PETUNJUK PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBYEK

Oleh:

Achmad Maududie
Dwiretno Istiyadi Swasono
Mohammad Zarkasi
Tio Dharmawan

PROGRAM STUDI SISTEM INFORMASI / INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS JEMBER
2023

LEMBAR PENGESAHAN

Nama Matakuliah : Pemrograman Berorientasi Obyek
Kode : KSU 1208
Program Studi : Sistem Informasi / Teknologi Informasi / Informatika
Fakultas : Fakultas Ilmu Komputer

Tim Penyusun

Koordinator

a. Nama : Achmad Maududie
b. NIP/NRP : 197004221995121001
c. Program Studi : Sistem Informasi
d. Fakultas : Ilmu Komputer

Anggota 1

a. Nama : Dwiretno Istiyadi Swasono
b. NIP/NRP : 197803302003121003
c. Program Studi : Sistem Informasi
d. Fakultas : Ilmu Komputer

Anggota 2

a. Nama : M. Zarkasi
b. NIP/NRP : 199011112019031018
c. Program Studi : Teknologi Informasi
d. Fakultas : Ilmu Komputer

Anggota 3

a. Nama : Tio Dharmawan
b. NIP/NRP : 199111122022031011
c. Program Studi : Informatika
d. Fakultas : Ilmu Komputer

Mengetahui,
Wakil Dekan 1

Jember, 12-11-2023
Koordinator,

Achmad Maududie, ST., M.Sc.
NIP: 197004221995121001

Achmad Maududie, ST., M.Sc.
NIP: 197004221995121001

PRAKATA

Puji dan syukur kami haturkan kepada Tuhan Yang Maha Kuasa, karena atas berkat dan petunjuk-Nyalah, maka seluruh proses penyusunan buku Petunjuk Praktikum algoritma dan pemrograman 2 dapat terlaksana. Buku Petunjuk Praktikum ini berisi tentang panduan dari sejumlah kegiatan praktikum yang bertujuan untuk memberikan gambar nyata dari teori-teori yang telah diberikan melalui tatap muka di dalam kelas.

Petunjuk di setiap kegiatan praktikum dalam buku ini, disusun dengan mengikuti struktur sebagai berikut, yiatu: 1) Capaian Pembelajaran, 2) Tujuan Percobaan, 3) Tinjauan Pustaka, 4) Alat dan Bahan, 4) Prosedur Kerja, 5) Analisis Data, 6) Kesimpulan, 7) Daftar Pustaka, dan 8) Hasil Penilaian. Dengan struktur tersebut diharapkan mahasiswa mendapatkan kesempatan dalam melakukan percobaan, belajar menganalisis hasil percobaan, serta menarik kesimpulan untuk membuat gambaran umum tentang hasil percobaan, analisis, pembahasan, dan/atau pengujian hipotesa yang ada.

Tim penyusun buku Petunjuk Praktikum <nama matakuliah> sadar bahwa buku ini masih belum sempurna. Oleh karena itu, kami sangat mengharapkan kritik dan saran yang konstruktif dari para pembaca untuk perbaikan panduan praktikum. Semoga seluruh niat baik kita, diberkati oleh Tuhan Yang Maha Kuasa.

Jember, 12-11-2023

Tim Penyusun
Buku Petunjuk Praktikum
Pemrograman Berorientasi Obyek

DAFTAR ISI

TATA TERTIB PRAKTIKUM

1. Praktikan wajib mengenakan pakaian rapi standar kuliah dan memberikan perhatian penuh pada kegiatan praktikum saat pelaksanaannya.
2. Praktikan diwajibkan untuk menjaga ketenangan dalam pelaksanaan praktikum.
3. Praktikan dilarang meninggalkan ruangan tanpa seizin asisten.
4. Praktikan yang datang terlambat lebih dari 15 menit akan dikenakan sanksi.
5. Praktikan yang tidak mengikuti praktikum tanpa keterangan tidak akan mendapat nilai.
6. Praktikan wajib melakukan presensi menggunakan tanda tangan (non-mmp)
7. Segala bentuk pelanggaran terkait poin 1 s/d 4 akan dikenakan sanksi berupa pengurangan nilai).

1. CLASS DIAGRAM

a. Capaian Pembelajaran

Kode CPL	:	CPL 06
Nama CPL	:	Mampu memvalidasi berbagai teknologi yang digunakan dalam pembuatan suatu program, aplikasi dan atau sistem informasi.
Kode CPMK	:	CPMK 04
Nama CPMK	:	Mampu merancang aplikasi menggunakan paradigma OOP secara penuh
Nama Sub-CPMK	:	mahasiswa mampu mendesain algoritma rekursif untuk menyelesaikan persoalan algoritma tertentu

b. Tujuan Percobaan

Dalam praktikum class diagram pada mata kuliah pemrograman berorientasi objek, terdapat beberapa tujuan utama:

1. Pemahaman Konsep OOP:

   Memberikan pemahaman yang lebih mendalam tentang konsep-konsep fundamental dalam Pemrograman Berorientasi Obyek (OOP). Ini termasuk kelas, objek, enkapsulasi, pewarisan, polimorfisme, dan asosiasi antar kelas.

2. Representasi Visual Struktur Program:

   Class diagram digunakan untuk merepresentasikan secara visual struktur program OOP. Ini memungkinkan para mahasiswa untuk memahami bagaimana kelas-kelas terkait dan berinteraksi satu sama lain dalam suatu program.

3. Pembelajaran Penggunaan Tool Modelling:

   Memperkenalkan penggunaan alat pemodelan seperti UML (Unified Modeling Language) untuk membuat class diagram. Ini memungkinkan

mahasiswa memahami cara menggunakan alat ini untuk merancang dan merepresentasikan desain program secara visual.

