalankan dengan debug build mode, program Anda memasuki mode break.

Gunakan metode `System.Diagnostics.Debug.Assert` secara bebas untuk menguji kondisi yang seharusnya benar jika kode Anda benar. Misalnya, Anda telah menulis fungsi pembagian integer. Menurut aturan matematika, pembagi tidak pernah bisa menjadi nol. Anda dapat mengujinya menggunakan assertion:

```
int IntegerDivide ( int dividend , int divisor )
{
    Debug.Assert ( divisor != 0 );
    return ( dividend / divisor );
}
```

Saat Anda menjalankan kode ini di bawah debugger, pernyataan pernyataan dievaluasi, tetapi dalam versi rilis, perbandingan tidak dibuat, jadi tidak ada biaya tambahan. Sebagai catatan Perhatikan bahwa call method `System.Diagnostics.Debug.Assert` akan hilang saat pada saat membuat Release version dari kode tersebut.

6.2.2 Exceptions Handling

Fitur exception handling bahasa C # membantu kita menangani situasi yang tidak terduga atau luar biasa yang terjadi saat program sedang berjalan Exception handling menggunakan keyword the try, catch, and finally untuk mencoba actions yang mungkin tidak berhasil, menangani kegagalan saat Anda memutuskan bahwa tindakan tersebut masuk akal, dan membersihkan sumber daya setelahnya. Exceptions dibuat dengan menggunakan kata kunci throw.

Dalam contoh ini, method untuk menguji pembagian dengan nol dan menangkap error yang terjadi. Tanpa exception handling, program ini akan berhenti dengan kesalahan `DivideByZeroException` yang tidak tertangani.

```

using System;

public class ExceptionTest
{
    static double SafeDivision(double x, double y)
    {
        if (y == 0)
            throw new DivideByZeroException();
        return x / y;
    }

    public static void Main()
    {
        // Input for test purposes. Change the values to see
        // exception handling behavior.
        double a = 98, b = 0;
        double result;

        try
        {
            result = SafeDivision(a, b);
            Console.WriteLine("{0} divided by {1} = {2}", a, b,
result);
        }
        catch (DivideByZeroException)
        {
            Console.WriteLine("Attempted divide by zero.");
        }
    }
}

```

MODUL 7 GRAMMAR-BASED INPUT PROCESSING (PARSING)

TUJUAN PRAKTIKUM

1. Menguasai penerapan grammar-based input processing pada C#.Net
2. Menguasai penggunaan format JSON pada C#.Net

7.1 Parsing

Parsing adalah proses menganalisis serangkaian simbol, baik dalam bahasa alami, bahasa komputer, atau struktur data, sesuai dengan aturan tata bahasa formal. Istilah parsing berasal dari bahasa Latin pars (orationis), yang berarti bagian (dari ucapan).

Dalam linguistik komputasi, parsing digunakan untuk merujuk pada analisis formal oleh komputer dari sebuah kalimat atau rangkaian kata lain ke dalam konstituennya, menghasilkan pohon parse yang menunjukkan hubungan sintaksisnya satu sama lain, yang mungkin juga berisi informasi semantik dan lainnya (p-values). Beberapa algoritma parsing mungkin menghasilkan hutan parse atau daftar pohon parse untuk input yang ambigu secara sintaksis.

Dalam ilmu komputer, istilah tersebut digunakan dalam analisis bahasa komputer, mengacu pada analisis sintaksis dari kode masukan menjadi bagian-bagian komponennya untuk memudahkan penulisan penyusun dan juru bahasa. Istilah ini juga dapat digunakan untuk menggambarkan perpecahan atau pemisahan.

Secara sederhana Parsing adalah proses mengubah teks yang diformat menjadi struktur data. Jenis struktur data dapat berupa representasi yang sesuai dari informasi yang diukir dalam source text. Sebuah program yang melakukan parsing disebut Parser. Terdapat beberapa jenis parsing sebagai berikut:

- Tree Parsing adalah pilihan umum dan standar untuk penguraian XML, penguraian HTML, penguraian JSON, dan penguraian bahasa pemrograman apa pun. Pohon keluaran disebut Parse Tree atau Abstract Syntax Tree. Dalam konteks HTML, ini disebut Model Objek Dokumen (DOM).
- Parsing file CSV dapat menghasilkan Daftar nilai atau objek Daftar Rekaman.
- Graph Parsing adalah pilihan untuk penguraian bahasa alami.

7.2 Parsing String

Parsing mengonversi string yang mewakili tipe dasar .NET menjadi tipe dasar tersebut. Misalnya, Parsing digunakan untuk mengonversi string ke angka floating-point atau ke nilai tanggal dan waktu. Metode yang paling umum digunakan untuk melakukan parsing adalah metode Parse. Hal itu karena parsing adalah operasi kebalikan dari pemformatan (yang melibatkan pengubahan tipe dasar menjadi representasi stringnya), banyak aturan dan konvensi yang sama berlaku. Sama seperti pemformatan menggunakan objek yang mengimplementasikan antarmuka IFormatProvider untuk menyediakan informasi pemformatan yang jelas, parsing juga menggunakan objek yang mengimplementasikan antarmuka IFormatProvider untuk menentukan cara menafsirkan representasi string. Berikut ini contoh source code parsing numeric dari string:

```
using System;
using System.Globalization;
public class Example
{
    public static void Main()
    {
        string[] values = { "1,304.16", "$1,456.78", "1,094", "152",
                           "123,45 €", "1 304,16", "Ae9f" };
        double number;
        CultureInfo culture = null;
        foreach (string value in values) {
            try {
                culture = CultureInfo.CreateSpecificCulture("en-US");
                number = Double.Parse(value, culture);
                Console.WriteLine("{0}: {1} --> {2}", culture.Name, value,
number);
            }
            catch (FormatException) {
                Console.WriteLine("{0}: Unable to parse '{1}'.",
                                culture.Name, value);
                culture = CultureInfo.CreateSpecificCulture("fr-FR");
                try {
                    number = Double.Parse(value, culture);
                    Console.WriteLine("{0}: {1} --> {2}", culture.Name, value,
number);
                }
                catch (FormatException) {
```

```

        Console.WriteLine("{0}: Unable to parse '{1}'.",
                           culture.Name, value);
    }
}
Console.WriteLine();
}
}
}

```

Selain digunakan untuk melakukan parsing dari string menuju numeric, Parsing juga dapat digunakan untuk melakukan parsing dari string menuju date time, char, boolean, dan enum seperti pada source dibawah ini.

```

string dateInput = "Jan 1, 2009";
var parsedDate = DateTime.Parse(dateInput);
Console.WriteLine(parsedDate);
// Displays the following output on a system whose culture is en-US:
//      1/1/2009 00:00:00

string MyString1 = "A";
char MyChar = Char.Parse(MyString1);
// MyChar now contains a Unicode "A" character.

string MyString2 = "True";
bool MyBool = bool.Parse(MyString2);
// MyBool now contains a True Boolean value.

string MyString3 = "Thursday";
DayOfWeek MyDays = (DayOfWeek)Enum.Parse(typeof(DayOfWeek), MyString3);
Console.WriteLine(MyDays);
// The result is Thursday.

```

7.3 Parsing JSON

JSON singkatan dari *JavaScript Object Notation*, adalah suatu format ringkas pertukaran data komputer. Formatnya berbasis teks dan terbaca-manusia serta digunakan untuk merepresentasikan struktur data sederhana dan larik asosiatif (disebut objek). Format JSON sering digunakan untuk mentransmisikan data terstruktur melalui suatu koneksi jaringan pada suatu proses yang disebut serialisasi. Aplikasi utamanya adalah pada pemrograman aplikasi web AJAX dengan

berperan sebagai alternatif terhadap penggunaan tradisional format XML. Parsing diperlukan untuk mengubah objek yang tersimpan dalam JSON agar dapat dibaca oleh program computer. Berikut ini adalah contoh source code C# untuk melakukan parsing JSON.

```
using System;
using System.Text.Json;
using System.Text.Json.Serialization;

namespace Fields
{
    public class Forecast
    {
        public DateTime Date;
        public int TemperatureC;
        public string Summary;
    }

    public class Forecast2
    {
        [JsonInclude]
        public DateTime Date;
        [JsonInclude]
        public int TemperatureC;
        [JsonInclude]
        public string Summary;
    }

    public class Program
    {
        public static void Main()
        {
            var json =
                @"{"Date":"2020-09-06T11:31:01.923395-07:00","TemperatureC":-1,"Summary":"Cold"} ";
            Console.WriteLine($"Input JSON: {json}");

            var options = new JsonSerializerOptions
            {
```

```

        IncludeFields = true,
    };
    var forecast = JsonSerializer.Deserialize<Forecast>(json,
options);

    Console.WriteLine($"forecast.Date: {forecast.Date}");
    Console.WriteLine($"forecast.TemperatureC:
{forecast.TemperatureC}");
    Console.WriteLine($"forecast.Summary: {forecast.Summary}");

    var roundTrippedJson =
        JsonSerializer.Serialize<Forecast>(forecast, options);

    Console.WriteLine($"Output JSON: {roundTrippedJson}");

    options = new JsonSerializerOptions(JsonSerializerDefaults.Web);
    var forecast2 = JsonSerializer.Deserialize<Forecast2>(json);

    Console.WriteLine($"forecast2.Date: {forecast2.Date}");
    Console.WriteLine($"forecast2.TemperatureC:
{forecast2.TemperatureC}");
    Console.WriteLine($"forecast2.Summary: {forecast2.Summary}");

    roundTrippedJson =
        JsonSerializer.Serialize<Forecast2>(forecast2, options);

    Console.WriteLine($"Output JSON: {roundTrippedJson}");
}
}
}

// Produces output like the following example:
//
//Input JSON: { "date":"2020-09-06T11:31:01.923395-07:00","temperatureC":-
1,"summary":"Cold"}
//
//forecast.Date: 9/6/2020 11:31:01 AM
//forecast.TemperatureC: -1

```

```
//forecast.Summary: Cold
//Output JSON: { "date":"2020-09-06T11:31:01.923395-07:00","temperatureC":-1,"summary":"Cold"}

//forecast2.Date: 9/6/2020 11:31:01 AM
//forecast2.TemperatureC: -1
//forecast2.Summary: Cold
//Output JSON: { "date":"2020-09-06T11:31:01.923395-07:00","temperatureC":-1,"summary":"Cold"}
```



MODUL 8 RUNTIME CONFIGURATION DAN INTERNATIONALIZATION

TUJUAN PRAKTIKUM

1. Menguasai penggunaan runtime configuration pada C#.Net
2. Menguasai penerapan internationalization pada C#.Net

8.1 Runtime Configuration

Runtime configuration adalah sebuah teknik binds nilai variabel dan program settings ketika program berjalan, bisanya dengan mengupdate dan membaca configuration file pada saat just-in-time mode. NET Core mendukung penggunaan configuration files dan environment variables untuk mengkonfigurasi aplikasi pada saat run time. Runtime configuration akan digunakan ketika:

1. Tidak memiliki atau mengontrol source code untuk aplikasi dan oleh karena itu tidak dapat mengkonfigurasinya secara terprogram.
2. Beberapa aplikasi yang dibuat berjalan pada waktu yang sama pada satu sistem, dan perlu dilakukan konfigurasi untuk kinerja yang optimal.

Beberapa nilai konfigurasi pada .NET Core dapat di set secara terprogram dengan memanggil method `AppContext.SetSwitch`. .NET Core menyediakan mekanisme 3 mekanisme untuk melakukan runtime configuration yaitu:

1. The `runtimeconfig.json` file
2. MSBuild properties
3. Environment variables

8.1.1 Runtimeconfig.json

Pada saat project dibuild, file `[appname].runtimeconfig.json` akan tergenerate pada direktori output. Jika file `runtimeconfig.template.json` ada di folder yang sama dengan file proyek, semua konfigurasi yang ada di dalamnya akan dimasukkan ke dalam file `[appname].runtimeconfig.json`.

