LAPORAN PRAKTIKUM



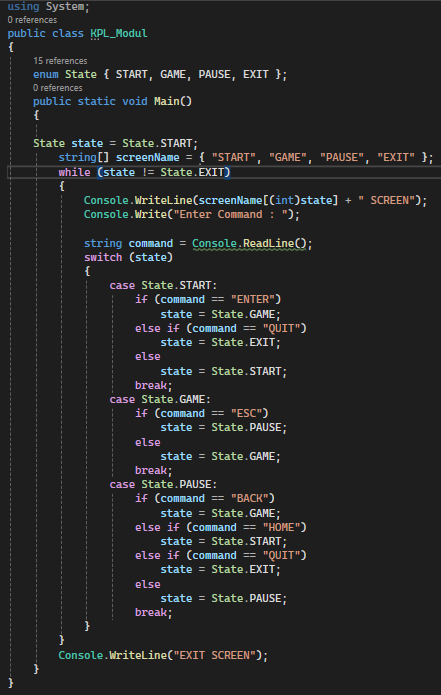
Nama Mahasiswa: Abdul Hakim Al Baihaqy  
Nim: 1201230031

Mata Kuliah Konstruksi Perangkat Lunak   
Semester Genap Tahun Ajaran 2025-2026

Program Studi S1 Rekayasa Perangkat Lunak  
Telkom University   
Direktorat Kampus Surabaya

1. **Modul KPL:**  
   **4.1 Automata Consturction**

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 overlaping 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.



 enum adalah tipe data khusus yang berisi sekumpulan konstanta bernama.

* State di sini merepresentasikan status atau kondisi dari aplikasi.
* Terdapat 4 status dalam State:

1. START : Layar awal
2. GAME : Layar permainan
3. PAUSE : Layar jeda (berhenti sementara)
4. EXIT : Keluar dari aplikasi

* state diatur pada State.START, artinya aplikasi dimulai dari **START SCREEN**.
* screenName adalah array yang menyimpan nama-nama status untuk ditampilkan ke layar.
* Program akan berjalan terus selama state **bukan** State.EXIT.
* Jika state berubah menjadi State.EXIT, maka program akan keluar dari *loop*.
* Program menampilkan nama layar saat ini berdasarkan state menggunakan **casting** (int)state untuk mengakses indeks di screenName.
* Pengguna diminta memasukkan **perintah (command)** menggunakan Console.ReadLine().
* Jika input adalah:

1. "ENTER" → Pindah ke State.GAME
2. "QUIT" → Keluar dari aplikasi (State.EXIT)
3. Perintah lain → Tetap di layar START

* Jika input adalah:

1. "ESC" → Pindah ke State.PAUSE
2. Perintah lain → Tetap di layar GAME

* Jika input adalah:

1. "BACK" → Kembali ke State.GAME
2. "HOME" → Kembali ke State.START
3. "QUIT" → Keluar dari aplikasi (State.EXIT)
4. Perintah lain → Tetap di layar PAUSE

* Ketika status berubah menjadi State.EXIT, *loop* berhenti dan menampilkan pesan **EXIT SCREEN**.

**4.2 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.

A computer screen shot of a program

AI-generated content may be incorrect.

**4.1.2.2 Indexed Access**

Indexed Acces adalah metode table-driven yang mengunakan table lain untuk menyimpan index dari table utama, hal ini bertujuan untuk mengurangi penggunan 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.

A computer screen shot of a program

AI-generated content may be incorrect.

Penjelasan:

* **grade**: Array yang menyimpan huruf nilai (dari terbaik hingga terendah).
* **rangeLimit**: Batas bawah skor untuk mendapatkan nilai tertentu.

1. Skor ≥ 80 → **A**
2. Skor ≥ 70 → **AB**
3. Skor ≥ 65 → **B**
4. Skor ≥ 60 → **BC**
5. Skor ≥ 50 → **C**
6. Skor ≥ 40 → **D**
7. Skor < 40 → **E**

* **maxGradeLevel**: Nilai tertinggi indeks dari array grade (yaitu 6 karena grade.Length - 1).

**2. Jurnal Mandiri Modul 3**

(Link Repository: <https://github.com/Ahab131/Repo_KonstruksiPL_Hakim>)

A. KodePos

Input:

A computer screen shot of a program

AI-generated content may be incorrect.