4. Pengembangan Kemampuan Desain:

   Memberikan latihan dalam desain struktur program berbasis OOP. Mahasiswa diberi tugas untuk merancang class diagram untuk kasus-kasus tertentu yang memungkinkan mereka untuk berpikir secara logis dan merencanakan struktur program dengan baik.

5. Kolaborasi dan Asosiasi Antar Kelas:

   Menyoroti hubungan dan interaksi antar kelas dalam suatu sistem. Melalui class diagram, mahasiswa belajar bagaimana kelas-kelas berkolaborasi, mewarisi, dan terhubung dalam suatu proyek atau sistem.

Dengan fokus pada representasi visual, pemahaman konsep OOP, dan penggunaan alat pemodelan, praktikum class diagram membantu memperdalam pemahaman mahasiswa tentang OOP dan meningkatkan kemampuan desain program berbasis objek.

c. Tinjauan Pustaka

Class diagram adalah jenis diagram dalam Unified Modeling Language (UML) yang digunakan untuk menggambarkan struktur dan hubungan antara kelas dalam suatu sistem berbasis objek. Class diagram memberikan representasi visual tentang entitas (kelas) dalam suatu sistem perangkat lunak serta hubungan dan interaksi di antara kelas-kelas tersebut

Komponen utama class diagram, yaitu:

1. Kelas: Merepresentasikan entitas dalam sistem. Setiap kelas memiliki atribut (variabel kelas) dan metode (fungsi atau perilaku).

2. Atribut: Variabel-variabel yang dimiliki oleh suatu kelas. Mereka mendefinisikan sifat dari suatu kelas.
3. Metode: Fungsi-fungsi atau perilaku dari suatu kelas. Mereka mendefinisikan apa yang bisa dilakukan oleh suatu kelas.
4. Relasi Antar Kelas: Hubungan antara kelas dalam sistem. Ini bisa berupa pewarisan, asosiasi, agregasi, atau komposisi.

Jenis relasi antar kelas dalam Class Diagram:

1. Pewarisan (Inheritance): Hubungan antara kelas induk (superclass) dan kelas anak (subclass). Subclass mewarisi atribut dan metode dari superclass.
2. Asosiasi Hubungan yang menunjukkan bagaimana kelas-kelas terhubung satu sama lain. Mereka bisa memiliki hubungan satu-ke-satu, satu-ke-banyak, atau banyak-ke-banyak.
3. Agregasi dan Komposisi: Menunjukkan bagaimana kelas-kelas berhubungan satu sama lain dalam konteks tertentu. Agregasi adalah hubungan yang lebih longgar, sedangkan komposisi adalah hubungan yang lebih kuat.

Class diagram memberikan gambaran yang jelas tentang struktur kelas dalam sistem dan bagaimana kelas-kelas ini saling berinteraksi. Ini memungkinkan para pengembang untuk merencanakan desain sistem dan memahami relasi antara entitas-entitas yang terlibat dalam sistem yang akan dibangun.

d. Alat Dan Bahan

Kebutuhan alat dalam melakukan percobaan:

1. Laptop / Komputer
2. Visual Paradigm

e. Prosedur Kerja

Percobaan 1: Membuat Class Diagram

1. Buka Visual Paradigm.
2. Pilih "New Project”
3. Setelah proyek dibuat, pilih "Diagram" di bagian kiri
4. Klik kanan, pilih "New" > "Class Diagram" atau buka proyek, lalu pilih "Class Diagram" dari daftar.
5. Di class diagram, klik kanan lalu pilih "Class" dari menu yang muncul.
6. Beri nama kelas "Customer" dan tekan Enter.
7. Klik dua kali pada kelas untuk membuka kotak dialog
8. Tambahkan atribut (misalnya, "name: String") dan metode (misalnya, "getName(): String") pada tab yang sesuai.
1.
2. ### Langkah 5: Hubungkan Kelas
3.
4. 1. Gunakan ikon "Association" untuk menghubungkan dua kelas. Tarik dari satu kelas ke kelas lainnya untuk membuat hubungan.
5. 2. Atur properti hubungan (misalnya, association, aggregation, atau inheritance) sesuai kebutuhan.
6.
7. ### Langkah 6: Menyimpan dan Mengelola
8.
9. 1. Pastikan untuk menyimpan proyek Anda secara teratur.
10. 2. Untuk mengelola class diagram, Anda bisa mengatur layout, menambahkan notasi, dan memformat elemen sesuai kebutuhan.
11.
12. ### Catatan:
13. Pastikan untuk menggunakan panduan yang disediakan oleh Visual Paradigm sesuai versi yang Anda gunakan karena antarmuka pengguna dapat sedikit berbeda antar versi.
14.
15. Tutorial Visual Paradigm dapat memberikan panduan yang lebih rinci dan lengkap tentang penggunaan alat pemodelan mereka. Tutorial ini dapat

ditemukan di situs web mereka atau dalam dokumentasi yang disertakan dengan perangkat lunak.

f. Hasil Dan Analisis Data

Berdasarkan seluruh percobaan yang terlah dilakukan, maka hasil dan pembahasan berdasarkan teori pendukung atas setiap hasilnya disajikan berikut ini.