Set runtime-configuration pada bagian `configProperties` dari file `runtimeconfig.json`. `configProperties` berbentuk seperti ini:

```
"configProperties": {  
  "config-property-name1": "config-value1",  
  "config-property-name2": "config-value2"  
}
```

Berikut ini adalah contoh file [appname].runtimeconfig.json:

```
{
  "runtimeOptions": {
    "tfm": "netcoreapp3.1",
    "framework": {
      "name": "Microsoft.NETCore.App",
      "version": "3.1.0"
    },
    "configProperties": {
      "System.GC.Concurrent": false,
      "System.Threading.ThreadPool.MinThreads": 4,
      "System.Threading.ThreadPool.MaxThreads": 25
    }
  }
}
```

8.1.2 MSBuild Properties

Beberapa runtime run-time configuration options dapat diset menggunakan MSBuild properties yang ada pada file .csproj or .vbproj file. MSBuild properties memiliki prioritas yang lebih tinggi dibandingkan runtimeconfig.template.json file. Hal itu menyebabkan apabila ada konfigurasi yang ada pada di MSBuild dan runtimeconfig.template.json juga, nilai konfigurasi yang digunakan yang berada pada MSBuild.

Berikut adalah contoh dari MSBuild properties for configuring run-time:

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>netcoreapp3.1</TargetFramework>
  </PropertyGroup>
  <PropertyGroup>
    <ConcurrentGarbageCollection>>false</ConcurrentGarbageCollection>
    <ThreadPoolMinThreads>4</ThreadPoolMinThreads>
    <ThreadPoolMaxThreads>25</ThreadPoolMaxThreads>
  </PropertyGroup>
</Project>
```

8.1.3 Environment variables

Environment variables dapat digunakan juga untuk melakukan run-time configuration. Pengaturan a run-time option menggunakan environment variable dapat diterapkan pada semua .NET Core apps. Konfigurasi environment variables secara umum memiliki prefix **COMPlus_**.

Berikut contoh mengatur environment variable melalui command line:

```
# Windows
set COMPlus_GCRetainVM=1
# Powershell
$env:COMPlus_GCRetainVM="1"
# Unix
export COMPlus_GCRetainVM=1
```

8.2 Internationalization

Internationalization adalah proses merancang aplikasi perangkat lunak sehingga dapat disesuaikan dengan berbagai bahasa dan wilayah tanpa perubahan teknik. Localization adalah proses mengadaptasi perangkat lunak internasional untuk wilayah atau bahasa tertentu dengan menerjemahkan teks dan menambahkan komponen khusus lokal. Lokalisasi (yang berpotensi dilakukan beberapa kali, untuk lokal yang berbeda) menggunakan infrastruktur atau fleksibilitas yang disediakan oleh internasionalisasi (yang idealnya dilakukan hanya sekali sebelum lokalisasi, atau sebagai bagian integral dari pembangunan yang sedang berlangsung).

Langkah pertama dari melakukan internationalization adalah membagi setiap bagian yang berpotensi locale-dependent (baik kode, teks, atau data) menjadi modul terpisah. Setiap modul kemudian dapat mengandalkan standard library atau diganti secara independen sesuai kebutuhan untuk setiap lokal.

Praktik yang berlaku saat ini adalah aplikasi untuk menempatkan teks dalam resource string yang dimuat selama eksekusi program sesuai kebutuhan. String ini, disimpan dalam resource files dan relatif mudah diterjemahkan. Katalog adalah tempat penyimpanan untuk translatable and translated strings pesan. Katalog biasanya terdiri dari sekumpulan file dalam format lokalisasi tertentu dan standard library untuk menangani format tersebut. Salah satu software library adalah gettext.

Jadi untuk mendapatkan aplikasi yang mendukung banyak bahasa, seseorang akan mendesain aplikasi untuk memilih relevant language resource file pada runtime. Kode yang diperlukan untuk mengelola verifikasi entri data dan banyak tipe data sensitif lokal lainnya juga harus mendukung persyaratan lokal yang berbeda. Sistem pengembangan dan sistem operasi modern menyertakan

sophisticated libraries untuk dukungan internasional jenis ini. Banyak masalah localization (mis. Arah penulisan, pengurutan teks) memerlukan perubahan yang lebih besar dalam perangkat lunak dari pada terjemahan teks.

8.2.1 Portable Object Localization

Portable Object (PO) adalah Sebuah tools pada ASP.NET Core yang berguna untuk melakukan Internationalization selain menggunakan .resx file. Berikut ini adalah keuntungan menggunakan PO:

1. Mendukung pluralization
2. Tidak terlalu complex dibandingkan .resx file
3. Dapat digunakan secara collaborative online editing tools.

Berikut ini adalah contoh file dari po untuk bahasa prancis (fr.po):

```
#: Services/EmailService.cs:29
msgid "Enter a comma separated list of email addresses."
msgstr "Entrez une liste d'emails séparés par une virgule."

#: Views/Email.cshtml:112
msgid "The email address is \"{0}\"."
msgid_plural "The email addresses are \"{0}\"."
msgstr[0] "L'adresse email est \"{0}\"."
msgstr[1] "Les adresses email sont \"{0}\""
```

Pada contoh berikut code yang digunakan dapat di download [disini](#). Berikut ini langkah langkah menggunakan Portable Object:

1. Referencing the package

Tambahkan referensi ke paket NuGet OrchardCore.Localization.Core.

File .csproj sekarang berisi baris yang mirip dengan berikut ini (nomor versi mungkin berbeda):

```
<PackageReference Include="OrchardCore.Localization.Core"
Version="1.0.0-rc2-13450" />
```

2. Registering the service

Tambahkan services yang diperlukan ke metode ConfigureServices dari Startup.cs:

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc()

    .AddViewLocalization(LanguageViewLocationExpanderFormat.Suffix);
    services.AddPortableObjectLocalization();
    services.Configure<RequestLocalizationOptions>(options =>
    {
        var supportedCultures = new List<CultureInfo>
        {
            new CultureInfo("en-US"),
            new CultureInfo("en"),
            new CultureInfo("fr-FR"),
            new CultureInfo("fr")
        };
        options.DefaultRequestCulture = new RequestCulture("en-US");
        options.SupportedCultures = supportedCultures;
        options.SupportedUICultures = supportedCultures;
    });
}
```

Tambahkan middleware Configure method pada *Startup.cs*:

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler("/Home/Error");
    }
    app.UseRouting();
    app.UseStaticFiles();
    app.UseRequestLocalization();
}
```



```
app.UseEndpoints(endpoints =>
{
    endpoints.MapControllerRoute(name: "default", pattern:
"{controller=Home}/{action=Index}/{id?}");
});
}
```

Tambahkan kode berikut ke tampilan Razor pilihan Anda. About.cshtml digunakan dalam contoh ini.

```
@using Microsoft.AspNetCore.Mvc.Localization
@inject IViewLocalizer Localizer
<p>@Localizer["Hello world!"]</p>
```

3. Creating a PO file

Buat file bernama <culture code> .po di folder root aplikasi Anda. Dalam contoh ini, nama file adalah fr.po karena bahasa Prancis digunakan:

```
msgid "Hello world!"
msgstr "Bonjour le monde!"
```

4. Testing the application

Jalankan aplikasi Anda, dan arahkan ke URL / Home / About. Teks Halo dunia! ditampilkan.

Arahkan ke URL / Home / About? Culture = fr-FR. Teks Bonjour le monde! ditampilkan.

MODUL 9 API DESIGN DAN CONSTRUCTION USING SWAGGER

TUJUAN PRAKTIKUM
1. Menguasai penggunaan Swagger untuk API Design
2. Menguasai penggunaan Swagger untuk API Construction

9.1 API

Application Programming Interface (API) adalah sebuah interface komputasi yang mendefinisikan interaksi antara beberapa perantara software. API mendefinisikan jenis call atau request yang dapat dibuat, bagaimana membuatnya, format data yang harus digunakan, konvensi yang harus diikuti, dll. API juga dapat menyediakan mekanisme ekstensi sehingga pengguna dapat memperluas fungsionalitas yang ada dengan berbagai cara. API dapat sepenuhnya dibuat khusus, khusus untuk suatu komponen, atau dirancang berdasarkan standar industri untuk memastikan interoperability. API menerapkan pemrograman modular, memungkinkan pengguna untuk menggunakan interface secara independen. Tujuan adanya API adalah untuk menyederhanakan pemrograman dengan mengabstraksi implementasi yang mendasarinya dan hanya mengekspos objek atau tindakan yang dibutuhkan pengembang. Dengan adanya hal itu pengembang tidak perlu memusingkan operasi yang terjadi pada API dan dapat lebih fokus menggunakan API dalam pembuatan software.

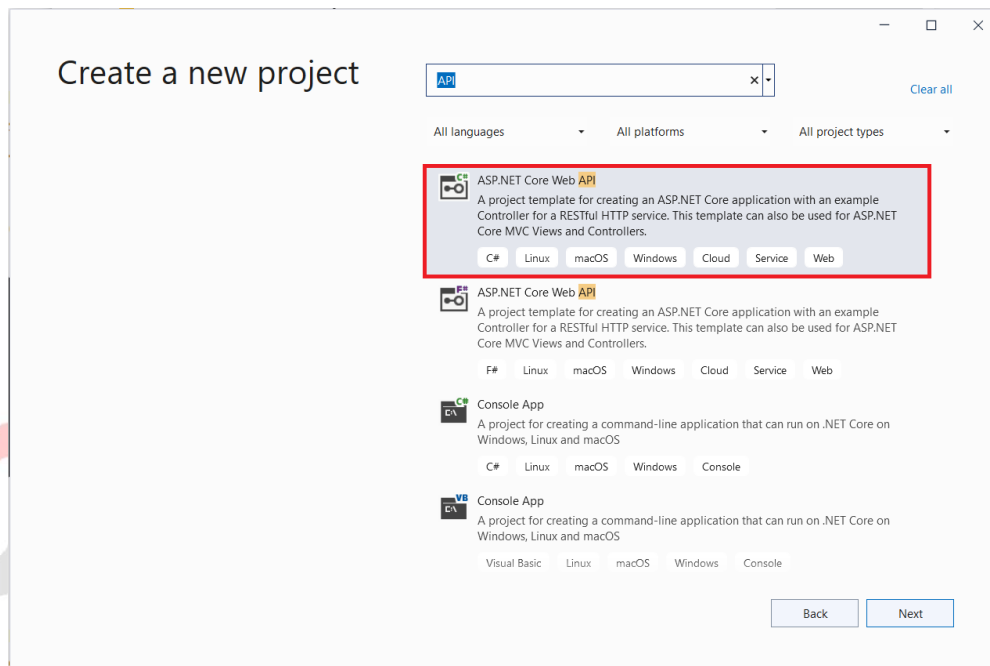
API design adalah proses pengembangan API yang mengekspos data dan fungsionalitas aplikasi untuk digunakan oleh pengembang dan pengguna. Desain API memiliki pengaruh signifikan pada penggunaannya. Prinsip information hiding menjelaskan peran antarmuka pemrograman yang memungkinkan pemrograman modular dengan menyembunyikan detail implementasi modul sehingga pengguna modul tidak perlu memahami kompleksitas di dalam modul. Dengan demikian, desain API mencoba menyediakan hanya alat yang diharapkan pengguna. Desain antarmuka pemrograman merupakan bagian penting dari arsitektur perangkat lunak, organisasi perangkat lunak yang kompleks. Berdasarkan hal diatas, Berikut ini beberapa hal yang harus terjawab sebelum membuat API:

1. Mengapa kami ingin menerapkan API ?
2. Hal apa yang ingin dicapai dengan API ?
3. Bagaimana plan untuk membuat API ?

9.2 Membuat Sample Project

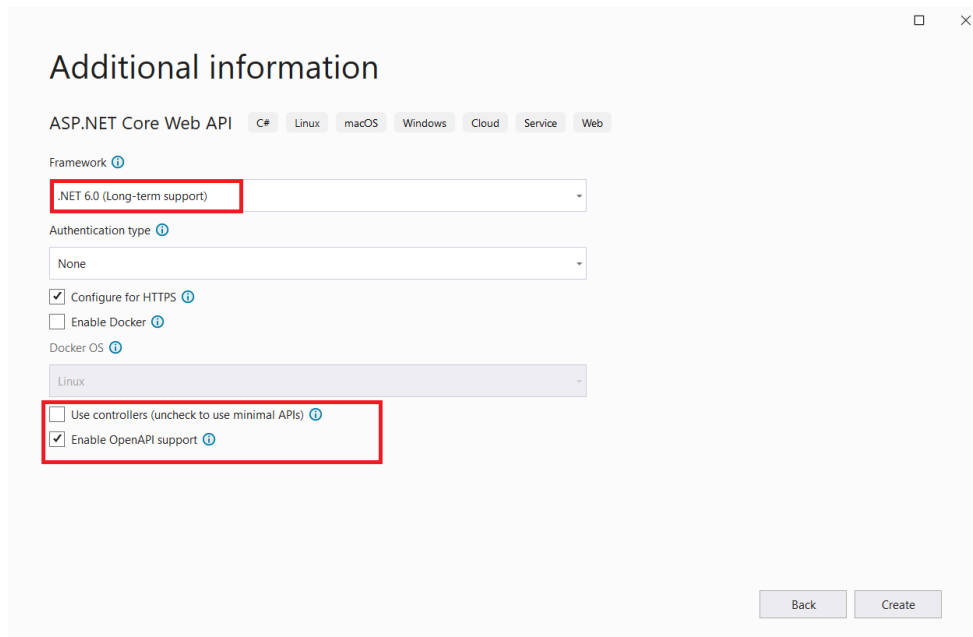
Visual Studio menyediakan project template yang dapat digunakan untuk mempelajari bagaimana kita dapat membuat web API. Berikut ini adalah tahapan pembuatan project baru yang menggunakan template tersebut:

1. Menjalankan IDE Visual Studio dan memilih “Create a new project”.
2. Pada halaman berikutnya, cari API di search bar dan pilih “ASP.NET Core Web API template” kemudian tekan tombol Next.



Gambar 9. 1 Membuat project baru pada Visual Studio

3. Beri nama project dan tekan tombol Next.
4. Pastikan .NET 6.0 (Long-term support) sudah dipilih (atau minimal .NET 5.0) dan opsi “Enable OpenAPI support” juga sudah dicentang.



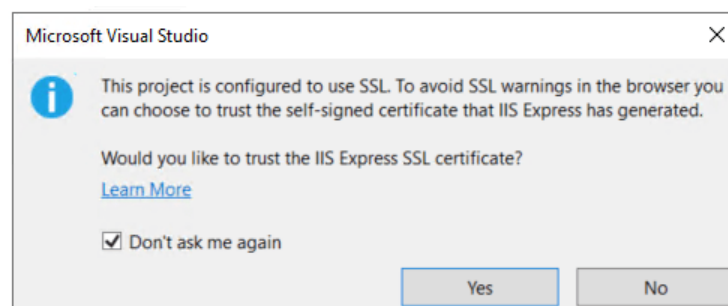
Gambar 9. 2 Memilih versi .NET pada project

5. Pilihan “Configure for HTTPS” juga boleh untuk tidak dicentang apabila terjadi masalah pada saat project tersebut mengalami masalah pada saat dijalankan.

9.3 Menjalankan Sample Project

Untuk menjalankan project yang telah dibuat sebelumnya, pada Visual Studio ditekan tombol “Run” atau dengan menggunakan shortcut “Ctrl+F5” untuk menjalankan tanpa debugger. Apabila setting sebelumnya diset untuk HTTPS, maka akan perlu dikonfigurasi beberapa hal sebagai berikut:

1. Mengizinkan dengan memilih tombol Yes pada pop up yang menanyakan IIS Express SSL certificate.



Gambar 9. 3 Popup SSL ISS Express

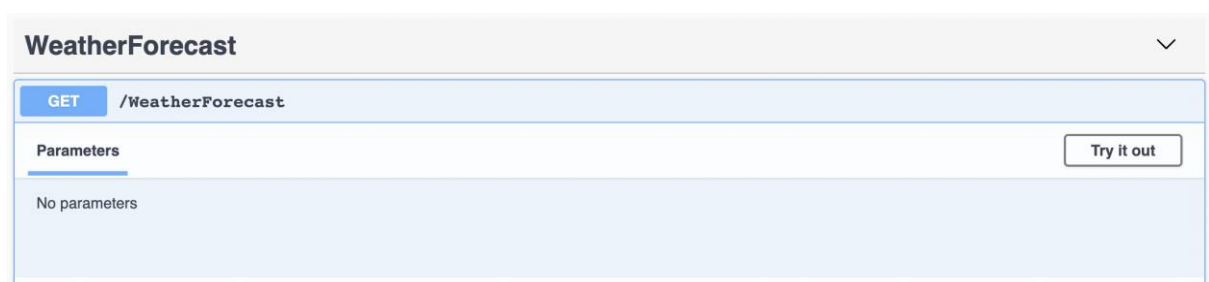
2. Setelah itu tekan tombol Yes kembali untuk melakukan instalasi SSL certificate yang dibutuhkan untuk menjalankan project dengan SSL tersebut.

Setelah program berjalan tanpa error, API dari project sample yang sudah dibuat akan dapat diakses melalui halaman (menggunakan browser): https://localhost:<port_number>/swagger/index.html



Gambar 9. 4 Test API dengan Swagger

Harap dicoba untuk mengamati website tersebut dengan memilih atau menekan tombol yang ada, misalnya menekan tombol “Try it out” setelah memilih tombol “GET /WeatherForecase”, kemudian menekan tombol “Execute”.



Gambar 9. 5 Hasil Test API dengan Swagger

Hasil output dari hasil eksekusi tersebut merupakan hasil dari source code yang berada di file “NamaSolution/Controllers/WeatherForecastController.cs”

MODUL 10 LIBRARY CONSTRUCTION

TUJUAN PRAKTIKUM

1. Menguasai pembangunan library pada C#.Net.

10.1 Library



Gambar 10. 1 Ilustrasi aplikasi yang menggunakan libvorbisfile untuk memutar file Ogg Vorbis.

Library adalah kumpulan sumber daya non-volatil yang digunakan oleh program komputer, seringkali untuk pengembangan perangkat lunak termasuk configuration data, documentation, help data, message templates, pre-written code and subroutines, classes, values or type specifications. Library juga merupakan kumpulan implementasi perilaku, ditulis dalam istilah bahasa, yang memiliki antarmuka yang terdefinisi dengan baik di mana perilaku tersebut dipanggil. Misalnya, orang yang ingin menulis program tingkat yang lebih tinggi dapat menggunakan library untuk melakukan panggilan sistem daripada mengimplementasikan panggilan sistem itu berulang kali. Selain itu, perilaku disediakan untuk digunakan kembali oleh beberapa program independen dengan melakukan call atau invoke fungsi.

Library code diatur sedemikian rupa sehingga dapat digunakan oleh banyak program yang tidak memiliki koneksi satu sama lain, sedangkan kode yang merupakan bagian dari program diatur untuk digunakan hanya dalam satu program itu. Perbedaan ini dapat memperoleh pengertian hierarkis ketika sebuah program tumbuh besar, seperti program berjuta-juta baris. Dalam kasus tersebut, mungkin ada internal libraries yang digunakan kembali oleh sub-bagian independen dari program besar. Fitur yang membedakan adalah library diatur untuk tujuan digunakan kembali oleh program atau sub-program independen, dan pengguna hanya perlu mengetahui antarmuka dan bukan rincian internal library.

Berikut ini beberapa keuntungan menggunakan library dalam konstruksi perangkat lunak:

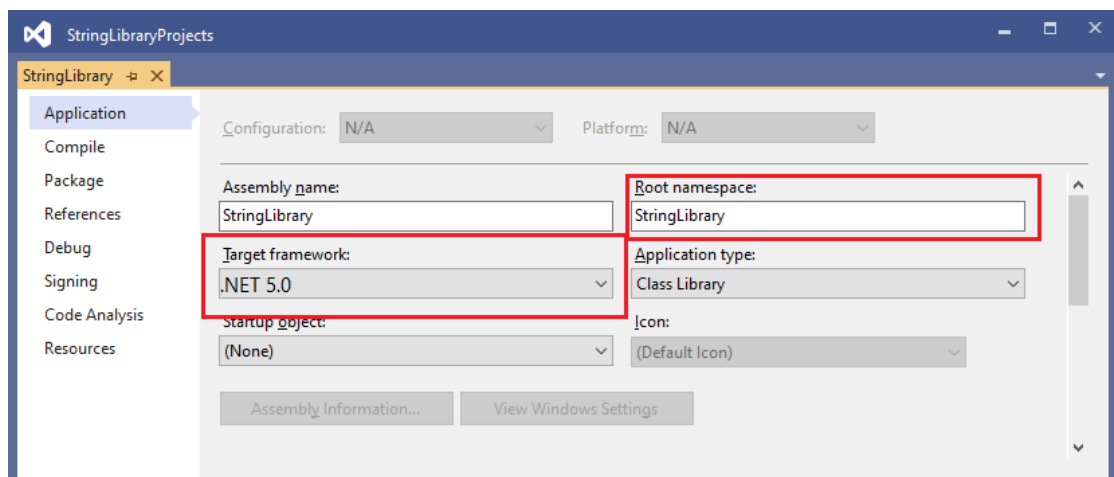
1. Code yang dibuat menjadi lebih modular,
2. Waktu pengerjaan menjadi lebih cepat.

3. Meningkatkan Reusable dari program.
4. Memudahkan distribusi kode.
5. Mengurangi jumlah baris code yang ada pada program.

Berdasarkan hal hal diatas, library sangat berguna dalam konsturksi program. Pada beberapa kasus, Terdapat fungsi fungsi yang dibutuhkan tidak terdapat pada library standard yang tersedia. Hal itu mengharuskan, kita untuk membuat library sendiri yang berguna bagi project yang sedang dikerjakan.

10.1.1 Membuat Custom Library

1. Buatlah sebuah project kosong (blank solution) pada visual studio dan beri nama prohect itu dengan **“ClassLibraryProjects”**.
2. Buka project tersebut
3. Tambahkan .NET class library baru bernama **“StringLibrary”** pada solution tersebut.
 - a. Klik kanan pada Solution Explorer dan pilih Add > New Project.
 - b. Pada halaman Add a new project, masukkan library di search box. Pilih C # dari Language list, lalu pilih Semua platform dari Platform list. Pilih Class Library template, lalu pilih Next.
 - c. Pada halaman Configure your new project, masukkan StringLibrary sebagai nama project, lalu pilih Next.
 - d. Di halaman Informasi tambahan, pilih .NET 5.0 (Saat Ini), lalu pilih create.
4. Pastikan bahwa targets library benar. Klik kanan pada library project di Solution Explorer, dan kemudian pilih Properti. Ikuti pengaturan berikut



Gambar 10. 2 StringLibraryProjects window.

5. Replace the code pada Class1.cs dengan kode berikut kemudian save.