Output:

A black and white screen with white text

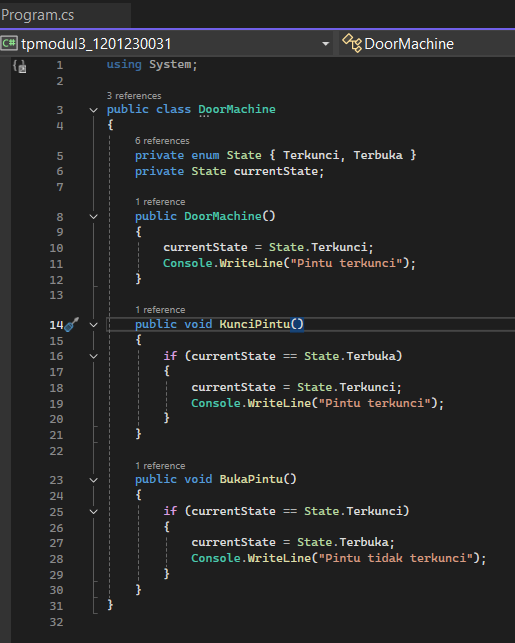
AI-generated content may be incorrect.

Penjelasan:

* using System; → Mengimpor pustaka System yang menyediakan fungsi dasar seperti Console.WriteLine().
* using System.Collections.Generic; → Mengimpor pustaka yang mendukung struktur data generic collection seperti Dictionary.
* class KodePos → Mendefinisikan kelas bernama KodePos yang berisi logika untuk mencari kode pos.
* private → Akses modifier yang berarti hanya bisa diakses di dalam kelas KodePos saja.
* static → Properti ini statis, artinya digunakan bersama oleh seluruh objek dari kelas ini.
* Dictionary<string, string> → Struktur data yang menyimpan key-value pair (kunci-nilai).
* Key: Nama kelurahan (tipe string).
* Value: Kode pos dari kelurahan tersebut (tipe string).
* Isi Dictionary: Menyimpan pasangan kelurahan dan kode pos di Kota Bandung.
* private → Akses modifier yang berarti hanya bisa diakses di dalam kelas KodePos saja.
* static → Properti ini statis, artinya digunakan bersama oleh seluruh objek dari kelas ini.
* Dictionary<string, string> → Struktur data yang menyimpan key-value pair (kunci-nilai).
* Key: Nama kelurahan (tipe string).
* Value: Kode pos dari kelurahan tersebut (tipe string).
* Isi Dictionary: Menyimpan pasangan kelurahan dan kode pos di Kota Bandung.
* public → Bisa diakses dari luar kelas.
* static → Bisa dipanggil tanpa membuat objek dari kelas KodePos.
* string GetKodePos(string kelurahan) → Metode ini menerima nama kelurahan sebagai input dan mengembalikan kode pos dalam bentuk string.
* kodePosTable.ContainsKey(kelurahan) → Mengecek apakah kelurahan yang dimasukkan ada di dalam Dictionary.
* Jika ada → Mengembalikan kode pos dari kelurahan tersebut.
* Jika tidak ada → Mengembalikan string "Kode Pos tidak ditemukan".
* static void Main(string[] args) → Method utama (entry point) yang akan dieksekusi pertama kali saat program dijalankan.
* Console.WriteLine() → Mencetak teks ke konsol.
* KodePos.GetKodePos("Batununggal") → Memanggil metode GetKodePos() di kelas KodePos dengan parameter "Batununggal".

B. DoorMachine

Input:



A screen shot of a computer program

AI-generated content may be incorrect.

Output:

A screenshot of a computer

AI-generated content may be incorrect.

Penjelasan:

* private → Hanya bisa diakses di dalam kelas DoorMachine.
* enum → Digunakan untuk mendefinisikan kumpulan nilai tetap (konstanta) yang merepresentasikan status/kondisi.
* State → Nama dari enumerasi ini.
* Terkunci → Status pintu dalam keadaan terkunci.
* Terbuka → Status pintu dalam keadaan terbuka.
* private → Hanya dapat diakses dalam kelas ini.
* State → Tipe data dari properti ini adalah State (menggunakan enum yang telah didefinisikan sebelumnya).
* currentState → Variabel ini menyimpan status pintu saat ini.
* public → Bisa diakses dari luar kelas.
* void → Tidak mengembalikan nilai.
* KunciPintu() → Method ini digunakan untuk mengunci pintu.
* if (currentState == State.Terbuka) → Hanya mengunci pintu jika saat ini pintu dalam keadaan terbuka.
* Jika pintu terbuka, status berubah menjadi terkunci dan mencetak pesan "Pintu terkunci".
* Jika pintu sudah terkunci, tidak melakukan apa-apa.
* public → Bisa diakses dari luar kelas.
* void → Tidak mengembalikan nilai.
* BukaPintu() → Method ini digunakan untuk membuka pintu.
* if (currentState == State.Terkunci) → Hanya membuka pintu jika saat ini pintu dalam keadaan terkunci.
* Jika pintu terkunci, status berubah menjadi terbuka dan mencetak pesan "Pintu tidak terkunci".
* Jika pintu sudah terbuka, tidak melakukan apa-apa.