Percobaan 1: Instalasi extensi Python dalam Visual Studio Code

1. Tampilkan hasil *percobaan 1* pada langkah 3 dan berikan penjelasan atas hasil saat anda mengetikkan “Python” pada text box “Search Extensions in Marketplace”.

g. Kesimpulan

Buatlah kesimpulan untuk bab ini berdasarkan semua percobaan yang telah dilakukan.

h. Tugas Mandiri

Buatlah program sederhana untuk menampilkan nim, nama lengkap, tanggal lahir, alamat tinggal, nama program studi, nama fakultas, dan lima nama lengkap teman anda satu angkatan.

i. Daftar Pustaka

Uzayr, S.B., 2021. Optimizing Visual Studio Code for Python Development. Apress.

j. Hasil Penilaian

Identitas Penilai (Asisten Praktikum)

NIM	
Nama	

Penilaian

No	Item	Skor
1	Pre-test (maks 5)	
2	Hasil analisis data (maks 30)	
3	Penarikan kesimpulan (maks 25)	
4	Tugas Mandiri (maks 15)	
5	Post-test (mask 15)	
6	Sikap (maks 10)	
	Total Nilai	

Jember, <tgl> <bulan> <tahun>

<Nama Lengkap Penilai>
NIM: xxxxxxxxxxxxxxx

2. INHERITANCE

a. Capaian Pembelajaran

1. Kode CPL : CPL 06
2. Nama CPL : Mampu memvalidasi berbagai teknologi yang digunakan dalam pembuatan suatu program, aplikasi dan atau sistem informasi.
3. Kode CPMK : CPMK 04
4. Nama CPMK : Mampu merancang aplikasi menggunakan paradigma OOP secara penuh
5. Nama Sub-CPMK : mahasiswa mampu mendesain algoritma rekursif untuk menyelesaikan persoalan algoritma tertentu

a. Tujuan Percobaan

Inheritance adalah konsep fundamental dalam Pemrograman Berorientasi Objek (OOP). Tujuan utama percobaan ini adalah untuk memberikan pemahaman mendalam tentang cara inheritance bekerja, bagaimana kelas dapat mewarisi properti dan perilaku dari kelas lain, dan bagaimana hubungan hierarki antara kelas-kelas tersebut dapat dibentuk.

Reusabilitas dan Organisasi Kode

Pewarisan memungkinkan penggunaan kembali kode yang ada. Dengan mewarisi sifat dan perilaku dari kelas induk (superclass), kelas anak (subclass) dapat memanfaatkan fitur-fitur yang telah didefinisikan sebelumnya. Ini meningkatkan reusabilitas kode dan mengurangi duplikasi, memperbaiki, dan memelihara kode.

Polimorfisme

Pemahaman tentang konsep polimorfisme yang terkait erat dengan inheritance. Polimorfisme memungkinkan objek dari kelas yang berbeda untuk diakses melalui antarmuka yang sama. Dalam percobaan inheritance, siswa dapat

mempelajari bagaimana polimorfisme dapat diterapkan melalui penggunaan pewarisan

Desain dan Struktur Program

Percobaan inheritance membantu mahasiswa memahami bagaimana mendesain struktur program dengan hierarki kelas yang terorganisir. Ini memungkinkan mereka untuk merancang program dengan cara yang logis dan terstruktur, memisahkan dan mengelompokkan sifat dan perilaku yang serupa.

Penggunaan Alat Modelling

Percobaan ini juga memperkenalkan penggunaan alat pemodelan seperti UML untuk merepresentasikan konsep inheritance dalam bentuk class diagram. Ini membantu siswa untuk memahami bagaimana inheritance direpresentasikan secara visual dan bagaimana kelas-kelas terkait dalam hierarki.

Pemahaman inheritance penting dalam OOP karena memungkinkan penggunaan kode yang efisien, organisasi yang baik, dan perencanaan desain program yang berkualitas. Ini adalah dasar yang penting dalam pengembangan perangkat lunak berorientasi objek.

b. Tinjauan Pustaka

Inheritance (pewarisan) adalah konsep dalam pemrograman berorientasi objek di mana sebuah kelas (subclass atau turunan) dapat mewarisi sifat dan perilaku dari kelas lain (superclass atau induk). Dalam inheritance, subclass dapat menggunakan properti (atribut) dan metode (fungsi) dari superclass tanpa perlu menulis ulang kode yang sama.

1. Hubungan Hierarki: Inheritance menciptakan hierarki antara kelas-kelas. Kelas yang mewarisi disebut subclass atau turunan, sedangkan kelas yang memberikan warisan disebut superclass atau induk.

2. Pewarisan Atribut dan Metode: Subclass dapat mewarisi semua atribut dan metode dari superclass. Ini memungkinkan kelas untuk memperluas atau menambah fungsionalitas dari superclass.
3. Polimorfisme: Inheritance memungkinkan polimorfisme, di mana objek dari kelas yang berbeda dapat diakses melalui antarmuka yang sama. Hal ini memungkinkan fleksibilitas dalam penggunaan objek dan metode.
4. Reusabilitas Kode: Dengan inheritance, kode dapat digunakan kembali. Subclass dapat menggunakan sifat dan perilaku yang telah didefinisikan di superclass, mengurangi duplikasi kode.
5. Penggunaan 'is-a' Relationship: Inheritance menggambarkan hubungan 'is-a' di mana subclass adalah tipe dari superclass. Misalnya, "kucing" adalah tipe dari "hewan".