```
using System;
namespace UtilityLibraries
{
    public static class StringLibrary
    {
        public static bool StartsWithUpper(this string str)
        {
            if (string.IsNullOrEmpty(str))
                return false;
            char ch = str[0];
            return char.IsUpper(ch);
        }
    }
}
```

6. Pada menu bar, select **Build > Build Solution** or tekan Ctrl+Shift+B untuk memastikan project terkompile tanpa error.

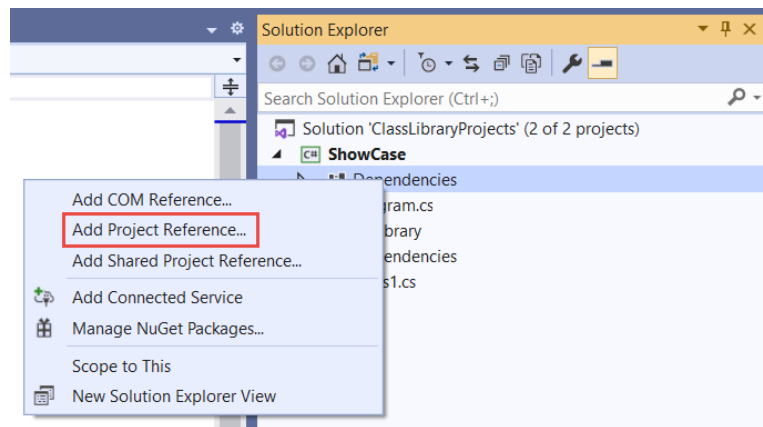
10.1.2 Menambahkan Custom Library pada Solution

1. Tambahkan .NET console application baru bernama "ShowCase" ke solusi.
 - a. Klik kanan pada solusi di Solution Explorer dan pilih Add> New project.
 - b. Di Add a new project page, Tuliskan console pada search box. Pilih C # dari Language list, lalu pilih Semua platform dari daftar Platform.
 - c. Pilih Console Application template, Kemudian klik next.
 - d. Masukkan ShowCase pada project name box. Kemudian klik next.

2. Replace code pada Program.cs menjadi berikut, kemudian save.

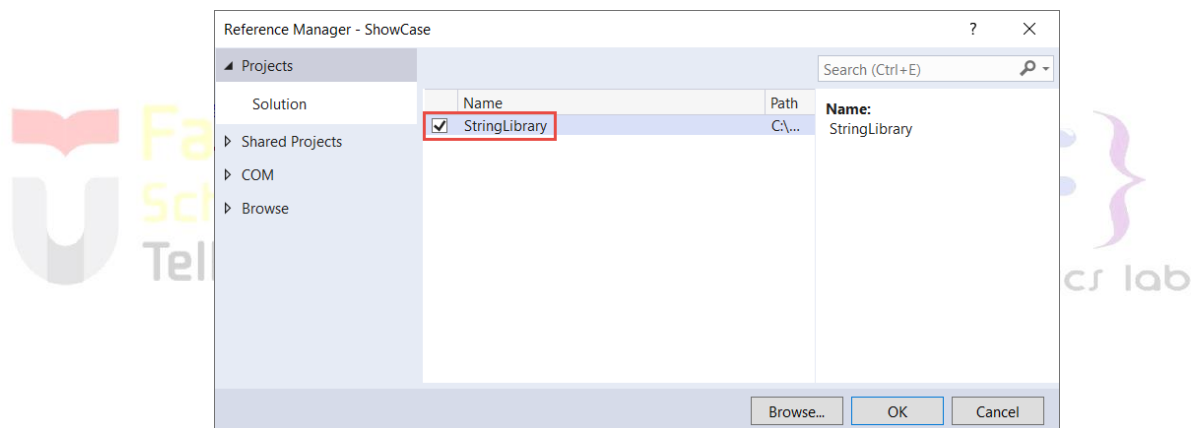
```
using System;
using UtilityLibraries;
class Program
{
    static void Main(string[] args)
    {
        int row = 0;
        do
        {
            if (row == 0 || row >= 25)
                ResetConsole();
            string input = Console.ReadLine();
            if (string.IsNullOrEmpty(input)) break;
            Console.WriteLine($"Input: {input} {"Begins with
uppercase? ",30}: " +
                                $"{(input.StartsWithUpper() ? "Yes" :
"No")}\n");
            row += 3;
        } while (true);
        return;
        // Declare a ResetConsole local method
        void ResetConsole()
        {
            if (row > 0)
            {
                Console.WriteLine("Press any key to continue...");
                Console.ReadKey();
            }
            Console.Clear();
            Console.WriteLine("\nPress <Enter> only to exit;
otherwise, enter a string and press <Enter>:\n");
            row = 3;
        }
    }
}
```

3. Pada **Solution Explorer**, klik kanan pada ShowCase project's Dependencies node, dan select Add Project Reference.



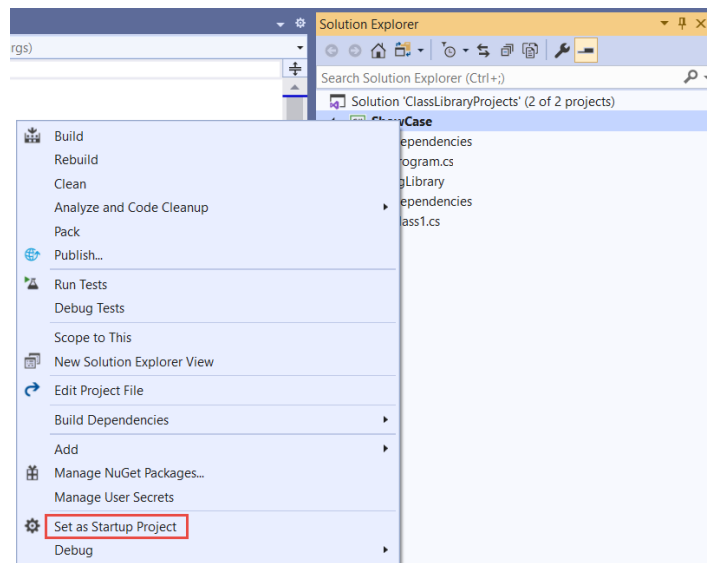
Gambar 10. 3 Add Project Reference.

4. Pada **Reference Manager** dialog, select the **StringLibrary** project, dan select **OK**.



Gambar 10. 4 Reference Manager.

5. Pada **Solution Explorer**, klik kanan pada **ShowCase** project dan select **Set as StartUp Project** pada context menu.



Gambar 10. 5 Solution Explorer.

6. Tekan Ctrl+F5 untuk mengkompile dan menjalankan program.



Gambar 10. 6 Tombol Debug.

MODUL 11 Penilaian Tugas Besar CLO2

TUJUAN PRAKTIKUM
1. Mampu mempertanggungjawabkan pemilihan tools konstruksi.
2. Mampu mempertanggungjawabkan pemilihan teknik dan teknologi konstruksi.

11.1 Tugas besar

Pada presentasi tugas besar, anda diharuskan untuk menunjukkan kemampuan dalam menerapkan dan menjustifikasi penggunaan tools, teknik, teknologi, dan pendekatan terhadap konstruksi yang dilakukan. Presentasi akan dilakukan secara berkelompok dimana anda akan diminta menjelaskan apa saja yang sudah anda lakukan pada tugas besar mata kuliah Konstruksi Perangkat Lunak. Presentasi kemudian dilanjutkan dengan demo program, unit test, dan performance analysis.

Sesi terakhir presentasi adalah tanya jawab dimana secara individu anda akan diminta mempertanggungjawabkan keputusan-keputusan yang diambil selama menyelesaikan tugas besar mata kuliah Konstruksi Perangkat Lunak. Pada tanya jawab ini akan dinilai kemampuan anda untuk merasionalisasikan penggunaan teknologi-teknologi konstruksi yang ada berdasarkan kelebihan dan kekurangan dari teknologi tersebut.

11.2 Hal yang harus dipersiapkan

Saat presentasi pastikan anda sudah menyiapkan hal-hal berikut agar presentasi berjalan lancar:

1. Laporan dan tayangan presentasi yang menjelaskan apa saja yang sudah diterapkan pada tugas besar.
2. Source code dari perangkat lunak yang dibangun, tersimpan dalam repository.
3. Executable code dari perangkat lunak yang dibangun dan dapat didemonstrasikan pada saat presentasi.
4. Unit test code yang dapat didemonstrasikan pada saat presentasi.

Performance tools yang siap untuk dijalankan untuk menunjukan performa hasil konstruksi.

MODUL 12 PERFORMANCE ANALYSIS, UNIT TESTING, DAN DEBUGGING

TUJUAN PRAKTIKUM

1. Menguasai penerapan unit testing dan debugging pada C#.Net.
2. Menguasai penggunaan tools performance analysis pada visual studio.

12.1 Profiling

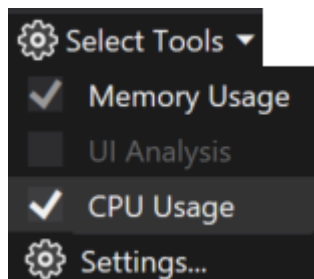
Profiling ("program profiling", "software profiling") adalah suatu bentuk analisis program dinamis yang mengukur, misalnya, kompleksitas ruang (memori) atau waktu suatu program, penggunaan instruksi tertentu, atau frekuensi dan durasi fungsi. Paling umum, informasi profil berfungsi untuk membantu pengoptimalan program, dan lebih khusus lagi, rekayasa kinerja.