1. **Jurnal Mandiri Modul 4**

(LinkRepository:<https://github.com/Ahab131/1201230031_KonstruksiPL2_Week3/tree/main>)

1. KodeBuah:

Input:

A computer screen shot of a program

AI-generated content may be incorrect.

Output:

A screenshot of a computer

AI-generated content may be incorrect.

Penjelasan:

* private: Hanya dapat diakses di dalam kelas KodeBuah.
* static: Bersifat statis, artinya hanya ada satu salinan untuk seluruh program (berlaku di tingkat kelas, bukan objek).
* readonly: Nilai dari variabel hanya bisa diinisialisasi sekali (biasanya di saat deklarasi atau di konstruktor).
* Dictionary<string, string>: Struktur data berbentuk pasangan key-value. Dalam hal ini:
* string pertama sebagai key (nama buah).
* string kedua sebagai value (kode buah).
* public: Bisa diakses dari luar kelas.
* static: Dapat diakses tanpa perlu membuat objek dari kelas KodeBuah.
* string: Tipe data yang dikembalikan oleh method ini.
* GetKodeBuah: Nama method untuk mendapatkan kode buah.
* string namaBuah: Parameter input berupa nama buah.
* Console.WriteLine(): Digunakan untuk mencetak teks ke layar konsol.
* Memanggil GetKodeBuah() untuk mencari kode berdasarkan nama buah:

1. Apel → "A00"
2. Durian → "L00"
3. Mangga (tidak ada di dictionary) → "Kode tidak ditemukan"

* TryGetValue(): Mencari nilai berdasarkan key.

1. Jika ditemukan, nilai disimpan di variabel kode.
2. Jika tidak ditemukan, variabel kode diabaikan.

* Operator ? : (Ternary Operator):

1. Jika buah ditemukan → Kembalikan kode buah.
2. Jika tidak ditemukan → Kembalikan "Kode tidak ditemukan".
3. PosisiKarakterGame:

Input:

A screen shot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A computer screen shot of a program

AI-generated content may be incorrect.

Output:

A screenshot of a computer screen

AI-generated content may be incorrect.

Penjelasan:

* Fungsi: Membuat tipe data khusus (enum) yang mewakili state atau posisi karakter dalam game.
* Isi Enum:

1. Berdiri → State awal karakter.
2. Jongkok → Posisi jongkok.
3. Tengkurap → Posisi tengkurap (lebih rendah dari jongkok).
4. Terbang → Posisi melayang di udara.

* posisiSaatIni: Menyimpan posisi karakter saat ini.
* nim: Menyimpan NIM untuk menangani logika tambahan.
* Konstruktor:

1. Mengatur posisi awal karakter menjadi Berdiri.
2. Menerima NIM sebagai parameter dan menyimpannya di atribut.

* Fungsi:

1. Menangkap aksi ketika Tombol S ditekan (bergerak ke bawah).
2. Jika NIM % 3 == 0, akan mencetak pesan khusus:

"tombol arah bawah ditekan".

* Transisi State:

1. Berdiri → Jongkok
2. Jongkok → Tengkurap
3. Terbang → Jongkok

* Fungsi:

1. Menangkap aksi ketika Tombol W ditekan (bergerak ke atas).
2. Jika NIM % 3 == 0, mencetak: "tombola rah atas ditekan".

* Transisi State:

1. Jongkok → Berdiri
2. Tengkurap → Jongkok
3. Berdiri → Terbang

* Fungsi public void TekanTombolX():

Menangkap aksi ketika Tombol X ditekan.

Hanya berfungsi dari Jongkok → Terbang.

* Fungsi private void UbahPosisi(Posisi posisiBaru):

1. Mengatur logika saat posisi berubah.
2. Menambahkan output khusus sesuai hasil NIM % 3:
3. NIM % 3 == 2:
   * + - Dari Terbang ke Jongkok: Output "posisi landing".
       - Dari Berdiri ke Terbang: Output "posisi take off".
4. NIM % 3 == 1:
   * + - Berdiri: Output "posisi standby".
       - Tengkurap: Output "posisi istirahat".

* Fungsi public void SimulasiPerubahan(): Mensimulasikan semua transisi dari diagram state secara berurutan.
* Urutan Simulasi:

1. Berdiri → Jongkok
2. Jongkok → Tengkurap
3. Tengkurap → Jongkok
4. Jongkok → Berdiri
5. Berdiri → Terbang
6. Terbang → Jongkok
7. Jongkok → Terbang