Dalam praktikum atau pengajaran pemrograman, inheritance penting karena memungkinkan struktur yang terorganisir dalam pembangunan program, memungkinkan reusabilitas kode yang efisien, dan meningkatkan fleksibilitas dan skalabilitas dalam pengembangan aplikasi.

d. Alat Dan Bahan

Alat yang dibutuhkan:

1. Komputer / Laptop
2. Program Visual Studio

e. Prosedur Kerja

Prosedur kerja dilakukan melalui langkah-langkah berikut:

1. Buka Visual Studio.
2. Pilih "Create a new project" atau "File" > "New" > "Project".
3. Pilih "Console App (.NET Core)" atau "Console App (.NET Framework)".
4. Di jendela solusi (Solution Explorer), klik kanan pada nama proyek.
5. Pilih "Add" > "Class" untuk membuat file kelas baru.
6. Tuliskan kode program berikut

```
1: using System;
2:
3: // Superclass atau kelas induk
4: class Vehicle
5: {
6:     public void Start()
7:     {
8:         Console.WriteLine("Vehicle is starting.");
9:     }
10:
11:     public void Stop()
12:     {
13:         Console.WriteLine("Vehicle is stopping.");
14:     }
15: }
16:
17: // Subclass atau kelas turunan
18: class Car : Vehicle
19: {
20:     public void Accelerate()
21:     {
22:         Console.WriteLine("Car is accelerating.");
23:     }
24:
25:     public void Brake()
26:     {
27:         Console.WriteLine("Car is braking.");
28:     }
29: }
30:
31: class Program
32: {
33:     static void Main(string[] args)
34:     {
35:         // Membuat objek dari kelas Car (subclass)
36:         Car myCar = new Car();
37:
38:         // Memanggil metode dari superclass
39:         myCar.Start();
40:         myCar.Stop();
41:
42:         // Memanggil metode dari subclass
43:         myCar.Accelerate();
```

```
44:         myCar.Brake();
45:     }
46: }
```

7. Tekan tombol `Ctrl + F5` atau pilih "Debug" > "Start Without Debugging" untuk menjalankan program.
8. Jendela konsol akan muncul dan menampilkan output dari program.

f. Hasil dan Analisis Data

Hasil dan analisis data dilakukan melalui beberapa langkah:

1. Screenshot luaran dari program.
2. Jelaskan setiap baris program.
3. Analisis dan jelaskan bagaimana konsep inheritance diterapkan pada contoh program diatas.

g. Kesimpulan

Simpulkan apa keuntungan dan kerugian dalam penggunaan konsep inheritance.

h. Tugas Mandiri

Rancang dan buatlah program yang menerapkan konsep inheritance. Laporan tugas praktikum yang terdiri dari:

1. Class Diagram
2. Kode Program
3. Luaran Program
4. Hasil Analisis

i. Daftar Pustaka

Farrel, J. 2017. Microsoft Visual C#: An Introduction to Object Oriented Programming.Cengage Learning.

j. Hasil Penilaian

Identitas Penilai (Asisten Praktikum)

NIM	
Nama	

Penilaian

No	Item	Skor
1	Pre-test (maks 5)	
2	Hasil analisis data (maks 30)	
3	Penarikan kesimpulan (maks 25)	
4	Tugas Mandiri (maks 15)	
5	Post-test (mask 15)	
6	Sikap (maks 10)	
	Total Nilai	

Jember, <tgl> <bulan> <tahun>

<Nama Lengkap Penilai>
NIM: xxxxxxxxxxxxxxx

6. POLYMORPHISM

a. Capaian Pembelajaran

1. Kode CPL : CPL 06
2. Nama CPL : Mampu memvalidasi berbagai teknologi yang digunakan dalam pembuatan suatu program, aplikasi dan atau sistem informasi.
3. Kode CPMK : CPMK 04
4. Nama CPMK : Mampu merancang aplikasi menggunakan paradigma OOP secara penuh
5. Nama Sub-CPMK : mahasiswa mampu mendesain algoritma rekursif untuk menyelesaikan persoalan algoritma tertentu

a. Tujuan Percobaan

Praktikum mengenai polymorphism dalam pemrograman berorientasi objek memiliki beberapa tujuan kunci:

Pemahaman Konsep Polimorfisme

Polimorfisme adalah salah satu prinsip penting dalam pemrograman berorientasi objek. Tujuan praktikum adalah untuk memberikan pemahaman yang mendalam tentang konsep ini. Polimorfisme memungkinkan objek dari kelas yang berbeda untuk diakses melalui antarmuka yang sama.

Implementasi Melalui Inheritance

Praktikum polymorphism sering kali berkaitan dengan inheritance. Tujuannya adalah menunjukkan bagaimana polimorfisme diekspresikan melalui pewarisan (inheritance) di mana kelas anak dapat menggunakan metode kelas induk tanpa perlu mengubah atau menimpa metode tersebut.

Penggunaan Interface dan Abstraksi

Praktikum ini juga bertujuan untuk memperkenalkan konsep interface dan abstraksi yang mendukung polimorfisme. Mahasiswa dapat memahami bagaimana

interface memungkinkan penggunaan polimorfisme untuk membuat objek dari kelas yang berbeda tetapi memiliki perilaku serupa.

Desain dan Struktur Program yang Fleksibel

Dengan memahami polimorfisme, praktikum membantu mahasiswa untuk merancang struktur program yang lebih fleksibel dan mudah diubah. Hal ini memungkinkan pengembang untuk menyesuaikan dan memperluas fungsionalitas tanpa merusak bagian yang sudah ada.

Peningkatan Reusabilitas Kode

Polimorfisme memungkinkan penggunaan kembali kode yang ada dan mengurangi duplikasi. Praktikum bertujuan untuk menunjukkan bagaimana polimorfisme meningkatkan reusabilitas kode dan mempromosikan pengembangan perangkat lunak yang lebih efisien.