Pembuatan profil dilakukan dengan source code program atau bentuk biner yang dapat dieksekusi menggunakan alat yang disebut profiler (atau code profiler). Profiler dapat menggunakan sejumlah teknik berbeda, seperti metode event-based, statistik, berinstrumen, dan simulasi.

12.1.1 Profiling CPU usage

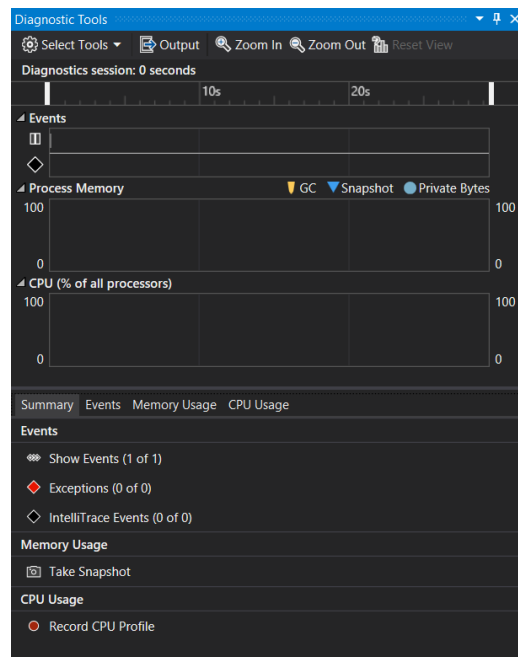
Berikut ini langkah langkah untuk mendapatkan profiling data CPU.

1. Buka project yang ingin didebug pada Visual Studio dan set breakpoint di aplikasi pada titik tempat kita ingin memeriksa penggunaan CPU.
2. Tetapkan breakpoint kedua di akhir fungsi atau wilayah kode yang ingin Anda analisis. Dengan mengatur dua breakpoint, Anda bisa membatasi pengumpulan data ke bagian kode yang ingin Anda analisis.
3. Diagnostic Tools window akan muncul secara otomatis kecuali Anda telah mematikannya. Untuk membuka kembali window tersebut, klik Debug> Windows> Show Diagnostic Tools.
4. Anda dapat memilih untuk melihat Penggunaan CPU, Penggunaan Memori, atau keduanya, dengan pengaturan Select Tools pada toolbar.



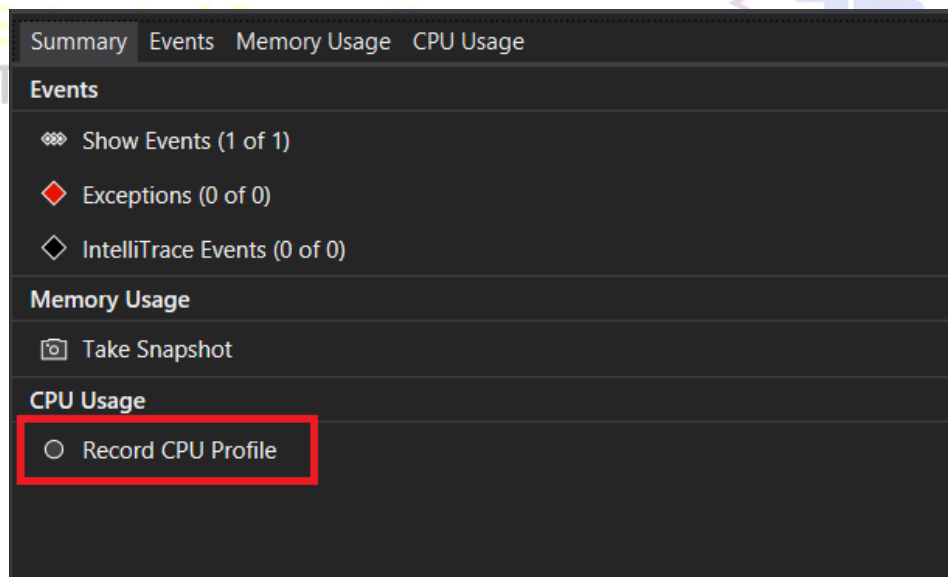
Gambar 12. 1 Usage Tools.

5. Klik **Debug > Start Debugging** (atau **Start** pada toolbar, atau **F5**).



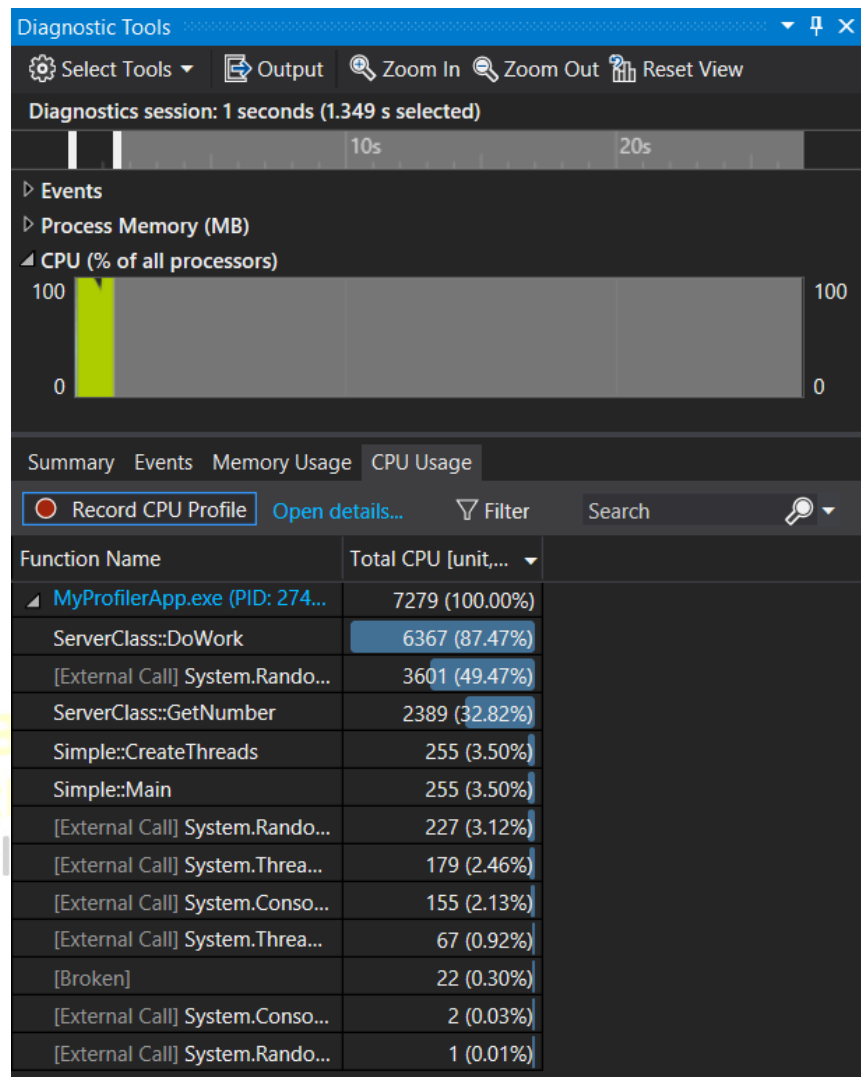
Gambar 12. 2 Diagnostic Tools.

6. Jalankan skenario yang akan menyebabkan breakpoint pertama terkena.
7. Saat debugger terpaused, aktifkan collection of the CPU Usage data, lalu buka tab CPU Usage.



Gambar 12. 3 Record CPU Profile.

8. Tekan F5 untuk menjalankan aplikasi ke breakpoint kedua dan tunggu hingga program terhenti di breakpoint kedua.



Gambar 12. 4 Hasil Record CPU Profile.

9. Jika ingin memilih region kode yang lebih spesifik untuk dianalisis, pilih region di timeline CPU (harus region yang menampilkan data profiling).

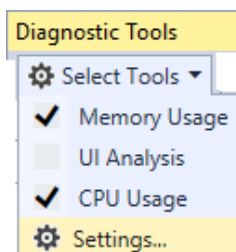


Gambar 12. 5 Penggunaan CPU.

12.1.2 Profiling Memory usage

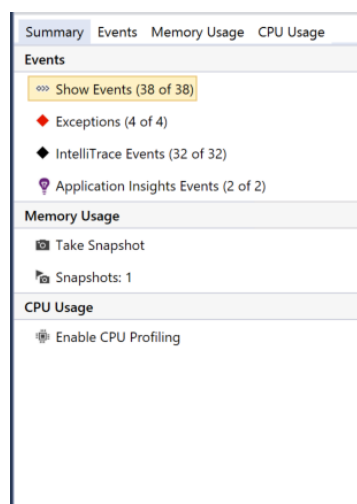
Berikut ini langkah langkah untuk mendapatkan profiling data Memory.

1. Buka project yang ingin didebug pada Visual Studio dan set breakpoint di aplikasi pada titik tempat kita ingin memeriksa penggunaan CPU.
2. Tetapkan breakpoint kedua di akhir fungsi atau wilayah kode yang ingin Anda analisis. Dengan mengatur dua breakpoint, Anda bisa membatasi pengumpulan data ke bagian kode yang ingin Anda analisis.
3. Diagnostic Tools window akan muncul secara otomatis kecuali Anda telah mematikannya. Untuk membuka kembali window tersebut, klik Debug> Windows> Show Diagnostic Tools.
4. Pilih memory usage pada select tools.



Gambar 12. 6 Memory Usage pada Select Tools.

5. Klik **Debug > Start Debugging** (atau **Start** pada toolbar, atau **F5**). Saat aplikasi selesai memuat, tampilan Ringkasan Alat Diagnostik muncul.



Gambar 12. 7 Start Debugging.

6. Untuk mengambil snapshot pada awal sesi debugging Anda, pilih Take snapshot pada toolbar ringkasan Penggunaan Memori.



Gambar 12. 8 Take Snapshot untuk mengambil snapshot pada awal I sesi debugging.

7. Jalankan skenario yang akan menyebabkan breakpoint pertama terkena.
8. Saat debugger terpause, ambil snapshoot, kemudian tekan F5 untuk menjalankan aplikasi lagi.
9. Pada saat debugger terpause lagi karena breakpoint kedua, ambil snapshoot lagi.

12.2 Unit Test

Unit tests biasanya tes otomatis yang ditulis dan dijalankan oleh pengembang perangkat lunak untuk memastikan bahwa bagian dari aplikasi (dikenal sebagai "unit") memenuhi desainnya dan berperilaku seperti yang diinginkan. Dalam pemrograman prosedural, sebuah unit bisa menjadi seluruh modul, tetapi lebih umum merupakan fungsi atau procedure individual. Dalam pemrograman berorientasi objek, unit sering kali merupakan seluruh antarmuka, seperti kelas, tetapi bisa menjadi metode individual. Dengan menulis tes terlebih dahulu untuk unit terkecil yang dapat diuji, kemudian perilaku gabungan di antara mereka, seseorang dapat membangun tes komprehensif untuk aplikasi yang kompleks.

Untuk mengisolasi masalah yang mungkin timbul, setiap kasus uji harus diuji secara independen. Pengganti seperti stub metode, objek tiruan, palsu, dan test harness dapat digunakan untuk membantu pengujian modul secara terpisah.

Selama pengembangan, pengembang perangkat lunak dapat menuliskan kriteria, atau hasil yang diketahui baik, ke dalam pengujian untuk memverifikasi kebenaran unit. Selama eksekusi kasus uji, framework mencatat pengujian yang gagal dalam kriteria apa pun dan melaporkannya dalam ringkasan. Untuk ini, pendekatan yang paling umum digunakan adalah uji - fungsi - nilai yang diharapkan.

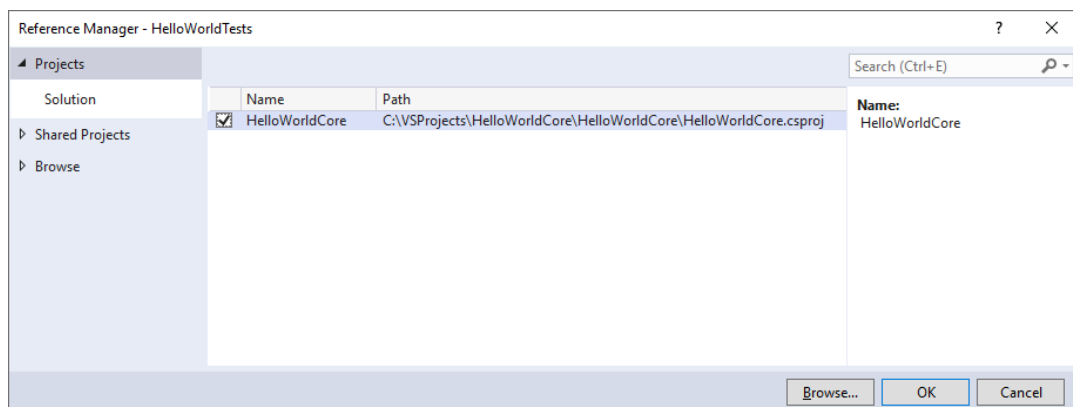
12.2.1 Membuat Unit Test

Berikut ini langkah langka untuk melakukan unit test

1. Buka the project yang akan dilakukan unit test di Visual Studio. Untuk Kemudahan contoh buatlah sebuah project baru bernama **“HelloWorldCore”** dan berisi code berikut:

```
namespace HelloWorldCore
{
    public class Program
    {
        public static void Main()
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

2. Pada **Solution Explorer**, pilih solution node. kemudian, pada bagian atas menu bar, select **File > Add > New Project**.
3. Pada new project dialog box, cari unit test yang akan digunakan yaitu MSTest lalu pilih itu dan klik next.
4. Pada project unit test, tambahkan referensi ke project yang ingin diuji dengan mengklik kanan pada Referensi atau Dependensi dan kemudian memilih Tambahkan Referensi.
5. Pilih project yang berisi kode yang akan diuji dan klik OK.



Gambar 12. 9 Reference Manager - HelloWorldTests.

6. Tambahkan code pada unit test method. Sebagai contoh tambahkan code seperti berikut

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System.IO;
using System;
namespace HelloWorldTests
{
    [TestClass]
    public class UnitTest1
    {
        private const string Expected = "Hello World!";
        [TestMethod]
        public void TestMethod1()
        {
            using (var sw = new StringWriter())
            {
                Console.SetOut(sw);
                HelloWorldCore.Program.Main();
                var result = sw.ToString().Trim();
                Assert.AreEqual(Expected, result);
            }
        }
    }
}
```

12.2.2 Run Unit Test

1. Buka Test Explorer (Test > Test Explorer from the top menu bar (atau Ctrl + E, T)).
2. Jalankan unit tests dengan mengklik **Run All** (atau **Ctrl + R, V**). Setelah tes selesai, tanda centang hijau menunjukkan bahwa tes berhasil. Ikon "x" merah menunjukkan bahwa tes gagal.

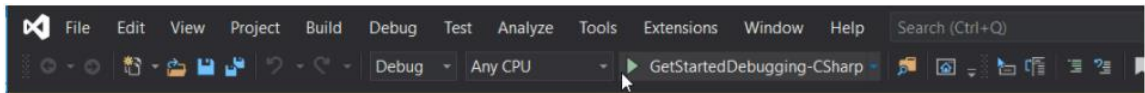
12.3 Debugging

Debugging adalah proses menemukan dan menyelesaikan bug (cacat atau masalah yang mencegah operasi yang benar) dalam program komputer, perangkat lunak, atau sistem. Taktik debugging dapat melibatkan debugging interaktif, analisis aliran kontrol, pengujian unit, pengujian integrasi, analisis file log, pemantauan pada aplikasi atau tingkat sistem, dump memori, dan

pembuatan profil. Banyak bahasa pemrograman dan alat pengembangan perangkat lunak juga menawarkan program untuk membantu debugging, yang dikenal sebagai debugger.

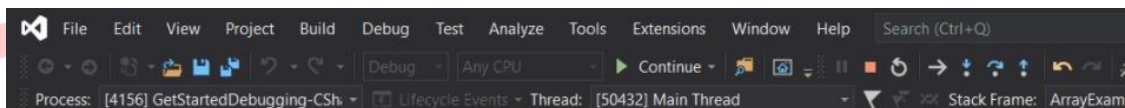
Berikut ini langkah langkah melakukan debugging pada visual studio:

1. Buka project yang ingin didebug.
2. Tambahkan breakpoint pada bagian yang ingin di debug dengan melakukan double klik pada nomor baris dari code tersebut hingga muncul bulatan berwarna merah.
3. Jalankan program dengan mengklik tombol Run.



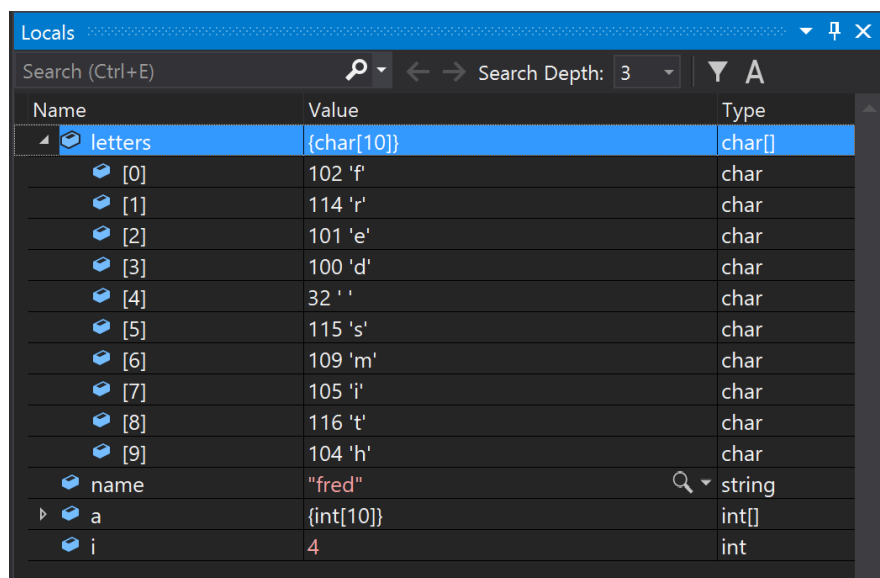
Gambar 12. 10 Run untuk menjalankan program.

4. Pada saat program berhenti karena terkena break point, arahkan kursor pada beberapa bagian code untuk mengetahui nilai variable yang telah tersimpan.
5. Untuk membuat program berjalan kembali, dapat dilakukan dengan mengklik tombol Continue.



Gambar 12. 11 Continue untuk kembali menjalankan program.

6. Pada local windows terdapat fitur untuk melihat nilai dari variable tersebut dengan menuliskan nama variabel tersebut pada local window.



Gambar 12. 12 Local windows.

MODUL 13 DESIGN PATTERN IMPLEMENTATION

TUJUAN PRAKTIKUM

1. Menguasai penerapan pattern singleton pada C#.Net.
2. Menguasai penerapan adapter pada C#.Net.
3. Menguasai penerapan command pada C#.Net.

13.1 Design patterns

Design patterns adalah sebuah solusi untuk sebuah problem yang sering terjadi pada software design. Hal tersebut adalah sebuah blueprints yang sudah terbuat sehingga bisa di sesuaikan untuk memecahkan sebuah masalah design yang terus terjadi pada kodingan kalian.

Kalian tidak bisa membuat solusi hanya dengan menemukan sebuah pattern dan copy pattern tersebut kedalam program kalian, tetapi dengan menggunakan perkumpulan library atau function. Pattern tersebut bukanlah sebuah program, tetapi sebuah konsep umum untuk memecahkan sebuah masalah yang spesifik. Kalian bisa mengikuti detail pattern dan mengimplementasi sebuah solusi yang cocok pada program yang kalian buat

Patterns sering di samakan dengan sebuah algoritma, karena dua konsep tersebut menggambarkan sebuah solusi untuk sebuah masalah. Walaupun sebuah algoritma selalu didefinisikan sebagai perkumpulan Tindakan yang dicapai untuk memenuhi tujuan, sedangkan sebuah pattern adalah sebuah deskripsi level yang tinggi pada sebuah solusi . sebuah code dari pattern yang sama diterapkan pada dua algoritma bisa saja berbeda.

Sebuah analogi dari algoritma adalah resep: memiliki sebuah step yang jelas untuk mencapai sebuah tujuan. Di samping itu, sebuah pattern adalah sebuah blueprint: kalian bisa melihat hasil yang seperti apa dan fitur yang seperti apa, tetapi urutan untuk mengimplementasi sesuai dengan kemauan kalian.

13.2 Pattern

Banyak pattern yang digambarkan dengan formal sehingga orang dapat memproduksi pattern dengan bermacam macam contex. Berikut adalah bagian yang biasa dipakai dalam definisi pattern:

- **Inten** dalam sebuah pattern dengan singkat menjelaskan masalah dan juga solusi
- **Motivasi** lebih dalam menjelaskan masalah dan solusi yang mungkin pada pattern.
- **Struktur** perkumpulan class menunjukan setiap bagian dari pattern dan bagaimana mereka terkait.

- **Contoh code** pada salah satu Bahasa pemrograman yang populer mempermudah untuk memahami ide pada pattern.

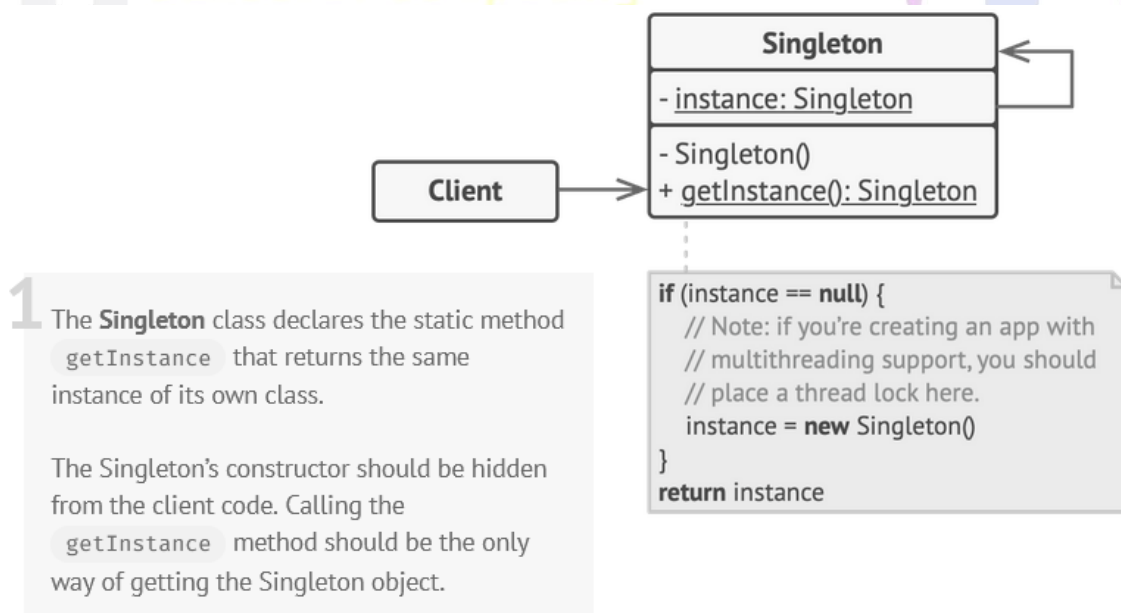
Beberapa katalog pattern memiliki hal penting yang lain, seperti penerapan pada sebuah pattern, Langkah implementasi dan relasi pada pattern lain nya.

Design patterns awalnya dikategorikan menjadi 3 sub-klasifikasi berdasarkan jenis masalah yang di pecahkan.

1. [Creational patterns](#) menyediakan kemampuan untuk membuat object sesuai dengan kriteria yang di butuhkan.
2. [Structural patterns](#) adalah tentang mengatur class dan object yang berbeda untuk membuat struktur yang besar dan menyediakan fungsionalitas yang baru.
3. [Behavioral patterns](#) adalah tentang mengidentifikasi komunikasi umum pada pattern antara object dan menyadari bahwa hal tersebut adalah pattern.

13.2.1 Singleton

Singleton adalah sebuah Creational pattern yang dapat kamu pastikan pada sebuah class yang hanya memiliki satu instance, Ketika memberikan akses point global untuk instance tersebut. berikut adalah contoh struktur singleton:



Gambar 13. 1 Contoh struktur singleton.

Berikut adalah contoh Pseduocode, dimana class connection berperan sebagai singleton. Class tersebut tidak memiliki public constructor, karena itu salah satu cara untuk memanggil object tersebut adalah dengan memanggil method *getInstance*. Method tersebut menyimpan object yang pertama dibuat dan mengembalikan panggilan selanjutnya. Berikut ini adalah contoh code singleton:

```
using System;

namespace Singleton
{
    // The Singleton class defines the `GetInstance` method that serves
    as an
    // alternative to constructor and lets clients access the same
    instance of
    // this class over and over.
    class Singleton
    {
        // The Singleton's constructor should always be private to
        prevent
        // direct construction calls with the `new` operator.
        private Singleton() { }

        // The Singleton's instance is stored in a static field. There
        there are
        // multiple ways to initialize this field, all of them have
        various pros
        // and cons. In this example we'll show the simplest of these
        ways,
        // which, however, doesn't work really well in multithreaded
        program.
        private static Singleton _instance;

        // This is the static method that controls the access to the
        singleton
        // instance. On the first run, it creates a singleton object and
        places
        // it into the static field. On subsequent runs, it returns the
        client
        // existing object stored in the static field.
        public static Singleton GetInstance()
```

```

    {
        if (_instance == null)
        {
            _instance = new Singleton();
        }
        return _instance;
    }

    // Finally, any singleton should define some business logic,
    which can
    // be executed on its instance.
    public static void someBusinessLogic()
    {
        // ...
    }
}

class Program
{
    static void Main(string[] args)
    {
        // The client code.
        Singleton s1 = Singleton.GetInstance();
        Singleton s2 = Singleton.GetInstance();

        if (s1 == s2)
        {
            Console.WriteLine("Singleton works, both variables
contain the same instance.");
        }
        else
        {
            Console.WriteLine("Singleton failed, variables contain
different instances.");
        }
    }
}

```


13.2.2 Adapter

Adapter adalah sebuah Structural design pattern, dimana memberikan object yang tidak cocok untuk di kolaborasikan. Adapter berperan sebagai pembungkus antara dua object. Mereka menyimpan panggilan untuk satu object dan merubahkannya kedalam bentuk format dan interface yang dapat di ketahui dengan object yang kedua. Contoh yang berada di bawah ini mengilustrasikan struktur pola design Adapter. Ini berfokus pada menjawab pertanyaan – pertanyaan dibawah:

- Terdiri dari class apa saja?
- Peran apakah yang ada tiap class?
- Dengan cara apa elemen pattern tersebut terkait?

