PROYEK UJIAN AKHIR SEMESTER GENAP STRUKTUR DATA DAN PRAKTIKUM STRUKTUR DATA APLIKASI SIMULASI GAME MONOPOLI MENGGUNAKAN BAHASA PEMROGRAMAN C++

Dosen Pengampu

Mohamad Bayu Wibisono, S.Kom., MM[(1)]

Disusun oleh:

Kelompok 7

Alghifari Rahbanni Ramadhan	(2410512180)
Dzulhas Syahari Muthahari	(2410512194)
Muhammad Fawwaz Ferdian Satriadi	(2410512198)
Muhammad Bagas Ramadhan	(2410512199)
Bagas Malik Ibrahim	(2410512201)

Jalan RS. Fatmawati Raya, Pd. Labu, Kec. Cilandak, Kota Jakarta Selatan, Daerah Khusus Ibukota Jakarta 12450, Sistem Informasi 2024/2025

KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya, penulis dapat menyelesaikan penulisan jurnal ini dengan judul **"Aplikasi Simulasi Game Monopoli Menggunakan Bahasa Pemrograman C++"**.

Monopoli merupakan salah satu permainan papan klasik yang populer di seluruh dunia. Permainan ini tidak hanya menghibur, tetapi juga mengajarkan strategi ekonomi, manajemen keuangan, dan pengambilan keputusan. Dalam era digital saat ini, pengembangan simulasi permainan Monopoli menggunakan bahasa pemrograman C++ menjadi sangat relevan untuk memahami konsep pemrograman berorientasi objek, pengelolaan memori dinamis, dan penerapan struktur data serta algoritma dalam konteks yang interaktif dan menyenangkan.

Tujuan dari penulisan jurnal ini adalah untuk memaparkan proses perancangan dan implementasi simulasi game Monopoli menggunakan bahasa C++, termasuk analisis aturan permainan, desain kelas dan objek, serta tantangan teknis yang dihadapi selama pengembangan. Diharapkan, jurnal ini dapat menjadi referensi yang bermanfaat bagi mahasiswa, pengembang pemula, maupun peneliti yang tertarik dalam pengembangan game berbasis console maupun sistem pemrograman berorientasi objek.

Penulis menyadari bahwa jurnal ini masih memiliki berbagai keterbatasan. Oleh karena itu, saran dan kritik yang membangun dari para pembaca sangat diharapkan guna perbaikan dan pengembangan lebih lanjut di masa yang akan datang.

Akhir kata, penulis mengucapkan terima kasih kepada semua pihak yang telah memberikan dukungan dalam penyusunan jurnal ini, termasuk dosen pembimbing, rekan-rekan sejawat, serta berbagai sumber referensi yang telah menjadi inspirasi dan sumber pengetahuan. Semoga jurnal ini dapat memberikan kontribusi nyata bagi perkembangan ilmu komputer, khususnya dalam bidang pemrograman game dan teknologi perangkat lunak.

Jakarta, 20 Juni 2025

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1 Latar Belakang

Permainan Monopoli telah menjadi salah satu permainan papan paling ikonik di dunia sejak pertama kali diperkenalkan pada awal abad ke-20. Permainan ini tidak hanya bersifat hiburan, tetapi juga mengajarkan pemain tentang manajemen keuangan, strategi investasi, dan pengambilan keputusan dalam kondisi yang dinamis. Dalam dunia pemrograman, Monopoli dapat menjadi studi kasus yang menarik karena melibatkan berbagai elemen kompleks seperti manajemen aset, transaksi keuangan, pergerakan pemain, dan interaksi antar objek.

Dalam konteks mata kuliah Struktur Data, permainan Monopoli menawarkan peluang untuk mengimplementasikan berbagai struktur data dan algoritma secara nyata. Beberapa contoh penerapannya meliputi:

- Linked List atau Circular Linked List untuk merepresentasikan papan permainan yang bersifat sirkuler.
- Graph untuk memodelkan hubungan antar properti (seperti kelompok warna) dan menentukan harga sewa.
- Queue atau Priority Queue untuk mengatur giliran pemain berdasarkan urutan atau prioritas tertentu.
- Tree (misalnya Binary Search Tree) untuk mengelola daftar properti yang dimiliki pemain agar pencarian dan penyortiran lebih efisien.
- Hash Table untuk menyimpan data pemain, properti, atau kartu-kartu dalam permainan dengan akses cepat.
- Array untuk representasi papan permainan

Selain itu, pengembangan game Monopoli menggunakan C++ memberikan tantangan sekaligus keunggulan karena bahasa ini mendukung pemrograman berorientasi objek (OOP) yang memudahkan dalam memodelkan entitas seperti Pemain, Properti, Dadu, dan Papan. C++ juga memungkinkan optimasi memori dan kecepatan eksekusi, yang penting dalam aplikasi berbasis simulasi.

Proyek ini tidak hanya bertujuan untuk memenuhi tugas Ujian Akhir Semester, tetapi juga sebagai sarana untuk mengasah kemampuan pemrograman, pemahaman struktur data, dan kemampuan problem-solving dalam skenario dunia nyata.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, proyek simulasi game Monopoli ini akan menjawab beberapa pertanyaan mendasar berikut:

Struktur Data & Algoritma:

- Bagaimana merancang struktur data yang optimal untuk merepresentasikan papan Monopoli, properti, pemain, dan transaksi keuangan?
- Algoritma apa yang paling efisien untuk menangani pergerakan pemain, penghitungan sewa, dan mekanisme pelelangan?

Implementasi Mekanisme Game:

- Bagaimana mengimplementasikan fitur-fitur utama Monopoli seperti pembelian properti, pembangunan rumah/hotel, penjara, dan kartu Kesempatan/Dana Umum?
- Bagaimana mensimulasikan interaksi antar pemain (AI atau manusia) dengan kompleksitas yang realistis?

Analisis Kinerja:

- Bagaimana mengevaluasi efisiensi struktur data dan algoritma yang digunakan dalam hal waktu eksekusi dan penggunaan memori?
- Apakah solusi yang diterapkan dapat diskalakan untuk papan yang lebih besar atau aturan yang lebih kompleks?

Antarmuka Pengguna:

- Bagaimana merancang antarmuka yang user-friendly, baik berbasis teks (CLI) atau grafis sederhana, untuk memudahkan pengujian?

1.3 Tujuan

Proyek ini dirancang dengan beberapa tujuan utama:

Implementasi Struktur Data:

- Menerapkan berbagai struktur data (linked list, graph, hash table, dll.) untuk memodelkan komponen-komponen dalam Monopoli.
- Menguji efisiensi struktur data yang berbeda dalam menangani operasi seperti pencarian properti, pembaruan kepemilikan, dan penghitungan sewa.

Pengembangan Game Dasar:
Membangun simulasi Monopoli dengan fitur inti seperti
- Pergerakan pemain berdasarkan dadu.
- Pembelian dan manajemen properti.
- Mekanisme sewa, penjara, dan kartu acak.
- Sistem keuangan (uang, hipotek, bangunan).

Pemrograman Berorientasi Objek:
- Memanfaatkan konsep OOP dalam C++ (class, inheritance, polymorphism) untuk mendesain entitas seperti Player, Property, dan Board.

Evaluasi Kinerja:
- Menganalisis kompleksitas waktu dan ruang dari algoritma yang digunakan.
- Membandingkan solusi alternatif (misalnya: array vs linked list untuk papan permainan).

Dokumentasi & Pengujian:
- Menyediakan dokumentasi yang jelas tentang desain dan implementasi.
- Melakukan pengujian fungsional untuk memastikan semua fitur bekerja sesuai aturan Monopoli.

1.4 Manfaat

Proyek ini memberikan manfaat bagi berbagai pihak, antara lain:

Bagi Mahasiswa:
- Memperdalam pemahaman tentang struktur data dan algoritma melalui studi kasus nyata.
- Melatih kemampuan pemrograman C++, debugging, dan pengembangan perangkat lunak.
- Meningkatkan keterampilan desain sistem dan problem-solving.

Bagi Pengajar/Dosen:
- Memberikan contoh aplikatif struktur data yang dapat digunakan sebagai bahan ajar.
- Memperkaya referensi tugas atau proyek praktikum untuk mata kuliah terkait.

Bagi Pengembang Game:

- Menyediakan kerangka dasar (framework) untuk pengembangan game board lainnya seperti Catan atau ular tangga.
- Demonstrasi penerapan AI sederhana untuk pemain komputer (bot).

Bagi Komunitas Pemrograman:

- Proyek open-source ini dapat dikembangkan lebih lanjut dengan fitur tambahan seperti GUI, multiplayer online, atau aturan kustom.
- Kontribusi untuk repositori proyek edukatif berbasis C++ dan struktur data.

Dengan demikian, proyek ini tidak hanya bernilai akademis tetapi juga memiliki dampak praktis dalam pengembangan perangkat lunak dan pendidikan teknologi.

BAB II

METODOLOGI

2.1 Identifikasi Masalah

Permainan Monopoli, meskipun terlihat sederhana, melibatkan berbagai komponen kompleks yang saling berinteraksi secara dinamis. Dalam konteks pemrograman dan struktur data, beberapa permasalahan utama yang diidentifikasi meliputi:

- **Representasi papan permainan** yang bersifat sirkuler, yang memerlukan struktur data yang sesuai seperti circular linked list atau array dengan pointer siklik.
- **Manajemen data properti dan pemain**, termasuk transaksi jual-beli, pembayaran sewa, serta status kepemilikan dan pembangunan (rumah/hotel).
- **Sistem giliran dan pergerakan pemain**, yang harus mengikuti aturan permainan serta memperhitungkan efek dari kartu acak atau kondisi tertentu (masuk penjara, bebas sewa, dll).
- **Pengelolaan data dengan efisien**, seperti pencarian properti, pengecekan status pemain, dan pemrosesan kartu dengan waktu yang optimal.

Permasalahan-permasalahan ini menjadi dasar dalam perancangan dan implementasi sistem agar game dapat berjalan stabil, efisien, dan sesuai aturan.

2.2 Perancangan program

Tahap ini bertujuan untuk merancang struktur program dan pemetaan komponen-komponen game ke dalam konsep pemrograman, khususnya berorientasi objek dan penggunaan struktur data.

a. Desain Struktur Data

- Papan permainan: Diimplementasikan menggunakan circular linked list untuk memungkinkan pergerakan pemain secara melingkar tanpa batas.
- Properti: Disimpan sebagai objek dalam linked list dan juga dalam struktur tambahan seperti hash table untuk akses cepat berdasarkan nama atau ID.
- Pemain: Disimpan dalam array atau list, dengan setiap elemen merepresentasikan objek Player yang menyimpan status uang, lokasi, dan daftar properti.

- Kartu acak: Digunakan queue (untuk giliran kartu yang terus dipakai berulang) dan stack (untuk efek yang hanya digunakan sekali).
- Antrian pemain: Diatur menggunakan queue agar giliran berjalan sesuai siklus.

b. Desain Class

- **Player**: menyimpan nama, posisi, uang, daftar properti, dan status penjara.
- **Property**: menyimpan nama, harga beli, harga sewa, status bangunan, dan pemilik.
- **Board**: menyimpan node papan berupa properti, dadu, dan elemen lain.
- **Game**: sebagai pengendali utama logika permainan dan interaksi antar objek.