Penggunaan dalam Dunia Nyata

Praktikum juga sering kali memberikan contoh kasus nyata atau studi kasus yang menunjukkan bagaimana polimorfisme digunakan dalam aplikasi nyata, membantu mahasiswa memahami aplikasi praktis dari konsep ini

b. Tinjauan Pustaka

Polymorphism adalah konsep dalam pemrograman berorientasi objek di mana suatu objek dapat memiliki banyak bentuk atau perilaku. Secara harfiah, "polymorphism" berasal dari kata Yunani yang berarti "banyak bentuk".

Terdapat dua jenis polimorfisme

1. Polimorfisme Compile-Time (atau Early Binding): Polimorfisme ini terjadi selama kompilasi. Contohnya adalah method overloading di mana nama method tetap sama, tetapi parameter atau tipe data yang berbeda.
2. Polimorfisme Run-Time (atau Late Binding): Polimorfisme ini terjadi saat runtime. Ini umumnya terkait dengan inheritance dan method overriding di mana method yang sama dipanggil dari objek berbeda.

Contoh Polimorfisme:

1. Method Overloading: Dalam C#, method `calculate` bisa memiliki beberapa versi berdasarkan parameter yang diberikan, misalnya, `calculate(int a)`, `calculate(int a, int b)`, atau `calculate(double a)`.
2. Method Overriding: Dalam inheritance, subclass dapat mengganti (override) metode dari superclass. Sebagai contoh, sebuah method `draw()` pada kelas `Shape` dapat di-override oleh kelas `Circle` atau `Square` yang mewarisi dari `Shape`.

Polimorfisme memungkinkan menulis kode yang lebih fleksibel dan reusable. Hal ini juga memungkinkan berbagai objek dari kelas yang berbeda untuk digunakan dengan cara yang seragam melalui antarmuka yang sama. Ini mempermudah penulisan kode dan meningkatkan fleksibilitas dalam pengembangan perangkat lunak.

d. Alat Dan Bahan

Alat yang dibutuhkan:

1. Komputer / Laptop
2. Program Visual Studio

e. Prosedur Kerja

Prosedur kerja dilakukan melalui langkah-langkah berikut:

1. Buka Visual Studio.
2. Pilih "Create a new project" atau "File" > "New" > "Project".
3. Pilih "Console App (.NET Core)" atau "Console App (.NET Framework)".
4. Di jendela solusi (Solution Explorer), klik kanan pada nama proyek.
5. Pilih "Add" > "Class" untuk membuat file kelas baru.
6. Tuliskan kode program berikut

```
1: using System;
2:
3: class Calculator
4: {
5:     public int Add(int a, int b)
6:     {
7:         return a + b;
8:     }
9:
10:     public double Add(double a, double b)
11:     {
12:         return a + b;
13:     }
14: }
15:
16: class Shape
17: {
18:     public virtual void Draw()
19:     {
20:         Console.WriteLine("Drawing a shape.");
21:     }
22: }
23:
24: class Circle : Shape
25: {
26:     public override void Draw()
27:     {
28:         Console.WriteLine("Drawing a circle.");
29:     }
30: }
31:
32: class Square : Shape
33: {
34:     public override void Draw()
35:     {
36:         Console.WriteLine("Drawing a square.");
37:     }
38: }
39:
40: class Program
41: {
42:     static void Main(string[] args)
43:     {
44:         // Polimorfisme Compile-Time (Overloading)
45:         Calculator calc = new Calculator();
```

```
46:         int sumInt = calc.Add(3, 5); // Memanggil metode Add(int, int)
47:         double sumDouble = calc.Add(3.5, 4.7); // Memanggil metode
Add(double, double)
48:
49:         Console.WriteLine("Sum of integers: " + sumInt);
50:         Console.WriteLine("Sum of doubles: " + sumDouble);
51:
52:         // Polimorfisme Run-Time (Overriding)
53:         Shape[] shapes = { new Shape(), new Circle(), new Square() };
54:
55:         foreach (var shape in shapes)
56:         {
57:             shape.Draw(); // Memanggil Draw() sesuai dengan jenis objek
58:         }
59:     }
60: }
```

7. Tekan tombol `Ctrl + F5` atau pilih "Debug" > "Start Without Debugging" untuk menjalankan program.
8. Jendela konsol akan muncul dan menampilkan output dari program.

f. Hasil Dan Analisis Data

Hasil dan analisis data dilakukan melalui beberapa langkah:

1. Screenshot luaran dari program.
2. Jelaskan setiap baris program.
3. Analisis dan jelaskan bagaimana konsep inheritance diterapkan pada contoh program diatas.

g. Kesimpulan

Simpulkan apa keuntungan dan kerugian dalam penggunaan konsep polymorphism.

h. Tugas Mandiri

Rancang dan buatlah program yang menerapkan konsep polymorphism. Laporan tugas praktikum yang terdiri dari:

1. Class Diagram
2. Kode Program
3. Luaran Program
4. Hasil Analisis

i. Daftar Pustaka

Farrel, J. 2017. Microsoft Visual C#: An Introduction to Object Oriented Programming.Cengage Learning.

j. Hasil Penilaian

Identitas Penilai (Asisten Praktikum)

NIM	
Nama	

Penilaian

No	Item	Skor
1	Pre-test (maks 5)	
2	Hasil analisis data (maks 30)	
3	Penarikan kesimpulan (maks 25)	
4	Tugas Mandiri (maks 15)	
5	Post-test (mask 15)	
6	Sikap (maks 10)	
	Total Nilai	

Jember, <tgl> <bulan> <tahun>

<Nama Lengkap Penilai>
NIM: xxxxxxxxxxxxxxx

6. ENCAPSULATION

a. Capaian Pembelajaran

Kode CPL	:	CPL 07
Nama CPL	:	Mampu mendesain konsep pemrograman dan algoritma untuk menyelesaikan permasalahan menggunakan perangkat lunak dan sistem komputer
Kode CPMK	:	CPMK 01
Nama CPMK	:	mengimplementasikan desain algoritma untuk menyelesaikan permasalahan tertentu
Nama Sub-CPMK	:	mahasiswa mampu mendesain algoritma rekursif untuk menyelesaikan persoalan algoritma tertentu

b. Tujuan Percobaan

Praktikum mengenai encapsulation dalam pemrograman berorientasi objek bertujuan untuk:

Mempelajari Konsep Dasar OOP

Encapsulation adalah salah satu pilar utama dari Pemrograman Berorientasi Objek (OOP). Tujuan praktikum adalah untuk memberikan pemahaman yang mendalam tentang konsep dasar ini kepada para mahasiswa.