```
using System;

namespace RefactoringGuru.DesignPatterns.Adapter.Conceptual
{
    // The Target defines the domain-specific interface used by the
    // client code.
    public interface ITarget
    {
        string GetRequest();
    }

    // The Adaptee contains some useful behavior, but its interface is
    // incompatible with the existing client code. The Adaptee needs some
    // adaptation before the client code can use it.
    class Adaptee
    {
        public string GetSpecificRequest()
        {
            return "Specific request.";
        }
    }

    // The Adapter makes the Adaptee's interface compatible with the
    // Target's
    // interface.
    class Adapter : ITarget
    {

```

```

        private readonly Adaptee _adaptee;

        public Adapter(Adaptee adaptee)
        {
            this._adaptee = adaptee;
        }

        public string GetRequest()
        {
            return $"This is '{this._adaptee.GetSpecificRequest()}';"
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Adaptee adaptee = new Adaptee();
            ITarget target = new Adapter(adaptee);

            Console.WriteLine("Adaptee interface is incompatible with the
client.");
            Console.WriteLine("But with adapter client can call it's
method.");

            Console.WriteLine(target.GetRequest());
        }
    }
}

```

Berikut adalah hasil output program :

```

Adaptee interface is incompatible with the client.
But with adapter client can call it's method.
This is 'Specific request.'

```

Gambar 13. 2 Output program.

13.2.3 Command

Command adalah sebuah Behavioral design pattern yang mengubah request atau operasi sederhana kedalam object. Contoh di bawah ini menggambarkan design pattern command. Ini berfokus pada menjawab pertanyaan – pertanyaan di bawah :

1. Terdiri dari class apa saja?
2. Peran apakah yang ada tiap class?
3. Dengan cara apa elemen pattern tersebut terkait?

```
using System;

namespace RefactoringGuru.DesignPatterns.Command.Conceptual
{
    // The Command interface declares a method for executing a command.
    public interface ICommand
    {
        void Execute();
    }

    // Some commands can implement simple operations on their own.
    class SimpleCommand : ICommand
    {
        private string _payload = string.Empty;

        public SimpleCommand(string payload)
        {
            this._payload = payload;
        }

        public void Execute()
        {
            Console.WriteLine($"SimpleCommand: See, I can do simple
things like printing ({this._payload})");
        }
    }

    // However, some commands can delegate more complex operations to
    other
```

```

// objects, called "receivers."
class ComplexCommand : ICommand
{
    private Receiver _receiver;

    // Context data, required for launching the receiver's methods.
    private string _a;

    private string _b;

    // Complex commands can accept one or several receiver objects
along
    // with any context data via the constructor.
    public ComplexCommand(Receiver receiver, string a, string b)
    {
        this._receiver = receiver;
        this._a = a;
        this._b = b;
    }

    // Commands can delegate to any methods of a receiver.
    public void Execute()
    {
        Console.WriteLine("ComplexCommand: Complex stuff should be
done by a receiver object.");
        this._receiver.DoSomething(this._a);
        this._receiver.DoSomethingElse(this._b);
    }
}

// The Receiver classes contain some important business logic. They
know how
// to perform all kinds of operations, associated with carrying out a
// request. In fact, any class may serve as a Receiver.
class Receiver
{
    public void DoSomething(string a)
    {
        Console.WriteLine($"Receiver: Working on ({a}.)");
    }
}

```

```

    }

    public void DoSomethingElse(string b)
    {
        Console.WriteLine($"Receiver: Also working on ({b}.)");
    }
}

// The Invoker is associated with one or several commands. It sends a
// request to the command.
class Invoker
{
    private ICommand _onStart;

    private ICommand _onFinish;

    // Initialize commands.
    public void SetOnStart(ICommand command)
    {
        this._onStart = command;
    }

    public void SetOnFinish(ICommand command)
    {
        this._onFinish = command;
    }

    // The Invoker does not depend on concrete command or receiver
    classes.

    // The Invoker passes a request to a receiver indirectly, by
    executing a
    // command.
    public void DoSomethingImportant()
    {
        Console.WriteLine("Invoker: Does anybody want something done
before I begin?");
        if (this._onStart is ICommand)
        {
            this._onStart.Execute();
        }
    }
}

```

```

    }

    Console.WriteLine("Invoker: ...doing something really
important...");

    Console.WriteLine("Invoker: Does anybody want something done
after I finish?");
    if (this._onFinish is ICommand)
    {
        this._onFinish.Execute();
    }
}

class Program
{
    static void Main(string[] args)
    {
        // The client code can parameterize an invoker with any
commands.
        Invoker invoker = new Invoker();
        invoker.SetOnStart(new SimpleCommand("Say Hi!"));
        Receiver receiver = new Receiver();
        invoker.SetOnFinish(new ComplexCommand(receiver, "Send
email", "Save report"));

        invoker.DoSomethingImportant();
    }
}

```

MODUL 14 CLEAN CODE

TUJUAN PRAKTIKUM
1. Menguasai penerapan clean code pada code C#.

14.1 Clean Code

Mari diawalkan modul ini dengan sebuah quote: “siapapun bisa menulis code yang bisa dipahami oleh computer. Programmer yang baik menulis code yang bisa dipahami oleh manusia – [Martin Fowler](#). Quote tersebut menjelaskan esensi dalam clean code.

Saat kita berbicara tentang kode bersih, kita berbicara tentang gaya pengembangan yang berfokus pada pembaca yang menghasilkan perangkat lunak yang mudah untuk ditulis, dibaca, dan dipelihara. Clean code adalah Code yang mudah dipahami dan mudah diubah.

Kata "Clean" telah menjadi sangat trendi saat ini jika Anda melihat desain, fotografi, dll. Orang mencari hal-hal bersih karena saat ini hidup kita sangat rumit dan kami ingin memilih opsi yang bersih dan jelas karena itu menenangkan kita dan menyelamatkan kita yang berharga waktu. Ini sama dalam pengembangan perangkat lunak dan arsitektur, jika Anda memiliki lebih banyak code daripada yang Anda butuhkan, seharusnya tidak ada, tidak boleh ada tambahan apa pun.

code Anda harus seefisien, mudah dibaca, dan mudah dipelihara, sebaik mungkin, walaupun hanya memecahkan masalah, Anda harus selalu meluangkan sedikit waktu ekstra untuk fokus pada desain code Anda, pada arsitektur. code Anda harus dapat dimengerti, harus bersih. Ini berarti kode mudah dibaca, baik pembaca itu adalah penulis asli kode atau orang lain. Tidak boleh ada keraguan dan kesalahpahaman. Misalnya, berikut ini harus jelas: alur eksekusi aplikasi, bagaimana objek yang berbeda berkolaborasi satu sama lain, peran dan tanggung jawab masing-masing kelas, setiap tujuan metode, tujuan dari setiap ekspresi dan variabel, dll.

Selain itu, sangat penting untuk memiliki kemampuan untuk memperluas dan *refactor* ulang code Anda dengan mudah. Ini dapat dicapai jika orang yang membuat perubahan memahami code dan juga merasa yakin bahwa perubahan yang dimasukkan dalam kode tidak merusak fungsionalitas yang ada. Agar code mudah diubah, Anda perlu memastikan bahwa Anda memperhitungkan bahwa kelas dan metode kecil dan hanya memiliki satu tanggung jawab, bahwa class memiliki API public yang jelas dan ringkas, class dan metode dapat diprediksi dan berfungsi seperti yang diharapkan, code mudah diuji dan memiliki pengujian unit, pengujian tersebut mudah dipahami dan mudah diubah, dll.

Yang penting untuk diingat adalah bahwa pembuat code yang bersih memastikan dia sepenuhnya memahami masalah sebelum mulai membuat kode. Ini seperti membangun rumah:

fondasi dan arsitektur adalah kuncinya! Dalam jangka panjang, ini akan menghemat waktu dan uang Anda untuk "mengulang" pekerjaan. Berikut ini beberapa penerapan clean code:

14.1.1 Penamaan Method

Metode harus memiliki nama kata kerja atau frase kata kerja seperti `postPayment`, `deletePage`, atau `save`. Aksesori, mutator, dan predikat harus diberi nama sesuai nilainya dan diawali dengan `get`, `set`, dan sesuai dengan standar `javabeans`. Saat konstruktor kelebihan beban, gunakan metode pabrik statis dengan nama yang menjelaskan argumen. Sebagai contoh :

```
// Lebih disarankan
Complex fulcrumPoint = Complex.FromRealNumber(23.0);

// Tidak Disarankan
Complex fulcrumPoint = new Complex(23.0);
```

14.1.2 Penamaan Variable

Nama ada di mana-mana dalam perangkat lunak. Kami memberi nama variabel kami, fungsi kami, argumen kami, kelas, dan paket kami. Kami memberi nama file sumber kami dan direktori yang memuatnya. Kami menamai file `jar` dan file `war` dan file `ear` kami. Kami memberi nama terus menerus. Karena kita melakukan begitu banyak hal, lebih baik kita melakukannya dengan baik. Berikut ini adalah beberapa aturan sederhana untuk membuat nama baik.

- Gunakan nama yang mengungkapkan tujuan
- Hindari disinformasi
- Buat perbedaan yang berarti
- Gunakan nama yang dapat diucapkan
- Gunakan nama yang mudah dicari
- Hindari encoding

Seperti contoh :

```
Int d; //waktu yang jalan
```


Nama variable d tidak mengungkapkan apa apa. Dan tidak menimbulkan perasaan waktu, maupun itu hari. Kita harus memberi nama yang mendefinisi apa yang di timbangkan dan unit pengukuran itu, seperti:

```
int elapsedTimeInDays;  
int daysSinceCreation;  
int daysSinceModification;  
int fileAgeInDays;
```



Fakultas Informatika
School of Computing
Telkom University



MODUL 15 Review Tugas Besar

TUJUAN PRAKTIKUM

1. Mampu melakukan evaluasi terhadap kualitas hasil konstruksi.

15.1 Tugas besar

Pada akhir perkuliahan Konstruksi Perangkat Lunak, anda bersama teman kelompok diwajibkan untuk mendemonstrasikan kemampuan untuk menerapkan apa saja yang sudah dipelajari selama perkuliahan maupun di praktikum. Selain dalam bentuk pengoperasian tools, penerapan teknologi, dan menghasilkan kode dengan baik, anda juga diminta untuk dapat mempertanggungjawabkan pemilihan tools, teknologi, dan pendekatan yang anda pakai untuk menyelesaikan masalah yang anda angkat.

Untuk memastikan bahwa anda sudah siap untuk menyelesaikan tugas besar anda silahkan isi checklist di bawah ini. Diskusikan dengan asisten praktikum apakah jawaban anda pada checklist ini sudah benar atau tidak.

No	Kriteria	Ya/Tidak
1	Requirement dari masalah yang saya angkat sudah terdefinisi dengan cukup sehingga dapat dipahami.	
2	Saya menggunakan IDE untuk membantu saya menerapkan code solusi terhadap requirement yang saya angkat.	
3	Saya mampu mempertanggungjawabkan pilihan saya untuk menggunakan atau tidak menggunakan GUI Builder.	
4	Saya menyimpan source code yang saya buat dalam sebuah repository github bersama kelompok saya.	
5	Teknologi yang saya terapkan pada tugas besar ini dipilih berdasarkan requirement yang saya punya dan saya dapat mempertanggungjawabkannya.	
6	Saya mampu mempertanggungjawabkan pilihan saya untuk menggunakan atau tidak menggunakan design pattern.	
7	Saya sudah menerapkan standar pemrograman yang baik sehingga source code yang saya buat dapat dipahami dan diperbaiki di waktu mendatang.	

8	Saya sudah melakukan unit testing menggunakan tools yang tersedia secara baik.	
9	Saya sudah melakukan pengukuran performa terhadap aplikasi yang saya bangun.	

Jika anda atau teman kelompok anda masih mendapatkan jawaban tidak di satu atau lebih kriteria, manfaatkan waktu yang tersisa hingga waktu presentasi akhir untuk memperbaiki pengerjaan tugas besar anda.



MODUL 16 Presentasi Tugas Besar

TUJUAN PRAKTIKUM
<ol style="list-style-type: none">1. Mampu mempertanggungjawabkan pemilihan tools konstruksi.2. Mampu mempertanggungjawabkan pemilihan teknik dan teknologi konstruksi.

16.1 Tugas besar

Pada presentasi tugas besar, anda diharuskan untuk menunjukkan kemampuan dalam menerapkan dan menjustifikasi penggunaan tools, teknik, teknologi, dan pendekatan terhadap konstruksi yang dilakukan. Presentasi akan dilakukan secara berkelompok dimana anda akan diminta menjelaskan apa saja yang sudah anda lakukan pada tugas besar mata kuliah Konstruksi Perangkat Lunak. Presentasi kemudian dilanjutkan dengan demo program, unit test, dan performance analysis.

Sesi terakhir presentasi adalah tanya jawab dimana secara individu anda akan diminta mempertanggungjawabkan keputusan-keputusan yang diambil selama menyelesaikan tugas besar mata kuliah Konstruksi Perangkat Lunak. Pada tanya jawab ini akan dinilai kemampuan anda untuk merasionalisasikan penggunaan teknologi-teknologi konstruksi yang ada berdasarkan kelebihan dan kekurangan dari teknologi tersebut.

16.2 Hal yang harus dipersiapkan

Saat presentasi pastikan anda sudah menyiapkan hal-hal berikut agar presentasi berjalan lancar:

1. Laporan dan tayangan presentasi yang menjelaskan apa saja yang sudah diterapkan pada tugas besar.
2. Source code dari perangkat lunak yang dibangun, tersimpan dalam repository.
3. Executable code dari perangkat lunak yang dibangun dan dapat didemonstrasikan pada saat presentasi.
4. Unit test code yang dapat didemonstrasikan pada saat presentasi.
5. Performance tools yang siap untuk dijalankan untuk menunjukan performa hasil konstruksi.

DAFTAR PUSTAKA

- [1] Tim Dosen MK SISTER. 2020.*Slide Perkuliahan Sister Paralel dan Terdistribusi*. Fakultas Informatika. Telkom University, Bandung.
- [2] "Graphical user interface builder", En.wikipedia.org, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Graphical_user_interface_builder. [Accessed: 11- Feb- 2021]
- [3] "Create a Windows Forms app with C# - Visual Studio", Docs.microsoft.com, 2021. [Online]. Available: <https://docs.microsoft.com/en-us/visualstudio/ide/create-csharp-winform-visual-studio?toc=%2Fvisualstudio%2Fget-started%2Fcsharp%2Ftoc.json&bc=%2Fvisualstudio%2Fget-started%2Fcsharp%2Fbreadcrumb%2Ftoc.json&view=vs-2019>. [Accessed: 11- Feb- 2021]
- [4] "GitHub", Id.wikipedia.org, 2021. [Online]. Available: https://id.wikipedia.org/wiki/GitHub#cite_note-hugeinvestment-4. [Accessed: 11- Feb- 2021]
- [5] "Design by contract", En.wikipedia.org, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Design_by_contract. [Accessed: 11- Feb- 2021]
- [6] "Defensive programming", En.wikipedia.org, 2021. [Online]. Available: https://en.wikipedia.org/wiki/Defensive_programming. [Accessed: 11- Feb- 2021]
- [7] "Parsing", En.wikipedia.org, 2021. [Online]. Available: <https://en.wikipedia.org/wiki/Parsing>. [Accessed: 11- Feb- 2021]
- [8] "Create a .NET class library using Visual Studio - .NET", Docs.microsoft.com, 2021. [Online]. Available: <https://docs.microsoft.com/en-us/dotnet/core/tutorials/library-with-visual-studio>. [Accessed: 11- Feb- 2021]
- [9] "Clean Code: Explanation, Benefits, and Examples – Dzone Agile", dzone.com, 2021. [Online]. Available: <https://dzone.com/articles/clean-code-explanation-benefits-amp-examples>. [Accessed: 11- Feb- 2021]
- [10] "What's a design pattern?", Refactoring.guru, 2021. [Online]. Available: <https://refactoring.guru/design-patterns/what-is-pattern>. [Accessed: 11- Feb- 2021]
- [11] "Code Contracts", Docs.microsoft.com, 2021. [Online]. Available: <https://docs.microsoft.com/en-us/dotnet/framework/debug-trace-profile/code-contracts>. [Accessed: 11- Feb- 2021]
- [12] "Generics - C# Programming Guide", Docs.microsoft.com, 2021. [Online]. Available: <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/generics/#generics-overview>. [Accessed: 11- Feb- 2021]
- [13] "Assertions in Managed Code - Visual Studio", Docs.microsoft.com, 2021. [Online]. Available: <https://docs.microsoft.com/en-us/visualstudio/debugger/assertions-in-managed-code?view=vs-2019>. [Accessed: 11- Feb- 2021]
- [14] "Exception Handling - C# Programming Guide", Docs.microsoft.com, 2021. [Online]. Available: <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/exceptions/exception-handling>. [Accessed: 11- Feb- 2021]

- [15] "How to serialize and deserialize JSON using C# - .NET", Docs.microsoft.com, 2021. [Online]. Available: <https://docs.microsoft.com/en-us/dotnet/standard/serialization/system-text-json-how-to?pivot=dotnet-5-0>. [Accessed: 11- Feb- 2021]
- [16] "JSON", Id.wikipedia.org, 2021. [Online]. Available: <https://id.wikipedia.org/wiki/JSON>. [Accessed: 11- Feb- 2021]
- [17] "Measure CPU usage in your apps - Visual Studio", Docs.microsoft.com, 2021. [Online]. Available: <https://docs.microsoft.com/en-us/visualstudio/profiling/beginners-guide-to-performance-profiling?view=vs-2019>. [Accessed: 11- Feb- 2021]
- [18] "Measure memory usage in your apps - Visual Studio", Docs.microsoft.com, 2021. [Online]. Available: <https://docs.microsoft.com/en-us/visualstudio/profiling/memory-usage?view=vs-2019>. [Accessed: 11- Feb- 2021]
- [19] "Get started with unit testing - Visual Studio", Docs.microsoft.com, 2021. [Online]. Available: <https://docs.microsoft.com/en-us/visualstudio/test/getting-started-with-unit-testing?view=vs-2019&tabs=mstest>. [Accessed: 11- Feb- 2021]
- [20] "Create a .NET class library using Visual Studio - .NET", Docs.microsoft.com, 2021. [Online]. Available: <https://docs.microsoft.com/en-us/dotnet/core/tutorials/library-with-visual-studio>. [Accessed: 11- Feb- 2021]
- [21] ["Library (computing)", En.wikipedia.org, 2021. [Online]. Available: [https://en.wikipedia.org/wiki/Library_\(computing\)#Code_generation_libraries](https://en.wikipedia.org/wiki/Library_(computing)#Code_generation_libraries). [Accessed: 11- Feb- 2021]
- [22] "API", En.wikipedia.org, 2021. [Online]. Available: https://en.wikipedia.org/wiki/API#See_also. [Accessed: 11- Feb- 2021]
- [23] "ASP.NET Core web API documentation with Swagger / OpenAPI", Docs.microsoft.com, 2021. [Online]. Available: <https://docs.microsoft.com/en-us/aspnet/core/tutorials/web-api-help-pages-using-swagger?view=aspnetcore-5.0>. [Accessed: 11- Feb- 2021]
- [24] "Get started with Swashbuckle and ASP.NET Core", Docs.microsoft.com, 2021. [Online]. Available: <https://docs.microsoft.com/en-us/aspnet/core/tutorials/getting-started-with-swashbuckle?view=aspnetcore-5.0&tabs=visual-studio#package-installation>. [Accessed: 11- Feb- 2021]
- [25] "Get started with NSwag and ASP.NET Core", Docs.microsoft.com, 2021. [Online]. Available: <https://docs.microsoft.com/en-us/aspnet/core/tutorials/getting-started-with-nswag?view=aspnetcore-5.0&tabs=visual-studio#register-the-nswag-middleware>. [Accessed: 11- Feb- 2021]
- [26] "Run-time config options - .NET", Docs.microsoft.com, 2021. [Online]. Available: <https://docs.microsoft.com/en-us/dotnet/core/run-time-config/#runtimeconfigjson>. [Accessed: 11- Feb- 2021]
- [27] "Configure portable object localization in ASP.NET Core", Docs.microsoft.com, 2021. [Online]. Available: <https://docs.microsoft.com/en-us/aspnet/core/fundamentals/portable-object-localization?view=aspnetcore-5.0>. [Accessed: 11- Feb- 2021]

PENUTUP

Penyusunan modul ini ditujukan untuk menjadi pedoman pelaksanaan praktikum pada mata kuliah Konstruksi Perangkat Lunak Fakultas Informatika Telkom University. Materi-materi yang terdapat dalam modul ini diharapkan dapat membantu mahasiswa dalam mengimplementasikan apa yang telah dipelajari di kelas, sehingga tidak hanya sekedar mempelajari secara teori namun juga praktek secara langsung.

Modul ini memuat beberapa pokok pembahasan yaitu penggunaan *github*, penggunaan *gui builder*, *automata* dan *table-driven construction*, *generics*, *design by contract*, *runtime configuration*, *API*, *unit testing*, dan *clean code*. Pada setiap materinya diberikan beberapa pembahasan singkat dan juga contoh-contoh sederhana untuk mempermudah dalam memahami materi tersebut.

Manusia tidak luput dari kekurangan dan kesalahan. Maka dari itu, tim penyusun modul bersifat terbuka terhadap segala kritik dan saran yang mendorong kemajuan penyusunan modul ini.

Elok rupawan si buah hati

Paras bagai mentari fajar

Lab Informatika Telkom University

Mengucapkan selamat belajar



Fakultas Informatika
School of Computing
Telkom University



DAFTAR PERUBAHAN

12 Februari 2022

Penerbitan ulang berdasarkan Modul Praktikum Konstruksi Perangkat Lunak – S1 Rekayasa Perangkat Lunak Genap 20/21 oleh Jati Hiliamsyah Husen, S.T., M.Eng. dengan perubahan pada Modul 8 - Api Design Dan Construction Using Swagger oleh Muhammad Johan Alibasa, S.T., M.T., Ph.D.





Kontak Kami :



@dky2921g



@informaticslab



@informaticslab_telu



informaticslab@telkomuniversity.ac.id



informatics.labs.telkomuniversity.ac.id