2.3 Implementasi Program

Proses implementasi dilakukan menggunakan bahasa pemrograman **C++** dengan pendekatan berorientasi objek (Object-Oriented Programming). Beberapa tahap implementasi meliputi:

1. **Inisialisasi Struktur Data**
    - Papan dibuat dalam bentuk circular linked list berisi 40 petak.
    - Properti dan kartu di-load dari file atau di-hardcode untuk penyederhanaan.
2. **Fungsi Utama Game**

Fungsi	Peran
`main()`	Titik awal permainan, inisialisasi pemain dan pemanggilan fungsi utama
`menuUtama()`	**Pusat kontrol game** – menampilkan menu dan mengatur giliran pemain
`giliranPemain()`	**Menjalankan giliran pemain** – termasuk dadu, pindah, beli, bayar, efek petak
`buatPapan()`	**Membuat dan menghubungkan petak papan permainan**

`pemilihanKarakter()`	**Setup karakter pemain** – tiap pemain memilih karakter unik
`cekKebangkrutan()`	**Deteksi pemain bangkrut** – properti dilepas, pemain dinonaktifkan
`pemainAktif()`	**Mengecek jumlah pemain aktif** – jika 1 pemain tersisa, game berakhir
`efekKartuKesempatan()`	Memberikan **efek acak dari petak Kesempatan**
`efekKartuDanaUmum()`	Memberikan **efek acak dari petak Dana Umum**
`tampilkanPapan()`	Menampilkan semua petak di papan dan statusnya
`tampilkanKartuPemain()`	Menampilkan semua properti milik pemain tertentu
`tampilkanStatistik()`	Menampilkan ringkasan akhir permainan (saldo, properti, level)
`getNamaPemilik()`	Mengembalikan string nama pemilik untuk ditampilkan di papan

Tabel 2.3

3. **Interaksi dan Tampilan**
    - Menggunakan CLI (Command Line Interface) berbasis teks untuk interaksi pengguna.
    - Input dan output ditampilkan melalui terminal, dengan opsi menu dan perintah khusus.

2.4 Analisis Data Keakuratan Program

Analisis dilakukan untuk memastikan bahwa sistem bekerja sesuai dengan aturan permainan Monopoli. Beberapa metode yang digunakan:

- **Pengujian Fungsional (Functional Testing):**
    - Menguji setiap fitur (pembelian, pembayaran, penjara, dll.) dengan berbagai skenario input.
    - Pengujian logika giliran dan pergerakan dadu.
- **Validasi Aturan Game:**
    - Memastikan bahwa harga sewa, bangunan, dan efek kartu sesuai aturan.
    - Memastikan bahwa pemilik hanya menerima sewa jika properti belum digadaikan.
    - Menjaga ketepatan alur permainan: giliran, penjara, efek kartu, menang/kalah.
- **Pengujian Kinerja (Performance Testing):**
    - Mengukur waktu respon untuk operasi seperti pindah petak, beli properti, dan perhitungan sewa.
    - Mengamati penggunaan memori, terutama saat menyimpan banyak properti dan pemain.
- **Stres Testing (opsional):**
    - Mencoba game dalam kondisi ekstrem seperti 10+ pemain atau 100+ giliran untuk melihat stabilitas sistem.

BAB III

KAJIAN PUSTAKA

3.1 Game Monopoly

Monopoli adalah permainan papan klasik yang pertama kali diterbitkan secara luas oleh Parker Brothers pada tahun 1935 dan kini diterbitkan oleh Hasbro. Permainan ini bertujuan untuk menguasai seluruh properti di papan permainan dengan cara membeli, menyewa, dan membangun properti hingga pemain lain bangkrut. Monopoli telah mengalami banyak versi lokal dan tematik di seluruh dunia.

Secara umum, Monopoli melibatkan beberapa komponen utama seperti:

- Papan permainan dengan petak-petak properti, tempat khusus, dan instruksi.
- Uang permainan.
- Kartu Kesempatan dan Dana Umum.
- Token pemain, dua buah dadu.
- Rumah dan hotel sebagai bangunan.

Permainan ini tidak hanya bersifat rekreatif tetapi juga mengandung unsur edukatif, seperti manajemen keuangan, pengambilan risiko, dan negosiasi. Dalam konteks pemrograman, Monopoli menjadi studi kasus menarik karena dapat direpresentasikan melalui struktur data dan algoritma yang kompleks.

3.2 Konsep Dasar Game Monopoly

Konsep dasar permainan Monopoli mencakup mekanisme-mekanisme berikut:

1. **Pergerakan Pemain:**
   Pemain melempar dua dadu untuk menentukan jumlah langkah, lalu berpindah sesuai angka total dadu. Papan Monopoli bersifat melingkar, sehingga pergerakan akan terus berputar sepanjang permainan.
2. **Pembelian Properti:**
   Jika pemain mendarat di petak properti yang belum dimiliki, pemain dapat membelinya. Properti tersebut kemudian akan menjadi miliknya dan dapat menghasilkan pendapatan melalui sewa.
3. **Pembayaran Sewa:**
   Jika pemain mendarat di properti milik pemain lain, ia harus membayar sewa. Besaran sewa tergantung pada tingkat pembangunan (rumah/hotel) dan kelompok warna.

4. **Kartu Kesempatan dan Dana Umum:**
   Pemain akan mengambil kartu dari tumpukan saat mendarat di petak yang sesuai. Efeknya bisa menguntungkan atau merugikan (misalnya: mendapat uang, membayar denda, pindah petak, masuk penjara).
5. **Penjara dan Bebas dari Penjara:**
   Pemain bisa dipenjara melalui kartu atau mendarat di petak "Masuk Penjara". Pemain harus menunggu beberapa giliran, membayar denda, atau menggunakan kartu "Bebas dari Penjara".
6. **Bangunan dan Hipotek:**
   Pemain dapat membangun rumah atau hotel jika memiliki satu kelompok warna. Properti juga bisa digadaikan saat kekurangan uang.

Konsep-konsep ini menjadi dasar logika utama dalam implementasi game Monopoli secara digital.

3.3 Dasar Pemrograman dalam Pengembangan Game

Pengembangan game berbasis teks seperti Monopoli dalam C++ memerlukan pemahaman dasar pemrograman dan struktur data yang efisien. Beberapa konsep kunci yang relevan dalam pengembangan game ini adalah:

- **Object-Oriented Programming (OOP):**
  C++ mendukung paradigma OOP yang memungkinkan representasi entitas dalam game (seperti pemain, properti, kartu) sebagai objek. Konsep pewarisan (inheritance), enkapsulasi (encapsulation), dan polimorfisme (polymorphism) digunakan untuk menyederhanakan pengelolaan data dan logika game.
- **Struktur Data:**
  Penggunaan struktur data seperti linked list, array, stack, queue, graph, dan hash table sangat penting untuk merepresentasikan papan permainan, giliran pemain, status properti, dan kartu.
- **Algoritma:**
  Algoritma digunakan untuk menangani proses pergerakan pemain, evaluasi kondisi permainan, serta simulasi keputusan otomatis (untuk AI sederhana).
- **Modularisasi Program:**
  Game dibangun secara modular dengan memisahkan setiap fitur ke dalam fungsi atau class tertentu, sehingga kode lebih mudah dibaca, diuji, dan dikembangkan.
- **Manajemen Memori:**
  Dalam C++, penggunaan pointer dan dynamic memory allocation harus dikelola dengan hati-hati agar program tidak mengalami memory leak atau crash.

3.4 Definisi Aplikasi

Aplikasi dalam konteks ini adalah sebuah program perangkat lunak yang dirancang untuk menyimulasikan permainan Monopoli secara digital berbasis teks (CLI/Command Line Interface), dengan berbagai fitur utama permainan asli yang dapat dijalankan di lingkungan sistem operasi komputer.

Aplikasi ini berfungsi sebagai media interaktif yang memungkinkan pengguna menjalankan permainan Monopoli melawan pemain lain atau melawan komputer (AI). Selain sebagai sarana hiburan, aplikasi ini juga menjadi media pembelajaran untuk mengaplikasikan konsep-konsep struktur data dan algoritma secara nyata.

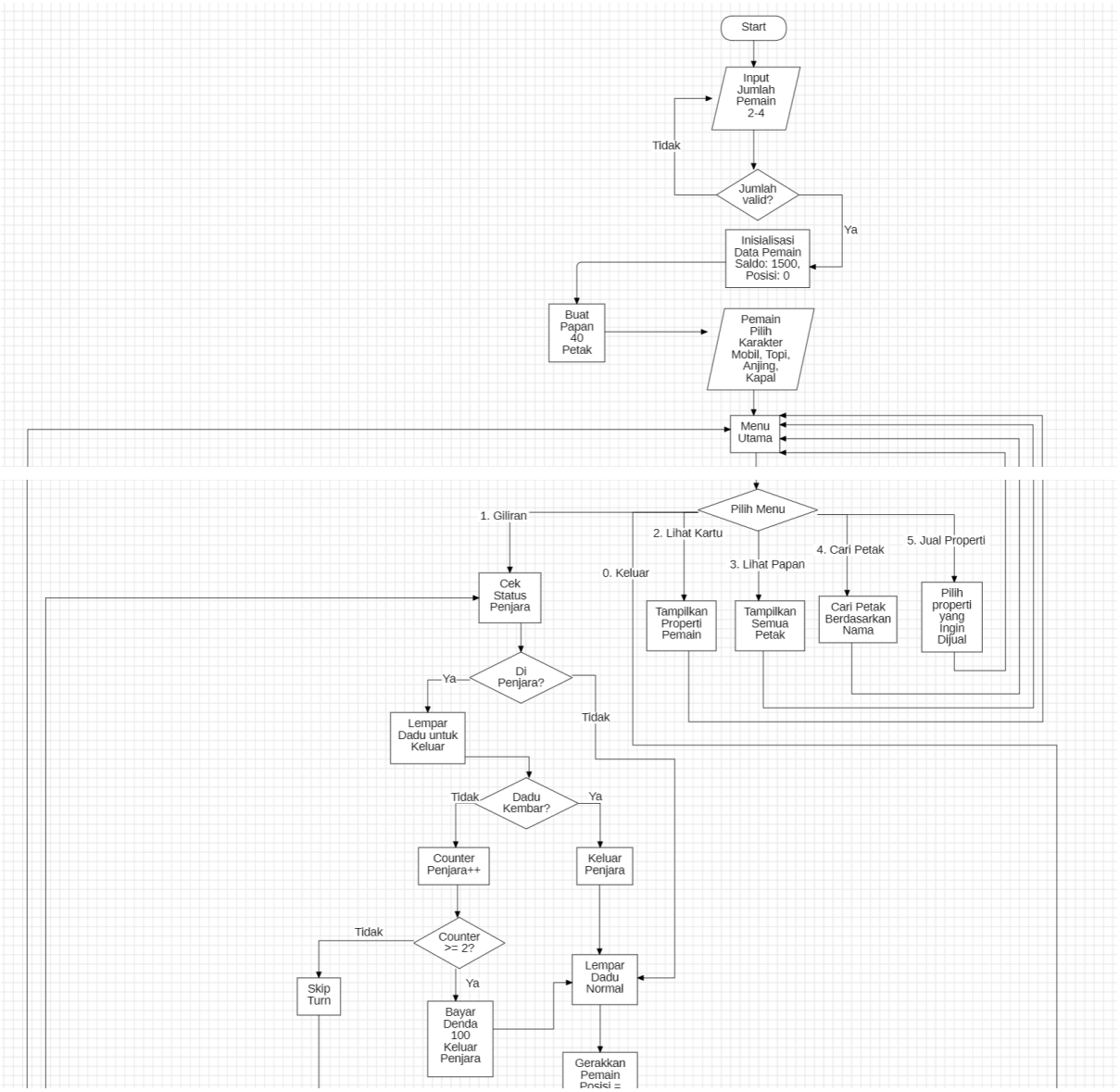
Beberapa ciri khas dari aplikasi ini antara lain:

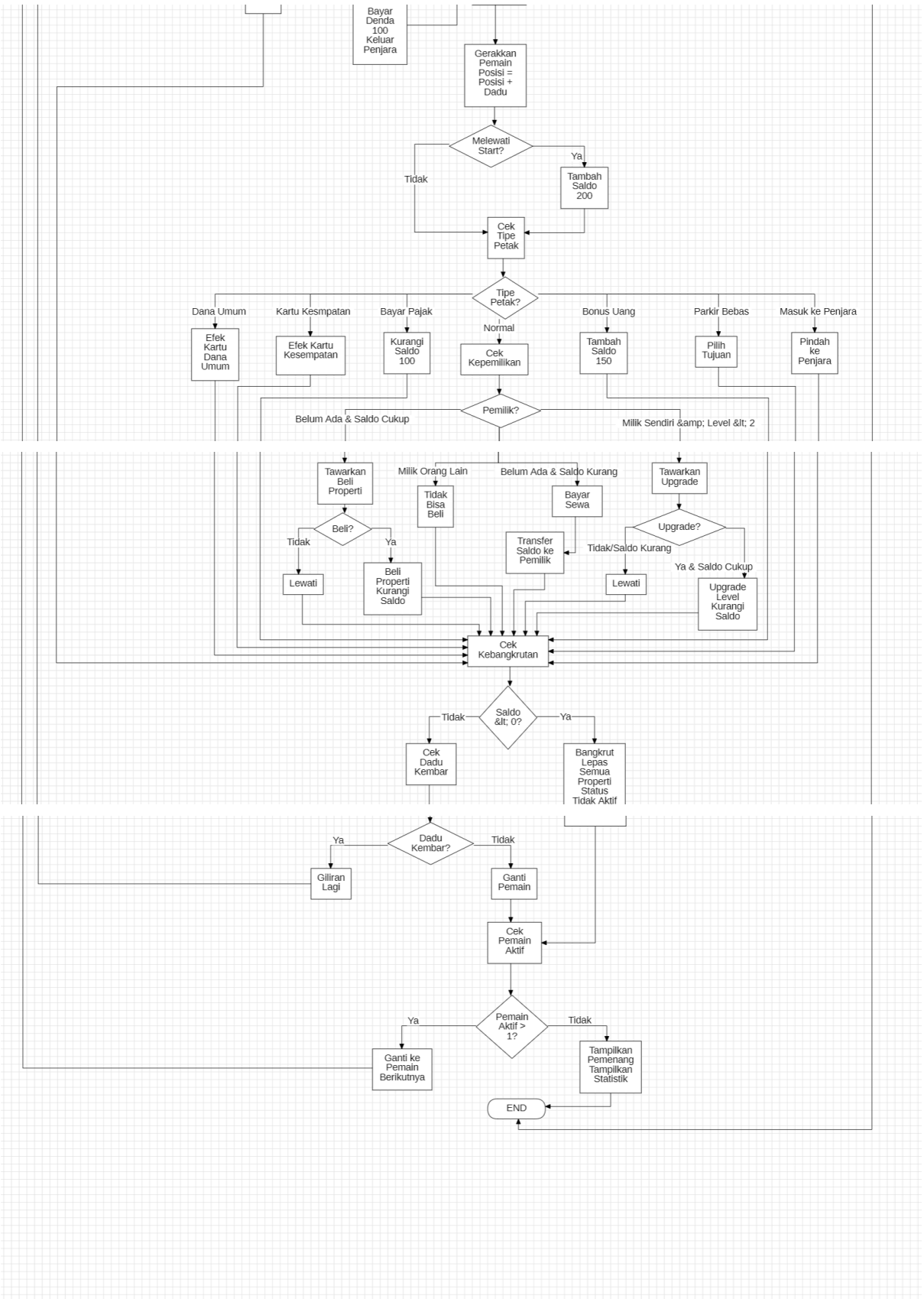
- Dibangun menggunakan bahasa C++.
- Berbasis teks (CLI), bukan GUI.
- Menggunakan prinsip OOP dan struktur data kompleks.
- Dapat dikembangkan lebih lanjut menjadi sistem multiplayer atau berbasis grafis.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Flowchart




Bayar Denda 100 Keluar Penjara
Gerakkan Pemain Posisi = Posisi + Dadu
Melewati Start?
Ya
Tidak
Tambah Saldo 200
Cek Tipe Petak
Tipe Petak?
Dana Umum
Kartu Kesmpatan
Bayar Pajak
Normal
Bonus Uang
Parkir Bebas
Masuk ke Penjara
Efek Kartu Dana Umum
Efek Kartu Kesempatan
Kurangi Saldo 100
Cek Kepemilikan
Tambah Saldo 150
Pilih Tujuan
Pindah ke Penjara
Pemilik?
Belum Ada & Saldo Cukup
Milik Sendiri & Level < 2
Tawarkan Beli Properti
Milik Orang Lain
Belum Ada & Saldo Kurang
Tawarkan Upgrade
Tidak Bisa Beli
Bayar Sewa
Beli?
Upgrade?
Tidak
Ya
Transfer Saldo ke Pemilik
Tidak/Saldo Kurang
Ya & Saldo Cukup
Lewati
Beli Properti Kurangi Saldo
Lewati
Upgrade Level Kurangi Saldo
Cek Kebangkrutan
Saldo < 0?
Tidak
Ya
Cek Dadu Kembar
Bangkrut Lepas Semua Properti Status Tidak Aktif
Dadu Kembar?
Ya
Tidak
Giliran Lagi
Ganti Pemain
Cek Pemain Aktif
Pemain Aktif > 1?
Ya
Tidak
Ganti ke Pemain Berikutnya
Tampilkan Pemenang Tampilkan Statistik
END

Gambar 4.1

4.2 Penjelasan Flowchart Permainan Monopoly

1. Tahap Inisialisasi Game

Setup Awal

- **Input Jumlah Pemain**: Sistem meminta input jumlah pemain antara 2-4 orang
- **Validasi**: Jika jumlah tidak valid, sistem akan meminta input ulang
- **Inisialisasi Data**: Setiap pemain mendapat saldo awal 1500 dan ditempatkan di posisi Start (petak 0)
- **Pembuatan Papan**: Sistem membuat papan dengan 40 petak berbagai jenis
- **Pemilihan Karakter**: Setiap pemain memilih karakter unik (Mobil, Topi, Anjing, Kapal)

2. Menu Utama

Setelah setup selesai, permainan masuk ke menu utama dengan 5 opsi:

Menu yang Tersedia:

1. **Giliran** - Mulai giliran bermain
2. **Lihat Kartu** - Menampilkan properti yang dimiliki pemain
3. **Lihat Papan** - Menampilkan seluruh petak di papan
4. **Cari Petak** - Mencari petak berdasarkan nama
5. **Jual Properti** - Menjual properti yang dimiliki
6. **Keluar** - Mengakhiri permainan

Catatan: Fitur "Tambah Properti Manual" telah dihapus dari menu utama

3. Sistem Giliran Bermain

Pengecekan Status Penjara

- Jika pemain di penjara, ada 3 kemungkinan:
    - **Lempar dadu kembar**: Langsung keluar penjara
    - **Gagal 2 kali**: Harus bayar denda 100 untuk keluar
    - **Gagal < 2 kali**: Skip turn dan tetap di penjara

Pelemparan Dadu Normal

- Pemain melempar 2 dadu (d1 & d2)
- Posisi baru = posisi lama + total dadu
- **Bonus Start**: Jika melewati petak Start, dapat bonus 200

Aturan Dadu Kembar

- Jika dadu kembar: pemain dapat giliran tambahan
- Jika 3x berturut-turut dadu kembar: langsung masuk penjara

4. Jenis-Jenis Petak

Petak Normal (Properti)

Berdasarkan status kepemilikan:

- **Belum dimiliki & saldo cukup**: Ditawarkan untuk membeli
- **Milik sendiri & level < 2**: Ditawarkan untuk upgrade
- **Milik orang lain**: Harus bayar sewa
- **Belum dimiliki & saldo kurang**: Tidak bisa berbuat apa-apa

Petak Khusus

- **Masuk Penjara**: Pemain langsung dipindah ke penjara
- **Parkir Bebas**: Pemain bisa memilih tujuan teleportasi
- **Bonus Uang**: Menambah saldo 150
- **Bayar Pajak**: Mengurangi saldo 100
- **Kesempatan**: Efek acak (bonus/penalty/teleport)
- **Dana Umum**: Efek acak bonus/penalty

5. Sistem Properti

Pembelian Properti

- Pemain ditawarkan membeli jika saldo mencukupi
- Jika membeli: saldo berkurang, properti menjadi milik pemain
- Jika tidak: properti tetap tidak dimiliki

Upgrade Bangunan

- Hanya bisa upgrade properti milik sendiri
- Maksimal level 2
- Biaya upgrade = (level + 1) × 150

Pembayaran Sewa

- Sewa = harga properti ÷ 2 + (level × 100)
- Saldo pemain berkurang, saldo pemilik bertambah

6. Sistem Jual Properti

Proses Penjualan

- Menampilkan daftar properti yang dimiliki pemain
- Pemain memilih properti yang ingin dijual
- Uang kembali = harga properti + (level × 100)
- Properti kembali menjadi tidak dimiliki

7. Pengecekan Kebangkrutan

Kondisi Bangkrut

- Jika saldo pemain < 0: pemain dinyatakan bangkrut
- Semua properti milik pemain dilepas
- Status pemain menjadi tidak aktif
- Pemain tidak bisa bermain lagi

8. Kondisi Akhir Permainan

Penentuan Pemenang

- Permainan berlanjut selama masih ada > 1 pemain aktif
- Jika hanya tersisa 1 pemain aktif: pemain tersebut menang
- Sistem menampilkan pemenang dan statistik akhir

Statistik Akhir

- Saldo akhir setiap pemain
- Jumlah properti yang dimiliki
- Total level bangunan

9. Fitur Pencarian

Cari Petak

- Pemain dapat mencari petak berdasarkan nama
- Sistem akan menampilkan informasi lengkap petak yang ditemukan
- Jika tidak ditemukan, sistem memberikan pesan error

10. Alur Berulang

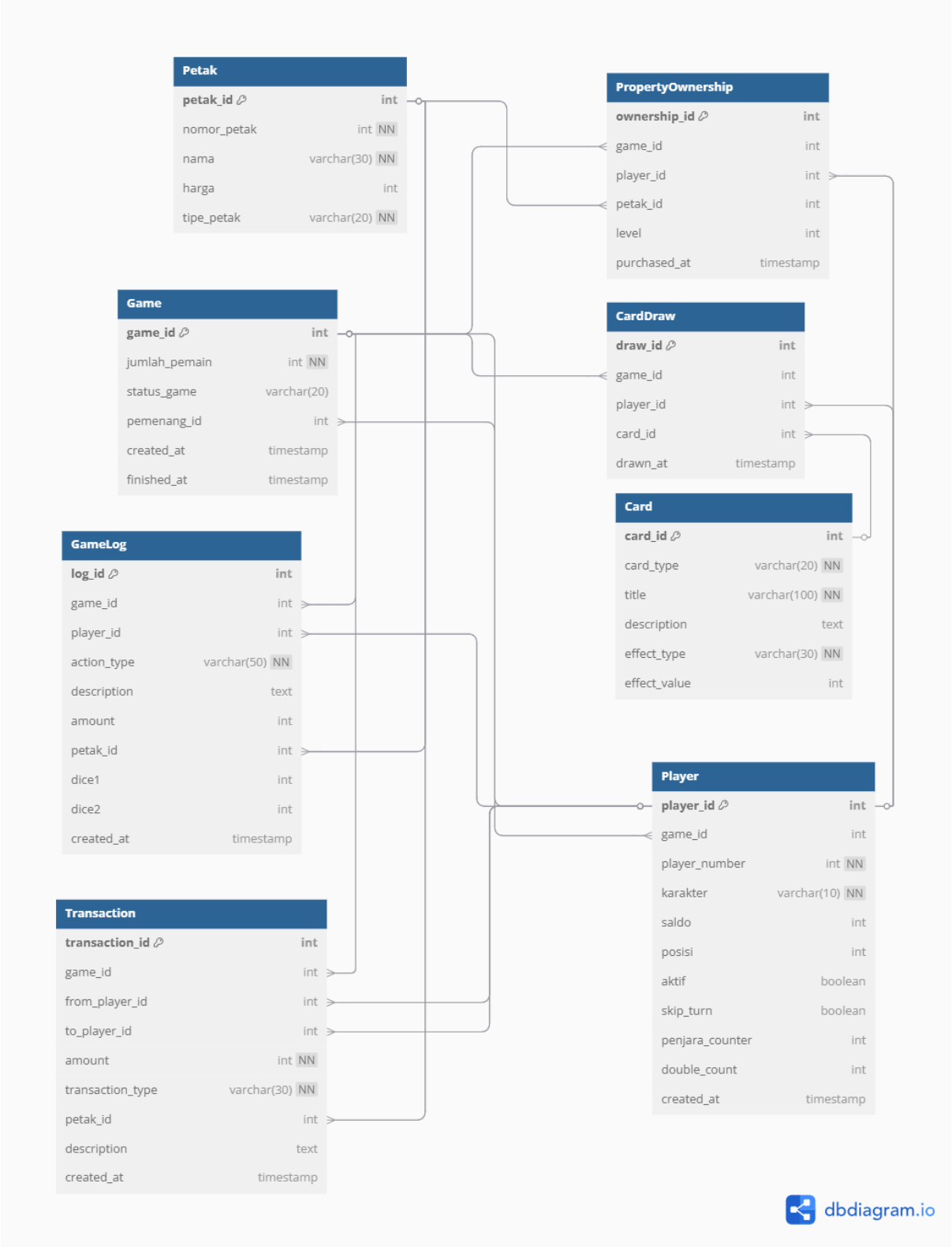
Permainan berjalan dalam loop:

1. Pemain memilih menu
2. Aksi dilakukan sesuai pilihan
3. Kembali ke menu utama
4. Ganti ke pemain berikutnya (jika memilih giliran)
5. Berulang hingga ada pemenang

Tambahan

- **Dadu Kembar**: Memberikan giliran tambahan tapi 3x berturut masuk penjara
- **Melewati Start**: Selalu dapat bonus 200
- **Penjara**: Ada beberapa cara keluar (dadu kembar, bayar denda, atau otomatis setelah 2 giliran)
- **Teleportasi**: Parkir Bebas memungkinkan pemain memilih tujuan
- **Kartu Acak**: Kesempatan dan Dana Umum memberikan efek random

4.2 Entity Relationship Diagram



Gambar 4.2

1. Struktur Database Utama

Tabel Game

- **Fungsi**: Mengelola session permainan
- **Key Features**:
    - Setiap permainan punya ID unik
    - Mencatat jumlah pemain (2-4)
    - Status game (active/finished)
    - Pemenang dan timestamp
- **Keuntungan**: Bisa menjalankan multiple game sessions bersamaan

Tabel Player

- **Fungsi**: Menyimpan data setiap pemain per game
- **Key Features**:
    - Terhubung ke game tertentu
    - Karakter unik per game (Mobil, Topi, Anjing, Kapal)
    - Real-time data: saldo, posisi di papan, status
    - Status khusus: di penjara, skip turn, double count
- **Constraint**: Kombinasi (game_id, player_number) harus unik

Tabel Petak

- **Fungsi**: Master data 40 petak di papan Monopoly
- **Key Features**:
    - Setiap petak punya nomor tetap (0-39)
    - Nama kota (Jakarta, Bandung, dll)
    - Harga dan tipe petak
    - Data statis yang digunakan semua game

2. Sistem Kepemilikan Properti

Tabel PropertyOwnership

- **Fungsi**: Mengelola kepemilikan properti per game
- **Key Features**:
    - Menghubungkan pemain dengan properti yang dimiliki
    - Level bangunan (0-2) untuk upgrade
    - Timestamp pembelian
- **Constraint**: Satu properti hanya bisa dimiliki satu pemain per game

3. Sistem Tracking & Audit

Tabel GameLog

- **Fungsi**: Mencatat semua aktivitas dalam game
- **Key Features**:
    - Log setiap aksi (lempar dadu, beli properti, bayar sewa)
    - Nilai dadu untuk setiap lemparan
    - Deskripsi detail aktivitas
    - Jumlah uang yang terlibat
- **Keuntungan**: Bisa replay game atau debug masalah

Tabel Transaction

- **Fungsi**: Mencatat semua transaksi keuangan
- **Key Features**:
    - Transfer uang antar pemain
    - Tipe transaksi (sewa, pembelian, pajak, bonus)
    - Dari pemain mana ke pemain mana
- **Keuntungan**: Audit trail keuangan yang lengkap

4. Sistem Kartu

Tabel Card

- **Fungsi**: Master data kartu Kesempatan dan Dana Umum
- **Key Features**:
    - Tipe kartu dan deskripsi
    - Efek yang ditimbulkan
    - Nilai efek (jumlah uang, dll)

Tabel CardDraw

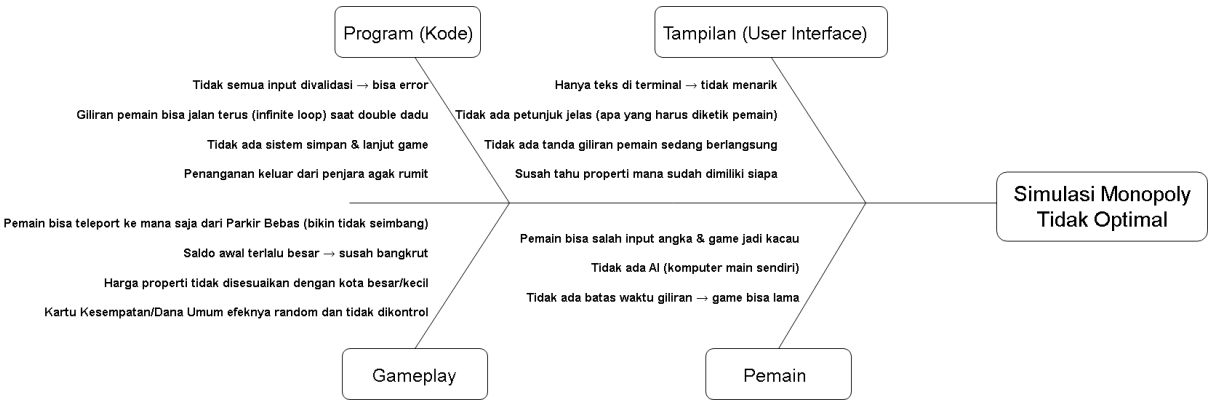
- **Fungsi**: Riwayat pengambilan kartu
- **Key Features**:
    - Siapa yang ambil kartu apa dan kapan
    - Terhubung ke game dan pemain tertentu

5. Relationship Antar Tabel

1. Game (1) ——— (N) Player
2. Game (1) ——— (N) PropertyOwnership
3. Game (1) ——— (N) GameLog
4. Game (1) ——— (N) Transaction
5. Game (1) ——— (N) CardDraw

6. Player (1) —— (N) PropertyOwnership
7. Player (1) —— (N) GameLog
8. Player (1) —— (N) Transaction (sebagai from/to)
9. Player (1) —— (N) CardDraw
10. Petak (1) —— (N) PropertyOwnership
11. Petak (1) —— (N) GameLog
12. Petak (1) —— (N) Transaction
13. Card (1) —— (N) CardDraw

4.3 FishBone Diagram



Gambar 4.3

Fishbone Diagram, atau diagram tulang ikan (disebut juga Ishikawa diagram), digunakan untuk mengidentifikasi akar penyebab utama dari suatu masalah kompleks secara sistematis. Dalam proyek pengembangan game Monopoli ini, fishbone digunakan untuk menganalisis penyebab dari permasalahan

"Simulasi Monopoly Tidak Optimal"

Diagram dibagi ke dalam empat kategori penyebab besar:

1. Program

Aspek teknis dalam pemrograman yang memengaruhi kelancaran permainan.

- Validasi input tidak lengkap: Pemain bisa memasukkan nilai yang salah tanpa dicegah, sehingga program bisa error.

- Logika double dadu tidak dibatasi: Jika pemain terus mendapatkan double, maka bisa terjadi giliran tak berakhir.
- Tidak ada sistem simpan dan lanjut game: Game hanya bisa dimainkan dalam satu sesi.
- Penanganan keluar dari penjara rumit: Prosedur dadu double dan penalti kurang intuitif atau transparan.

2. Gameplay

Hal-hal terkait *aturan main* dan keseimbangan dalam mekanisme game.

- Pemain bisa teleport bebas dari Parkir: Ini bisa membuat permainan tidak adil.
- Saldo awal terlalu besar: Menyulitkan skenario bangkrut, membuat game terasa terlalu panjang.
- Harga properti tidak proporsional: Semua kota hampir punya harga serupa tanpa mempertimbangkan skala/tingkat kota.
- Efek kartu acak tidak dikontrol: Efek kartu Kesempatan dan Dana Umum bisa terlalu untung atau terlalu merugikan, mengganggu keseimbangan.

3. Tampilan

Berhubungan dengan antarmuka pengguna (UI/UX).

- Hanya teks di terminal: Visual kurang menarik atau membosankan.
- Kurangnya petunjuk navigasi: Pemain pemula bisa bingung langkah apa yang harus dilakukan selanjutnya.

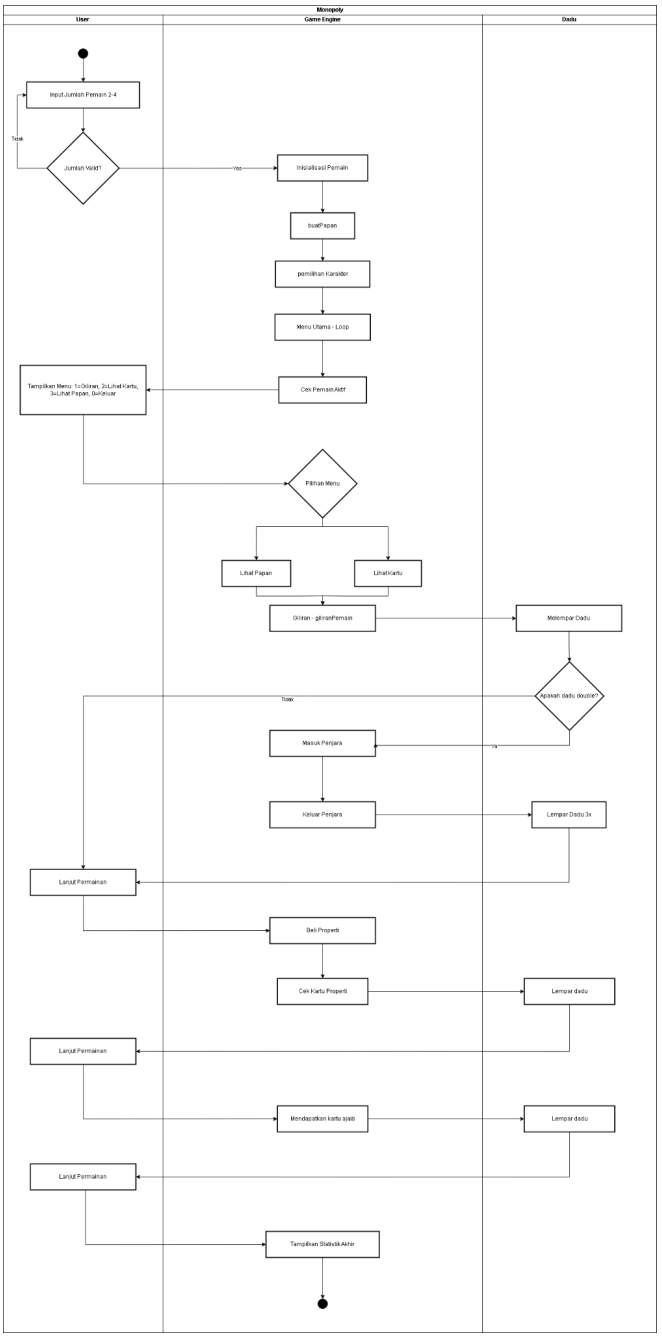
4. Pemain

Masalah yang muncul dari sisi pengguna permainan.

- Kesalahan input sering terjadi: Pemain bisa memasukkan angka di luar batas yang dibolehkan.
- Tidak ada tanda giliran: Giliran pemain tidak ditandai dengan jelas.
- Tidak ada AI: Tidak tersedia mode melawan komputer, jadi permainan hanya bisa dilakukan secara manual oleh 2–4 pemain manusia.

- Tidak ada batas waktu giliran: Pemain bisa terlalu lama berpikir atau meninggalkan permainan, memperlambat jalannya game.

4.4 Activity Diagram



Gambar 4.4

Activity Diagram digunakan untuk memvisualisasikan alur proses atau logika dalam sistem simulasi Monopoli. Diagram ini mencerminkan bagaimana pemain dan sistem berinteraksi dari awal permainan hingga akhir. Activity diagram ini juga membantu memahami urutan kegiatan utama dan bagaimana percabangan logika dikendalikan, termasuk interaksi dengan *game engine* dan *dadu*.