Memahami Pentingnya Pembungkusan Data

Praktikum ini membantu para mahasiswa memahami pentingnya menyembunyikan detail implementasi (data dan metode) dalam sebuah kelas. Ini membantu mencegah akses langsung ke data dari luar kelas dan memastikan bahwa data hanya dapat dimanipulasi melalui metode yang ditentukan.

Memperkenalkan Penggunaan Access Modifiers

Encapsulation berkaitan erat dengan penggunaan access modifiers (seperti public, private, protected). Praktikum membantu mahasiswa memahami

bagaimana menerapkan access modifiers untuk mengatur akses terhadap properti dan metode dalam kelas.

Menjaga Keamanan dan Konsistensi Data

Melalui praktikum ini, mahasiswa belajar bagaimana menggunakan konsep encapsulation untuk menjaga keamanan dan konsistensi data. Dengan mengizinkan akses terkontrol ke data dan memaksa aturan tertentu melalui metode, praktikum ini membantu mencegah perubahan yang tidak sah pada data.

Meningkatkan Pemeliharaan dan Fleksibilitas Kode

Encapsulation membantu dalam pemeliharaan kode dengan menyembunyikan detail implementasi. Ini memungkinkan perubahan pada bagian internal kelas tanpa mempengaruhi bagian lain dari program, yang pada gilirannya meningkatkan fleksibilitas kode.

Memperdalam Pemahaman Desain dan Struktur Program

Praktikum ini memperdalam pemahaman tentang desain yang baik dalam Pemrograman Berorientasi Objek. Ini membantu mahasiswa untuk merancang program dengan struktur yang terorganisir, memisahkan dan melindungi data dari eksternal, dan membuat kelas yang lebih mudah dipahami dan dipelihara.

c. Tinjauan Pustaka

Encapsulation adalah salah satu konsep utama dalam pemrograman berorientasi objek (OOP) yang melibatkan pembungkusan atau penyembunyian data bersama dengan metode yang beroperasi pada data tersebut. Hal ini memungkinkan penyembunyian detail implementasi di dalam suatu kelas, sementara hanya memperlihatkan metode publik yang dapat digunakan oleh pengguna kelas tersebut.

Komponen-komponen Utama Encapsulation:

1. Data dan Metode: Encapsulation melibatkan pengelompokan data (atribut) dan metode (fungsi atau perilaku) yang beroperasi pada data ke dalam satu unit tunggal, yaitu kelas.
2. Access Modifiers: Untuk menerapkan encapsulation, penggunaan access modifiers seperti `public`, `private`, dan `protected` sangat penting. Ini memungkinkan kontrol akses terhadap data dan metode dalam kelas.

Manfaat Encapsulation:

1. Keamanan Data: Melalui penggunaan access modifiers, encapsulation membantu dalam menjaga keamanan data dengan membatasi akses langsung dari luar kelas.
2. Pemeliharaan Kode: Dengan menyembunyikan detail implementasi, encapsulation memungkinkan perubahan pada bagian internal kelas tanpa mempengaruhi bagian lain dari program, meningkatkan pemeliharaan kode.
3. Reusabilitas: Dengan menyembunyikan detail implementasi, kelas yang dienkapsulasi dapat digunakan kembali di berbagai bagian program tanpa perlu mengetahui detail bagaimana implementasi tersebut dilakukan.

Contoh Encapsulation:

```
class Employee {
    private string name; // Data dienkapsulasi
    public void SetName(string newName) { // Metode untuk mengatur data
        name = newName;
    }
    public string GetName() { // Metode untuk mengambil data
        return name;
    }
}
```

Dalam contoh ini, atribut `name` dienkapsulasi dan hanya dapat diakses atau diubah melalui metode `SetName` dan `GetName`, yang mengontrol akses ke

data. Ini adalah contoh sederhana dari konsep encapsulation dalam pemrograman berorientasi objek.

d. Alat Dan Bahan

Alat yang dibutuhkan:

1. Komputer / Laptop
2. Program Visual Studio

e. Prosedur Kerja

Prosedur kerja dilakukan melalui langkah-langkah berikut:

1. Buka Visual Studio.
2. Pilih "Create a new project" atau "File" > "New" > "Project".
3. Pilih "Console App (.NET Core)" atau "Console App (.NET Framework)".
4. Di jendela solusi (Solution Explorer), klik kanan pada nama proyek.
5. Pilih "Add" > "Class" untuk membuat file kelas baru.
6. Tuliskan kode program berikut

```
1  using System;
2
3  public class Engine
4  {
5      protected string type;
6
7      public Engine(string engineType)
8      {
9          type = engineType;
10     }
11
12     public string GetType()
13     {
14         return type;
15     }
16 }
17
18 class Car
19 {
```

```
20    private string make;
21    private string model;
22    private int year;
23    protected Engine carEngine;
24
25    public Car(string carMake, string carModel, int carYear, Engine engine)
26    {
27        make = carMake;
28        model = carModel;
29        year = carYear;
30        carEngine = engine;
31    }
32
33    public string GetMake()
34    {
35        return make;
36    }
37
38    public string GetModel()
39    {
40        return model;
41    }
42
43    public int GetYear()
44    {
45        return year;
46    }
47
48    public Engine GetEngine()
49    {
50        return carEngine;
51    }
52 }
53
54 class Program
55 {
56    static void Main(string[] args)
57    {
58        Engine v6Engine = new Engine("V6");
59        Car myCar = new Car("Toyota", "Corolla", 2020, v6Engine);
60
61        Console.WriteLine("Car Make: " + myCar.GetMake());
62        Console.WriteLine("Car Model: " + myCar.GetModel());
63        Console.WriteLine("Car Year: " + myCar.GetYear());
64        Console.WriteLine("Car Engine Type: " + myCar.GetEngine().GetType());
```

```
65     }
66 }
```

7. Tekan tombol `Ctrl + F5` atau pilih "Debug" > "Start Without Debugging" untuk menjalankan program.
8. Jendela konsol akan muncul dan menampilkan output dari program.

f. Hasil Dan Analisis Data

Hasil dan analisis data dilakukan melalui beberapa langkah:

1. Screenshot luaran dari program.
2. Jelaskan setiap baris program.
3. Analisis dan jelaskan bagaimana konsep inheritance diterapkan pada contoh program diatas.

g. Kesimpulan

Simpulkan apa keuntungan dan kerugian dalam penggunaan konsep encapsulation.

h. Tugas Mandiri

Rancang dan buatlah program yang menerapkan konsep encapsulation. Laporan tugas praktikum yang terdiri dari:

1. Class Diagram
2. Kode Program
3. Luaran Program
4. Hasil Analisis

i. Daftar Pustaka

Farrel, J. 2017. Microsoft Visual C#: An Introduction to Object Oriented Programming.Cengage Learning.

j. Hasil Penilaian

Identitas Penilai (Asisten Praktikum)

NIM	
Nama	

Penilaian

No	Item	Skor
1	Pre-test (maks 5)	
2	Hasil analisis data (maks 30)	
3	Penarikan kesimpulan (maks 25)	
4	Tugas Mandiri (maks 15)	
5	Post-test (mask 15)	
6	Sikap (maks 10)	
	Total Nilai	

Jember, <tgl> <bulan> <tahun>

<Nama Lengkap Penilai>
NIM: xxxxxxxxxxxxxxx

7. ABSTRACT CLASS

a. Capaian Pembelajaran

1. Kode CPL : CPL 06
2. Nama CPL : Mampu memvalidasi berbagai teknologi yang digunakan dalam pembuatan suatu program, aplikasi dan atau sistem informasi.
3. Kode CPMK : CPMK 04
4. Nama CPMK : Mampu merancang aplikasi menggunakan paradigma OOP secara penuh
5. Nama Sub-CPMK : mahasiswa mampu mendesain algoritma rekursif untuk menyelesaikan persoalan algoritma tertentu

a. Tujuan Percobaan

Memahami Konsep Dasar Abstract Class

Praktikum ini bertujuan untuk memberikan pemahaman yang kokoh tentang konsep abstract class, yang merupakan salah satu dari prinsip-prinsip dasar dalam pemrograman berorientasi objek.

Menjelaskan Perbedaan Antara Abstract Class dan Interface

Praktikum akan membantu mahasiswa memahami perbedaan antara abstract class dan interface. Ini termasuk kapan dan bagaimana menggunakan abstract class dan kapan lebih baik menggunakan interface.

Mengetahui Implementasi Abstract Methods

Praktikum memberikan kesempatan kepada mahasiswa untuk belajar bagaimana menggunakan abstract methods di dalam abstract class, di mana method tersebut hanya dideklarasikan tanpa implementasi konkret. Mahasiswa akan belajar bahwa kelas turunan yang meng-extend abstract class harus memberikan implementasi untuk method-method tersebut.

Memahami Polimorfisme

Abstract class memungkinkan penggunaan polimorfisme, dan praktikum ini akan membantu mahasiswa dalam memahami bagaimana abstract class mendukung polimorfisme dalam pemrograman berorientasi objek.

Meningkatkan Pemahaman Desain

Praktikum akan membantu mahasiswa memahami bagaimana abstract class membantu dalam mendesain struktur program dengan memberikan kerangka kerja yang dapat digunakan oleh kelas turunannya.

Menerapkan Konsep Inheritance

Praktikum juga memberikan pemahaman tentang bagaimana abstract class dapat digunakan sebagai basis untuk inheritance, memfasilitasi reusabilitas dan struktur hirarkis dalam kode.

Mempraktikkan Kasus Penggunaan

Praktikum akan menyediakan kasus penggunaan konkret di mana penggunaan abstract class sangat bermanfaat, memperdalam pemahaman siswa tentang kapan dan bagaimana menggunakan konsep ini dalam pengembangan perangkat lunak.

b. Tinjauan Pustaka

Abstract class adalah kelas yang tidak dapat diinstansiasi secara langsung, yang berfungsi sebagai kerangka dasar untuk kelas turunannya. Abstract class dapat memiliki metode yang memiliki implementasi konkrit, tetapi juga dapat memiliki metode abstrak yang hanya dideklarasikan tanpa implementasi.

Poin Utama Abstract Class:

1. Tidak Dapat Diinstansiasi Langsung: Abstract class tidak dapat dijadikan objek langsung. Itu berperan sebagai kerangka atau blueprint untuk kelas turunannya.