Diagram ini terbagi menjadi tiga *swimlane* utama:

1. User (Pemain)
2. Monopoly Game Engine (Mesin logika permainan)
3. Dadu (Logika acak lemparan dadu)

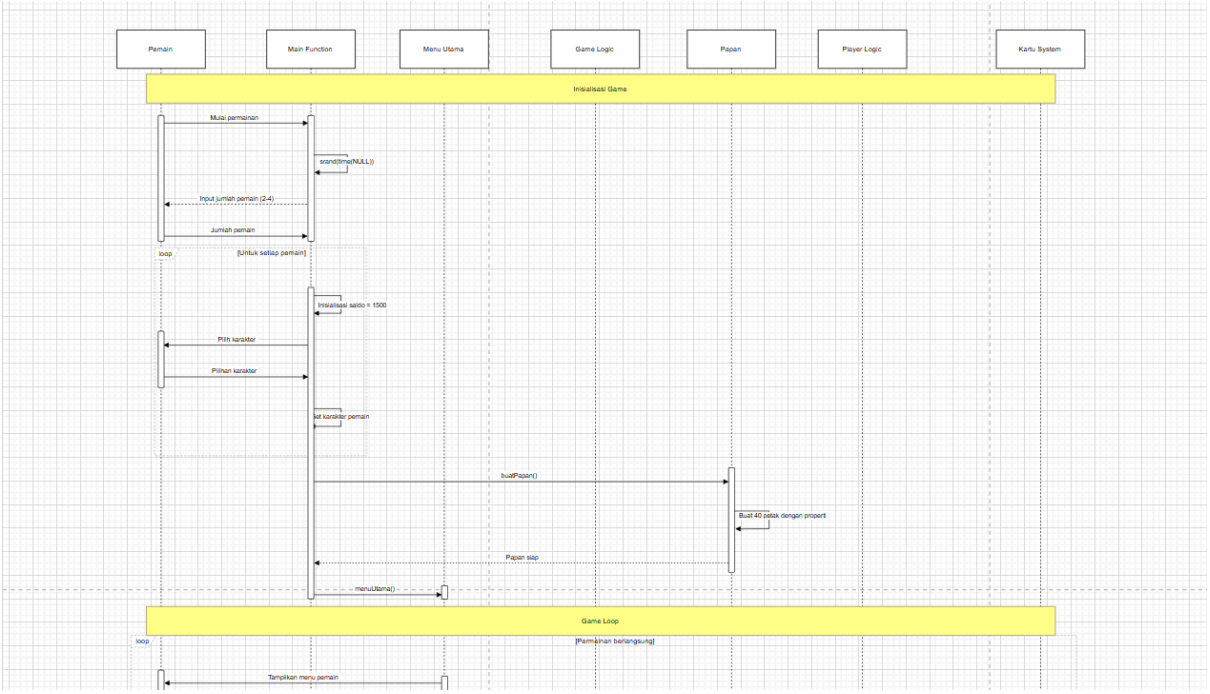
4.4.1 Penjabaran Alur Proses

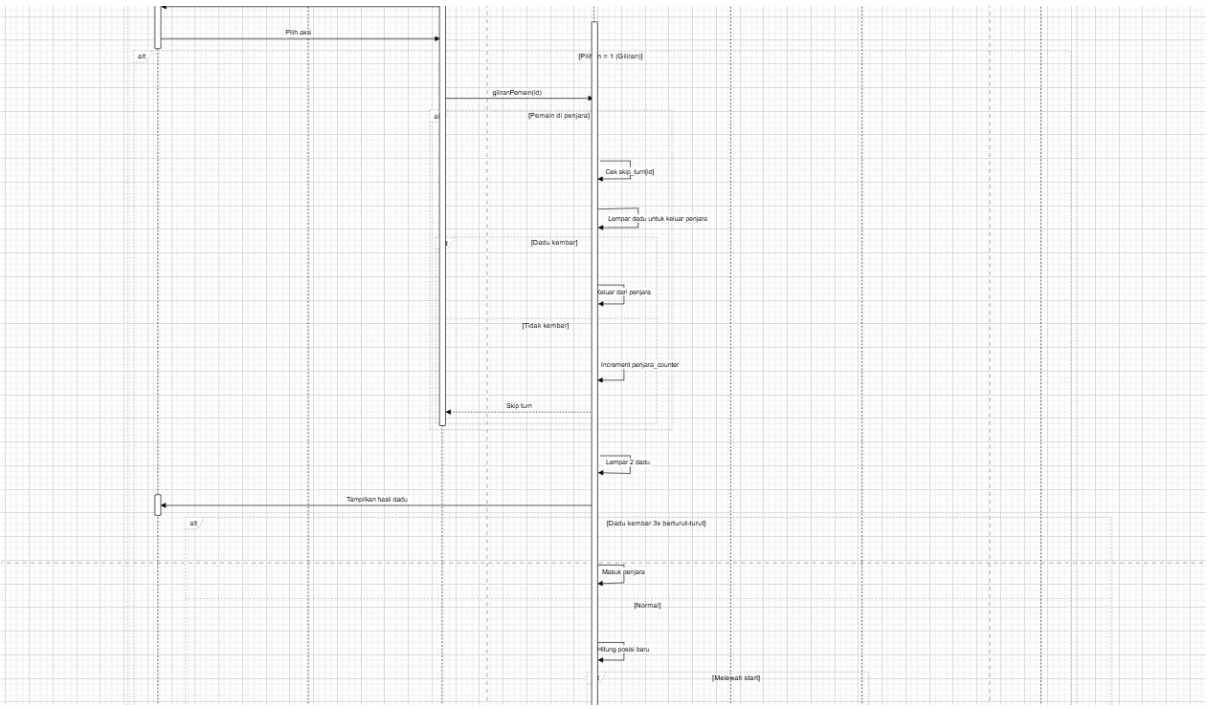
1. **Input Jumlah Pemain**
   Pemain memasukkan jumlah pemain (antara 2–4). Jika tidak valid, proses akan diulang.
2. **Inisialisasi Permainan**
   Jika jumlah valid, sistem akan menginisialisasi pemain, membuat papan (*buatPapan*), memilih karakter, dan masuk ke *menu utama* dalam bentuk loop.
3. **Menu Utama**
   Pemain diberi pilihan menu:
   - 1 = Giliran bermain
   - 2 = Lihat kartu
   - 3 = Lihat papan
   - 4 = Keluar permainan
4. **Cek Pemain Aktif & Pilihan Menu**
   Setelah memilih menu, sistem memeriksa apakah pemain masih aktif (belum bangkrut). Jika tidak, proses kembali ke loop.
5. **Giliran Pemain & Lempar Dadu**
   Jika pemain memilih untuk bermain, sistem akan masuk ke giliran pemain lalu melempar dadu. Sistem memeriksa apakah dadu menunjukkan angka ganda (*double*):
   - Jika double: pemain dapat lempar ulang hingga 3 kali. Jika terus double, masuk penjara.
   - Jika tidak double: lanjut ke efek petak.
6. **Efek Petak dan Aksi Pemain**
   Berdasarkan lokasi baru:
   - Masuk Penjara: jika terkena petak penjara.
   - Beli Properti: jika petak dapat dibeli.
   - Cek Kartu Properti: jika ingin melihat detail.

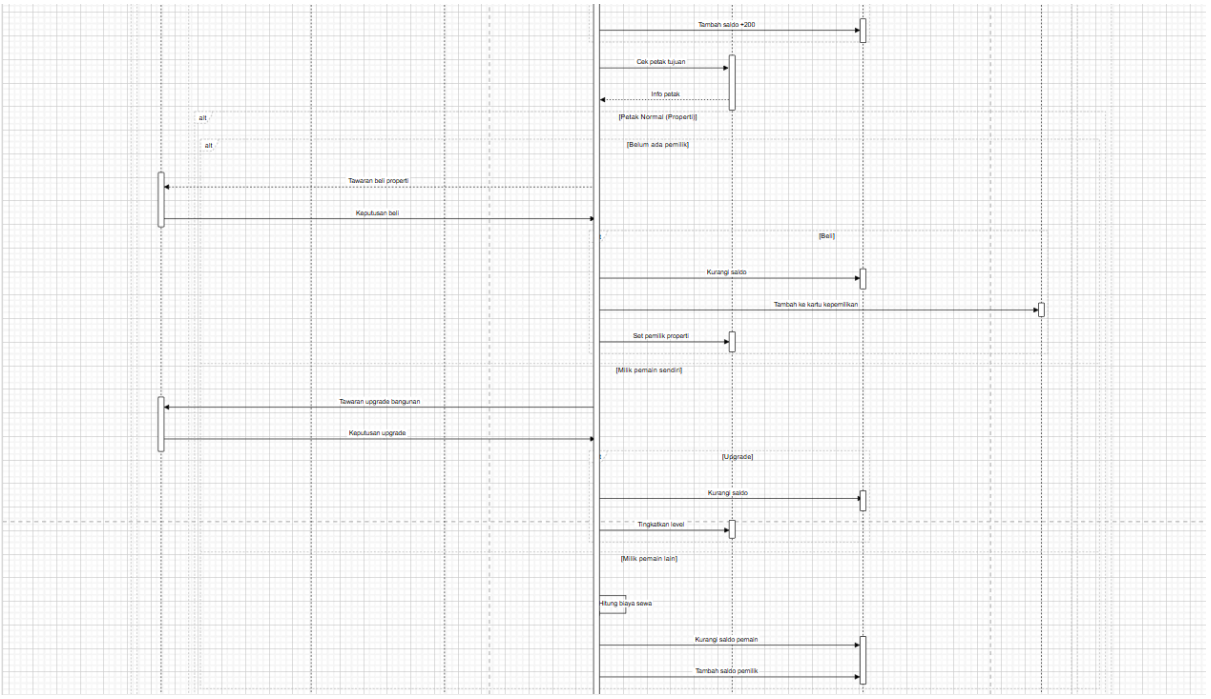
- Mendapat Kartu Spesial: jika petak memberikan kartu Dana Umum/Kesempatan.

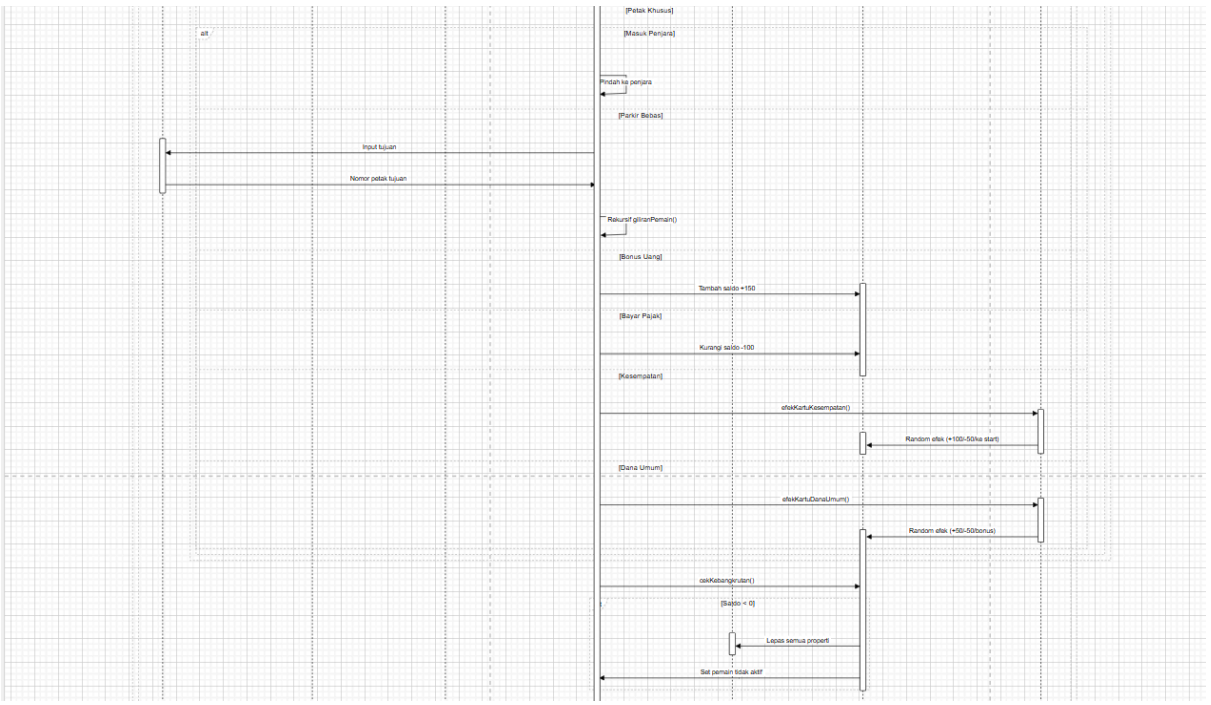
7. **Lempar Dadu & Ulangi**
   Setelah aksi selesai, pemain akan melempar dadu lagi jika sesuai kondisi, atau giliran beralih.
8. **Tampilkan Statistik Akhir**
   Jika permainan berakhir (misalnya hanya 1 pemain tersisa), sistem akan menampilkan statistik akhir (saldo, properti, pemenang).

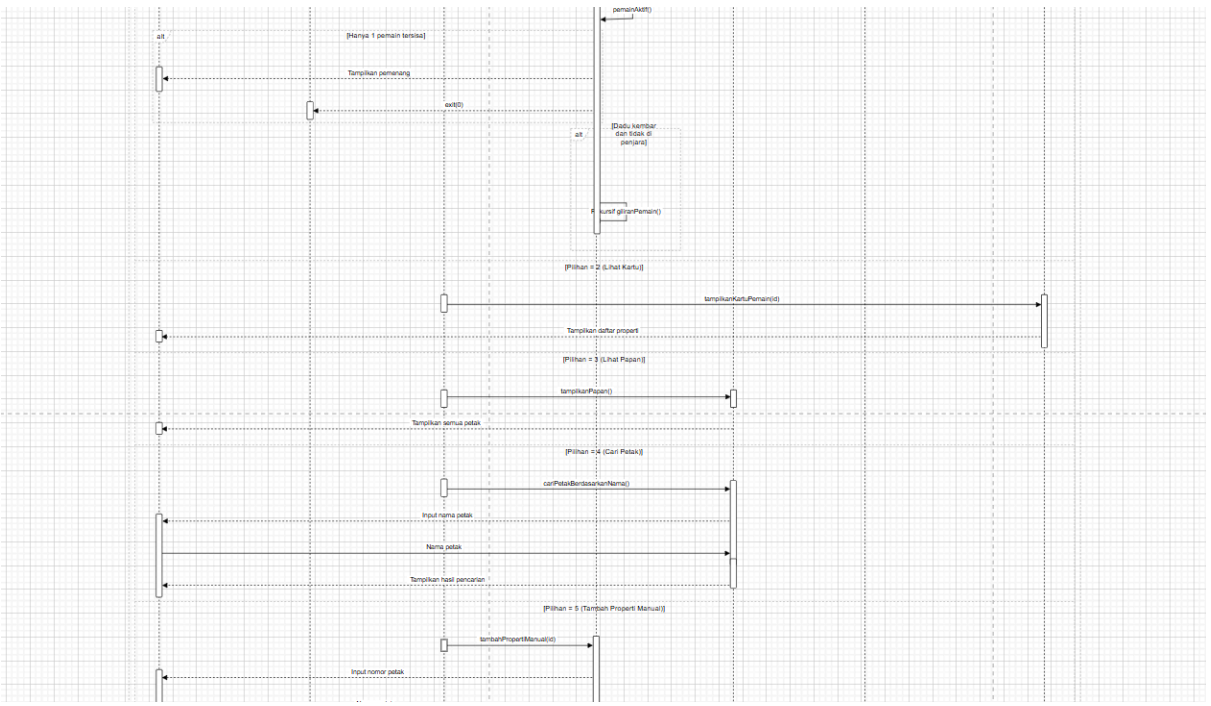
4.5 Sequence Diagram

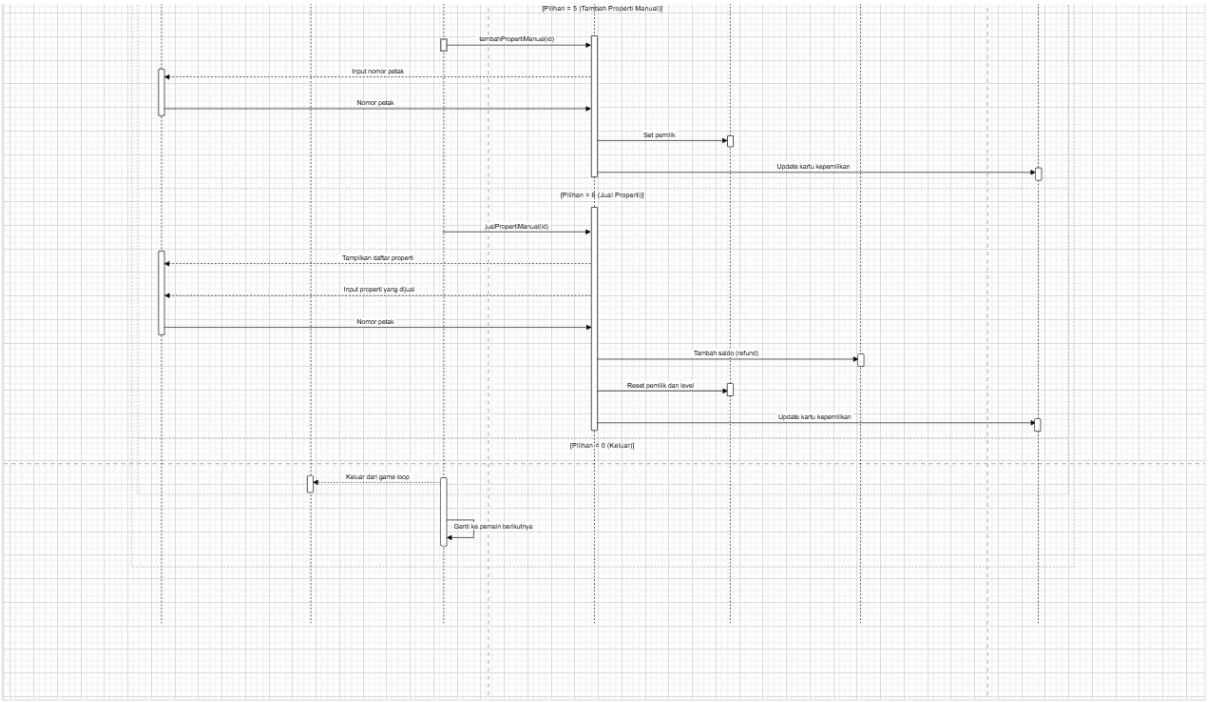



Pilih aksi
alt
[Pilih n = 1 (Giliran)]
gilironPemain(id)
alt
[Pemain di penjara]
Cek skip_turn[id]
Lempar dadu untuk keluar penjara
[Dadu kembar]
Keluar dari penjara
[Tidak kembar]
Increment penjara_counter
Skip turn
Lempar 2 dadu
Tampilkan hasil dadu
alt
[Dadu kembar 3x berturut-turut]
Masuk penjara
[Normal]
Hitung posisi baru
[Melewati start]


Tambah saldo +200
Cek petak tujuan
Info petak
alt
[Petak Normal (Property)]
alt
[Belum ada pemilik]
Tawaran beli properti
Keputusan beli
[Beli]
Kurangi saldo
Tambah ke kartu kepemilikan
Set pemilik properti
[Milik pemain sendiri]
Tawaran upgrade bangunan
Keputusan upgrade
[Upgrade]
Kurangi saldo
Tingkatkan level
[Milik pemain lain]
Hitung biaya sewa
Kurangi saldo pemain
Tambah saldo pemilik


[Petak Khusus]
alt
[Masuk Penjara]
Pindah ke penjara
[Parkir Bebas]
Input tujuan
Nomor petak tujuan
Rekursif giliranPemain()
[Bonus Uang]
Tambah saldo +150
[Bayar Pajak]
Kurangi saldo -100
[Kesempatan]
efekKartuKesempatan()
Random efek (+100/-50/ke start)
[Dana Umum]
efekKartuDanaUmum()
Random efek (+50/-50/bonus)
cekKebangkrutan()
[Saldo < 0]
Lepas semua properti
Set pemain tidak aktif


pemainAktif()
alt
[Hanya 1 pemain tersisa]
Tampilkan pemenang
exit(0)
alt
[Dadu kembar dan tidak di penjara]
Rekursif giliranPemain()
[Pilihan = 2 (Lihat Kartu)]
tampilkanKartuPemain(id)
Tampilkan daftar properti
[Pilihan = 3 (Lihat Papan)]
tampilkanPapan()
Tampilkan semua petak
[Pilihan = 4 (Cari Petak)]
cariPetakBerdasarkanNama()
Input nama petak
Nama petak
Tampilkan hasil pencarian
[Pilihan = 5 (Tambah Properti Manual)]
tambahPropertiManual(id)
Input nomor petak



Gambar 4.5

Sequence diagram ini menggambarkan **sistem permainan board game** (mirip Monopoly) yang melibatkan 7 komponen utama yang saling berkomunikasi untuk menjalankan permainan berbasis giliran.

4.5.1 Penjabaran Alur Proses

FASE 1: INISIALISASI GAME

Setup Awal

- **User** memulai permainan dengan menjalankan program
- **Main Function** melakukan inisialisasi random seed dengan `srand(time(NULL))`
- Sistem meminta **User** memasukkan jumlah pemain (2-4)

Pemilihan Karakter

- Untuk setiap pemain, sistem melakukan loop:
    - Inisialisasi saldo pemain = 1500
    - Meminta **User** memilih karakter (Mobil, Topi, Anjing, Kapal)
    - Menyimpan pilihan karakter untuk setiap pemain

Pembuatan Papan

- **Main** memanggil `buatPapan()` ke komponen **Board**
- **Board** membuat 40 petak dengan berbagai properti dan tipe
- Papan siap digunakan, kontrol diberikan ke **Menu Utama**

FASE 2: GAME LOOP UTAMA

Menu Pemain

Setiap giliran, sistem menampilkan menu dengan 7 pilihan:

1. Giliran (main)
2. Lihat Kartu
3. Lihat Papan
4. Cari Petak
5. Tambah Properti Manual
6. Jual Properti
7. Keluar

FASE 3: PILIHAN 1 - GILIRAN PEMAIN (Alur Paling Kompleks)

A. Pengecekan Status Penjara

Jika pemain di penjara (skip_turn = true):
- Lempar dadu untuk mencoba keluar
- Jika dadu kembar → Keluar penjara
- Jika tidak kembar → Increment counter, skip turn

B. Lempar Dadu Normal

- Sistem lempar 2 dadu
- Tampilkan hasil ke **User**
- **Pengecekan Khusus**: Jika dadu kembar 3x berturut-turut → Masuk penjara

C. Perpindahan Posisi

- Hitung posisi baru = (posisi lama + total dadu) % 40
- **Bonus Start**: Jika melewati petak "Mulai" → Tambah saldo +200

D. Pengecekan Jenis Petak

Petak Normal (Properti)
Jika belum ada pemilik:
- Tawarkan beli properti

- User memilih Ya/Tidak
- Jika Ya: Kurangi saldo, update kepemilikan

Jika milik pemain sendiri:
- Tawarkan upgrade bangunan
- User memilih Ya/Tidak
- Jika Ya: Kurangi saldo, tingkatkan level

Jika milik pemain lain:
- Hitung biaya sewa
- Kurangi saldo pemain aktif
- Tambah saldo pemilik properti

Petak Khusus

- **Masuk Penjara**: Pindah langsung ke petak penjara
- **Parkir Bebas**: User input tujuan, rekursif panggil giliran lagi
- **Bonus Uang**: Tambah saldo +150
- **Bayar Pajak**: Kurangi saldo -100
- **Kesempatan**: Random efek (+100/-50/pindah ke start)
- **Dana Umum**: Random efek (+50/-50/bonus acak)