2. Metode Abstrak dan Konkrit: Abstract class dapat memiliki metode dengan implementasi konkrit, serta metode abstrak yang hanya dideklarasikan tanpa implementasi. Metode abstrak harus diimplementasikan oleh kelas turunannya.
3. Mendukung Inheritance: Abstract class memungkinkan kelas turunannya untuk mewarisi properti dan metode dari abstract class, serta mengimplementasikan metode abstrak yang ada.

Manfaat Abstract Class:

- Menyediakan Struktur Kerangka: Abstract class membantu dalam menyediakan kerangka kerja atau blueprint yang dapat digunakan oleh kelas turunannya.
- Penerapan Polimorfisme: Abstract class mendukung polimorfisme, memungkinkan kelas turunannya untuk mengganti metode abstrak sesuai dengan kebutuhan mereka.
- Pembatasan Fungsionalitas: Dengan metode abstrak, abstract class membatasi fungsionalitas yang perlu diimplementasikan oleh kelas turunannya.

Contoh Abstract Class:

```
public abstract class Shape {
    public abstract double CalculateArea(); // Metode abstrak

    public void PrintDetails() {
        Console.WriteLine("Details of the shape");
    }
}

public class Circle : Shape {
    public override double CalculateArea() {
        // Implementasi perhitungan luas lingkaran
    }
}

public class Square : Shape {
    public override double CalculateArea() {
```

```
        // Implementasi perhitungan luas persegi
    }
}
```

Dalam contoh ini, `Shape` adalah abstract class dengan metode abstrak `CalculateArea()`. Kelas `Circle` dan `Square` mewarisi abstract class `Shape` dan memberikan implementasi untuk metode abstrak tersebut sesuai dengan kebutuhan mereka.

d. Alat Dan Bahan

Alat yang dibutuhkan:

1. Komputer / Laptop
2. Program Visual Studio

d. Prosedur Kerja

Prosedur kerja dilakukan melalui langkah-langkah berikut:

3. Buka Visual Studio.
4. Pilih "Create a new project" atau "File" > "New" > "Project".
5. Pilih "Console App (.NET Core)" atau "Console App (.NET Framework)".
6. Di jendela solusi (Solution Explorer), klik kanan pada nama proyek.
7. Pilih "Add" > "Class" untuk membuat file kelas baru.
8. Tuliskan kode program berikut

```
1  using System;
2
3  // Abstract class
4  public abstract class Vehicle
5  {
6      protected string make;
7      protected string model;
8
9      public Vehicle(string vehicleMake, string vehicleModel)
10     {
11         make = vehicleMake;
12         model = vehicleModel;
```

```
13      }
14
15      // Abstract method
16      public abstract void Start();
17      public abstract void Stop();
18  }
19
20  // Concrete class 1
21  public class Car : Vehicle
22  {
23      public Car(string carMake, string carModel) : base(carMake, carModel)
24      {
25          // Specific constructor logic
26      }
27
28      public override void Start()
29      {
30          Console.WriteLine($"Starting the {make} {model} car");
31      }
32
33      public override void Stop()
34      {
35          Console.WriteLine($"Stopping the {make} {model} car");
36      }
37  }
38
39  // Concrete class 2
40  public class Motorcycle : Vehicle
41  {
42      public Motorcycle(string bikeMake, string bikeModel) : base(bikeMake,
bikeModel)
43      {
44          // Specific constructor logic
45      }
46
47      public override void Start()
48      {
49          Console.WriteLine($"Starting the {make} {model} motorcycle");
50      }
51
52      public override void Stop()
53      {
54          Console.WriteLine($"Stopping the {make} {model} motorcycle");
55      }
56  }
```

```
57
58  // Entry point class
59  class Program
60  {
61      static void Main(string[] args)
62      {
63          Car myCar = new Car("Toyota", "Corolla");
64          myCar.Start();
65          myCar.Stop();
66
67          Motorcycle myMotorcycle = new Motorcycle("Honda", "CBR");
68          myMotorcycle.Start();
69          myMotorcycle.Stop();
70      }
71  }
```

9. Tekan tombol `Ctrl + F5` atau pilih "Debug" > "Start Without Debugging" untuk menjalankan program.
10. Jendela konsol akan muncul dan menampilkan output dari program.

d. Hasil Dan Analisis Data

Hasil dan analisis data dilakukan melalui beberapa langkah:

1. Screenshot luaran dari program.
2. Jelaskan setiap baris program.
3. Analisis dan jelaskan bagaimana konsep inheritance diterapkan pada contoh program diatas.

e. Kesimpulan

Simpulkan apa keuntungan dan kerugian dalam penggunaan konsep encapsulation.

f. Tugas Mandiri

Rancang dan buatlah program yang menerapkan konsep kelas abstract. Laporan tugas praktikum yang terdiri dari:

1. Class Diagram
2. Kode Program

3. Luaran Program
4. Hasil Analisis

g. Daftar Pustaka

Farrel, J. 2017. Microsoft Visual C#: An Introduction to Object Oriented Programming.Cengage Learning.

h. Hasil Penilaian

Identitas Penilai (Asisten Praktikum)

NIM	
Nama	

Penilaian

No	Item	Skor
1	Pre-test (maks 5)	
2	Hasil analisis data (maks 30)	
3	Penarikan kesimpulan (maks 25)	
4	Tugas Mandiri (maks 15)	
5	Post-test (mask 15)	
6	Sikap (maks 10)	
	Total Nilai	

Jember, <tgl> <bulan> <tahun>

<Nama Lengkap Penilai>
NIM: xxxxxxxxxxxxxxx