E. Pengecekan Akhir Giliran

- **Cek Kebangkrutan**: Jika saldo < 0 → Lepas semua properti, nonaktifkan pemain
- **Cek Pemenang**: Jika hanya 1 pemain tersisa → Tampilkan pemenang, keluar program
- **Dadu Kembar**: Jika dadu kembar dan tidak di penjara → Giliran lagi

FASE 4: PILIHAN MENU LAINNYA

Pilihan 2: Lihat Kartu

- **Menu → Card System** → Tampilkan daftar properti yang dimiliki pemain

Pilihan 3: Lihat Papan

- **Menu → Board** → Tampilkan semua 40 petak dengan info lengkap

Pilihan 4: Cari Petak

- **Board** meminta input nama petak dari **User**
- Pencarian case-insensitive
- Tampilkan hasil pencarian

Pilihan 5: Tambah Properti Manual

- **User** input nomor petak
- Sistem validasi petak
- Update kepemilikan di **Board** dan **Card System**

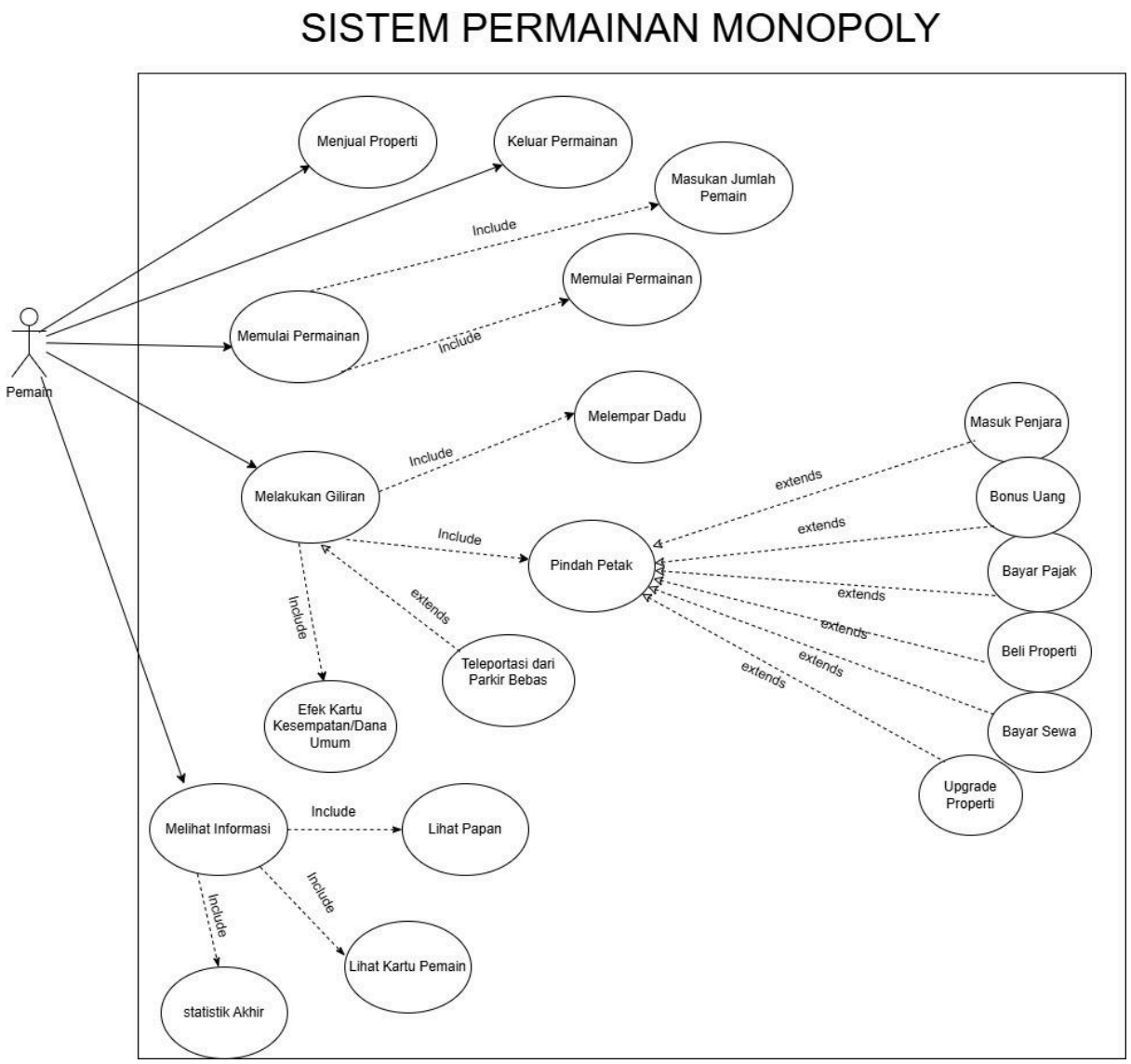
Pilihan 6: Jual Properti

- Tampilkan daftar properti yang dimiliki
- **User** pilih properti yang dijual
- Hitung refund (harga + level * 100)
- Update saldo, reset kepemilikan

Pilihan 0: Keluar

- Keluar dari game loop

4.6 Use Case Diagram



Gambar 4.6

Use case diagram ini menggambarkan bagaimana aktor utama, yaitu Pemain, berinteraksi dengan berbagai fungsi yang tersedia dalam sistem permainan Monopoly. Terdapat beberapa *use case utama* yang bisa dilakukan oleh pemain, yaitu: Memulai Permainan, Melakukan Giliran, Menjual Properti, Melihat Informasi, dan Keluar Permainan.

Saat pemain memilih Memulai Permainan, sistem akan menuntun pemain untuk Memasukkan Jumlah Pemain dan Memilih Karakter yang merupakan bagian (<<include>>) dari proses inisialisasi permainan. Setelah itu, permainan berjalan dalam siklus giliran.

Dalam setiap Melakukan Giliran, terdapat proses wajib (<<include>>), yaitu Melempar Dadu dan Pindah Petak, yang kemudian mengarahkan pemain ke berbagai

kemungkinan hasil. Petak yang didarati akan memicu *use case tambahan* (<<extends>>), seperti:

- Masuk Penjara jika berada di petak hukuman,
- Bonus Uang jika berada di petak hadiah,
- Bayar Pajak di petak pajak,
- Beli Properti jika petak belum dimiliki,
- Bayar Sewa jika petak dimiliki pemain lain,
- Upgrade Properti jika pemain mendarat di propertinya sendiri.

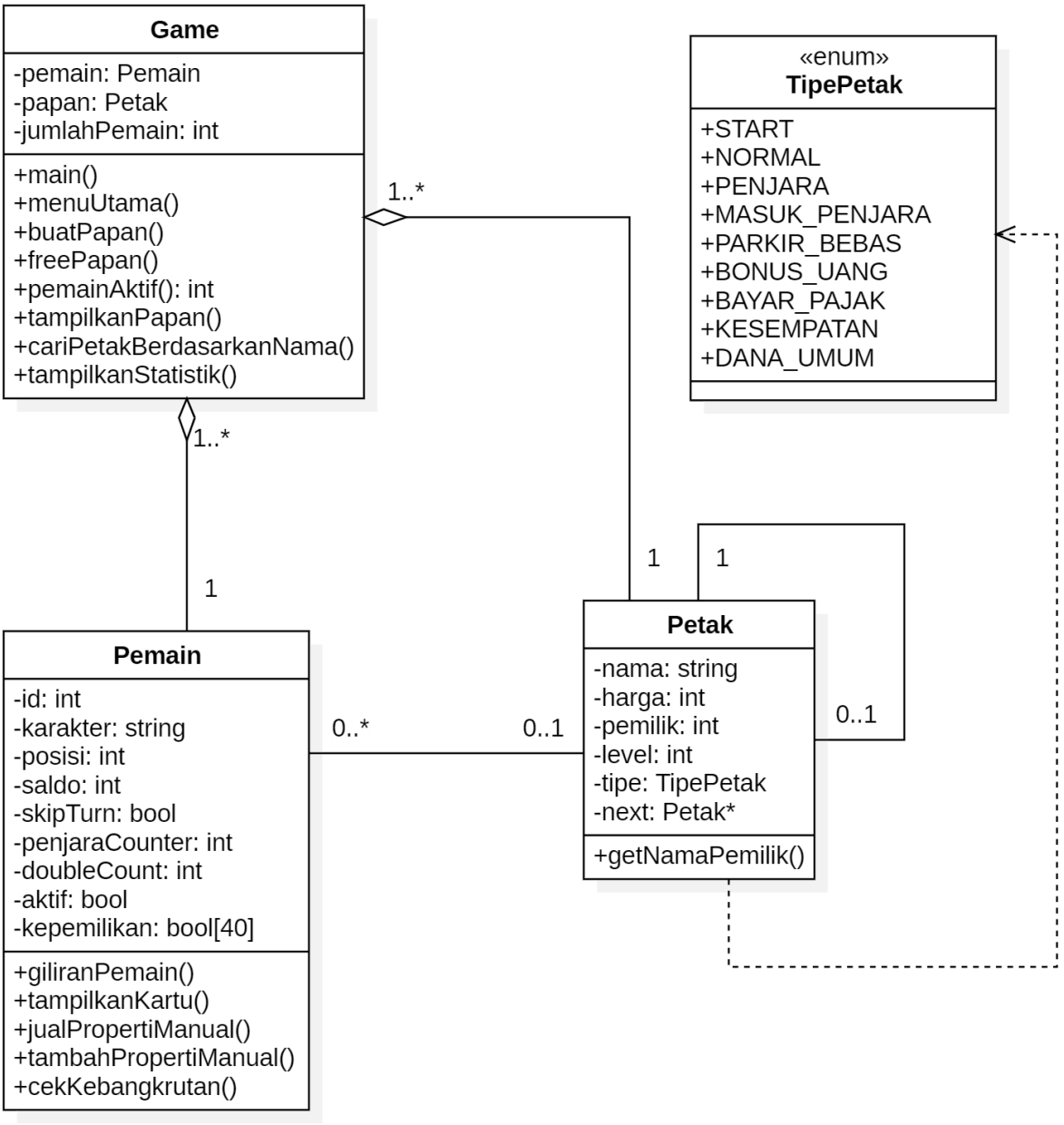
Jika pemain mendarat di petak Parkir Bebas, maka akan ada kemungkinan melakukan Teleportasi, yang diperluas dari (<<extends>>) alur pindah petak.

Selain itu, selama giliran, pemain juga bisa mendapatkan Efek Kartu Kesempatan atau Dana Umum, yang disertakan (<<include>>) dalam giliran.

Pemain juga memiliki akses untuk melihat informasi selama permainan melalui use case Melihat Informasi, yang mencakup Lihat Papan, Lihat Kartu Pemain, dan Statistik Akhir.

Sebagai bagian dari strategi permainan, pemain dapat menggunakan fitur Menjual Properti kapan saja, serta dapat memilih untuk Keluar Permainan di tengah permainan.

4.7 Class Diagram



Gambar 4.7

4.7.1 Penjelasan Tiap Kelas dan Relasinya

Class diagram ini menyusun struktur program menjadi 4 class utama dan 1 enum, yaitu:

1. Game

Peran: Sebagai controller utama yang mengatur semua komponen permainan.

- Atribut:
  - papan: Petak[40] → array berisi semua petak pada papan permainan.
  - pemain: Pemain[] → daftar pemain yang ikut bermain.
  - jumlahPemain: int → jumlah pemain aktif saat ini.

- Operasi (Method):
    - main() → memulai permainan.
    - menuUtama() → menampilkan dan menangani interaksi menu.
    - buatPapan() → inisialisasi semua petak.
    - freePapan() → menghapus alokasi memori petak.
    - pemainAktif() → mengecek jumlah pemain aktif.
    - tampilkanPapan() → menampilkan daftar petak.
    - cariPetakBerdasarkanNama() → mencari petak berdasarkan input nama.
    - tampilkanStatistik() → menampilkan hasil akhir permainan.
- Relasi:
    - Aggregation dengan Petak → Game memiliki 40 petak.
    - Aggregation dengan Pemain → Game memiliki 2–4 pemain.

  2. Pemain

  Peran: Menyimpan data dan status permainan masing-masing pemain.
- Atribut:
    - id: int → nomor unik pemain (0–3)
    - karakter: string → karakter pilihan (Mobil, Topi, dsb.)
    - posisi: int → posisi pemain di papan
    - saldo: int → uang yang dimiliki pemain
    - skipTurn: bool → apakah pemain melewatkan giliran
    - penjaraCounter: int → berapa kali gagal keluar dari penjara
    - doubleCount: int → jumlah double berturut-turut
    - aktif: bool → apakah pemain masih aktif
    - kepemilikan: bool[40] → status kepemilikan petak
- Operasi:
    - giliranPemain() → mengatur alur giliran pemain
    - tampilkanKartu() → menampilkan properti yang dimiliki
    - jualPropertiManual() → menjual properti milik sendiri
    - tambahPropertiManual() → menambah properti secara manual
    - cekKebangkrutan() → memeriksa jika saldo pemain negatif
- Relasi:
    - Association dengan Petak → Pemain dapat memiliki banyak properti (petak).
    - Association dengan string karakter → Pemain menyimpan nama karakter.

  3. Petak

  Peran: Mewakili setiap petak atau kotak dalam papan permainan.
- Atribut:
    - nama: string → nama kota atau jenis petak
    - harga: int → harga beli petak (hanya untuk NORMAL)
    - pemilik: int → ID pemain yang memiliki properti, -1 jika belum ada
    - level: int → level bangunan di atas petak (maks 2)
    - tipe: TipePetak → tipe petak (NORMAL, PENJARA, dsb.)
    - next: Petak* → menunjuk ke petak berikutnya (linked list)
- Operasi:

- getNamaPemilik() → menampilkan nama pemilik petak

- Relasi:
  - Association dengan Petak → petak membentuk linked list (next).
  - Dependency dengan TipePetak → setiap petak memiliki 1 tipe.

4. TipePetak (Enum)
Peran: Menentukan jenis petak di papan.

- Nilai:
  - START, NORMAL, PENJARA, MASUK_PENJARA, PARKIR_BEBAS, BONUS_UANG, BAYAR_PAJAK, KESMPATAN, DANA_UMUM

4.7.2 Relasi Antarkelas dan Kardinalitas

Class A	Class B	Relationship Type	Multiplicity	Penjelasan
Game	Petak	Aggregation	1 → 40	Game memiliki 40 petak (papan)
Game	Pemain	Aggregation	1 → 2..4	Game memiliki 2–4 pemain
Pemain	Petak	Association	0..*	Pemain bisa memiliki banyak properti
Petak	Petak	Association	1	Masing-masing petak terhubung ke petak berikutnya (linked list)
Petak	TipePetak	Dependency	1	Satu petak hanya punya satu tipe

Table 4.7.2

4.8 Pemrograman Awal

Pemrograman adalah proses menulis, menguji, dan memelihara kode instruksi untuk mengendalikan komputer. Kode ini ditulis dalam bahasa pemrograman sebagai alat komunikasi antara manusia dan komputer. Tujuan utama pemrograman adalah menghasilkan sistem atau keluaran yang dapat digunakan untuk memenuhi tujuan tertentu secara efisien dan terstruktur.

Dalam pengembangan program permainan Monopoly versi C ini, tahapan pemrograman awal dimulai dengan perancangan struktur data dan komponen logika permainan dasar. Bahasa pemrograman C digunakan karena kemampuannya yang kuat dalam pengelolaan memori, struktur kontrol, dan efisiensi eksekusi, yang sangat mendukung pembuatan game berbasis teks.

Adapun beberapa tahapan penting dalam pemrograman awal yang telah dilakukan antara lain:

- **Deklarasi Konstanta dan Struktur Data**

  Permainan ini menggunakan `#define` untuk mendefinisikan batasan seperti jumlah petak dan pemain maksimum. Struktur `Petak` dirancang sebagai simpul utama papan permainan, dengan atribut seperti nama, harga, pemilik, dan tipe petak.
- **Inisialisasi Papan Permainan**

  Fungsi `buatPapan()` berperan penting dalam membentuk papan monopoli sebanyak 40 petak, termasuk petak spesial seperti "Mulai", "Penjara", "Parkir Bebas", "Masuk Penjara", serta petak "Kesempatan" dan "Dana Umum". Setiap petak dihubungkan secara siklik agar pergerakan pemain dapat melingkar.
- **Pembuatan Logika Pemain dan Karakter**

  Pemain memilih karakter unik di awal permainan. Setiap pemain diatur memiliki posisi awal, saldo, dan properti kosong. Sistem penanda giliran juga diterapkan melalui array indeks pemain aktif.
- **Menu Utama dan Interaktif**

  Menu permainan dirancang agar pemain dapat memilih aksi seperti melanjutkan giliran, melihat properti, dan mengecek papan. Ini dilakukan melalui `menuUtama()` yang memberikan antarmuka teks sederhana namun interaktif.
- **Simulasi Giliran dan Dadu**

  Sistem pergerakan menggunakan dua dadu. Jika pemain mendapatkan angka dadu yang sama (double), mereka mendapat giliran tambahan. Jika hal tersebut terjadi tiga kali berturut-turut, pemain langsung masuk ke penjara.
- **Penanganan Properti dan Transaksi**

  Logika pembelian, pembayaran sewa, dan upgrade properti diatur melalui fungsi `giliranPemain()`. Pemain dapat meng-upgrade properti dari tanah menjadi rumah dan kemudian hotel dengan biaya tertentu.

- **Sistem Penjara dan Pembebasan**

  Pemain yang berada di petak penjara memiliki tiga opsi untuk keluar: membayar, mendapat dadu double, atau otomatis bebas setelah tiga giliran.
- **Statistik Akhir dan Pemenang**

  Ketika hanya tersisa satu pemain yang masih aktif (tidak bangkrut), permainan berhenti dan ditampilkan statistik akhir berupa total saldo, jumlah properti, dan level bangunan milik setiap pemain.

Tahapan pemrograman awal ini menjadi fondasi penting dalam membangun keseluruhan fitur permainan. Pendekatan modular dan penggunaan struktur data yang sistematis memudahkan proses pengembangan, pengujian, dan pemeliharaan kode secara keseluruhan.

4.8.1 Header File dan Definisi

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#include <time.h>
```

Bagian ini memuat seluruh file header standar yang digunakan untuk kebutuhan input/output, alokasi memori, manipulasi string, dan pembangkitan bilangan acak.

```
#define MAX_PETAK 40

#define MAX_PEMAIN 4

#define MAX_KARAKTER 4

#define MAX_PROPERTI 40
```

Sedangkan definisi makro digunakan untuk menetapkan batas maksimal dari petak, pemain, karakter, dan properti

4.8.2 Struktur Data Program

```
typedef enum { NORMAL, START, PENJARA, MASUK_PENJARA, PARKIR_BEBAS,
BONUS_UANG, BAYAR_PAJAK, KESMPATAN, DANA_UMUM } TipePetak;
```

```
typedef struct Petak {
    char nama[30];
    int harga;
    int pemilik;
    int level;
    TipePetak tipe;
    struct Petak* next;
} Petak;
```

Enum TipePetak digunakan untuk mengklasifikasikan tipe-tipe petak dalam permainan. Struct Petak digunakan sebagai elemen utama papan permainan, disusun membentuk linked list sirkular.

4.8.3 Inisialisasi Papan Permainan

```
void buatPapan() {
    for (int i = 0; i < MAX_PETAK; i++) {
        papan[i] = (Petak*)malloc(sizeof(Petak));
        strcpy(papan[i]->nama, namaKota[i]);
        papan[i]->harga = (tipePetak[i] == NORMAL) ? 100 + (i * 10) : 0;
        papan[i]->pemilik = -1;
        papan[i]->level = 0;
        papan[i]->tipe = tipePetak[i];
        papan[i]->next = papan[(i + 1) % MAX_PETAK];
    }
}
```

Fungsi ini membentuk papan permainan berupa linked list sirkular. Properti diinisialisasi dengan harga, nama, tipe, dan status belum dimiliki.

4.8.4 Pemilihan Karakter Pemain

```
void pemilihanKarakter() {
    int terpakai[MAX_KARAKTER] = {0};
```

```
    for (int i = 0; i < jumlah_pemain; i++) {
        int pilih;
        do {
            printf("Pemain %d pilih karakter:\n", i + 1);
            for (int j = 0; j < MAX_KARAKTER; j++)
                if (!terpakai[j]) printf("%d. %s\n", j + 1, karakter_list[j]);
            scanf("%d", &pilih);
        } while (pilih < 1 || pilih > MAX_KARAKTER || terpakai[pilih - 1]);
        terpakai[pilih - 1] = 1;
        strcpy(pemain_karakter[i], karakter_list[pilih - 1]);
    }
}
```

Fungsi ini memastikan setiap pemain memilih karakter unik dari daftar karakter_list.

4.8.5 Giliran Pemain

```
void giliranPemain(int id) {
    if (skip_turn[id]) {
        ... // logika keluar penjara
    }
    int d1 = rand() % 6 + 1, d2 = rand() % 6 + 1;
    ... // logika double, dadu, posisi
    ... // efek petak (beli properti, upgrade, sewa, kartu, dll.)
    cekKebangkrutan(id);
    if (d1 == d2 && !skip_turn[id]) giliranPemain(id); // double = giliran lagi
}
```

Fungsi ini adalah inti logika permainan. Mengelola lemparan dadu, pemrosesan petak, transaksi properti, upgrade, penalti penjara, hingga eliminasi pemain.

4.8.6 Efek Kartu Kesempatan dan Dana Umum

```
void efekKartuKesempatan(int id) {
    int efek = rand() % 3;
```

```
    ...
}

void efekKartuDanaUmum(int id) {
    int efek = rand() % 3;
    ...
}
```

Efek acak dari kartu kesempatan dan dana umum dapat menambah saldo, mengurangi saldo, atau memindahkan pemain.

4.8.7 Statistik Akhir dan Penentuan Pemenang

```
void tampilkanStatistik();
int pemainAktif();
```

Fungsi ini akan menampilkan status akhir permainan berupa jumlah saldo, properti, dan level bangunan. Jika hanya satu pemain tersisa, maka permainan selesai.

4.8.8 Fungsi Utama Program

```
int main() {
    srand(time(NULL));
    ... // input jumlah pemain dan inisialisasi
    buatPapan();
    pemilihanKarakter();
    menuUtama();
    return 0;
}
```

Fungsi utama mengatur alur permainan dari input awal hingga akhir sesi permainan, dengan struktur modular dan fungsi yang saling terpanggil secara teratur.

4.8.9 Menu Utama dan Interaktif

```
void menuUtama() {
    int giliran = 0;
    while (1) {
        if (!aktif[giliran]) { giliran = (giliran + 1) % jumlah_pemain; continue; }
        printf("\n--- Pemain %d (%s) ---\n", giliran + 1, pemain_karakter[giliran]);
        printf("1. Giliran\n2. Lihat Kartu\n3. Lihat Papan\n0. Keluar\nPilih: ");
        int p; scanf("%d", &p);
        if (p == 1) {
```

```
            giliranPemain(giliran);
            pemainAktif();
            giliran = (giliran + 1) % jumlah_pemain;
        } else if (p == 2) tampilkanKartuPemain(giliran);
        else if (p == 3) tampilkanPapan();
        else if (p == 0) break;
    }
}
```

Fungsi ini adalah antarmuka utama permainan. Pemain dapat memilih untuk menjalankan giliran, melihat kartu properti, melihat papan permainan, atau keluar dari permainan.

4.8.10 Tampilan Papan Permainan

```
void tampilkanPapan() {
  printf("\n\U0001F4E6 Daftar Petak:\n");
  for (int i = 0; i < MAX_PETAK; i++) {
    printf("Petak %2d: %-18s | Harga: %4d | Pemilik: %s\n", i,
        papan[i]->nama, papan[i]->harga, getNamaPemilik(papan[i]->pemilik));
  }
}
```

Fungsi ini mencetak seluruh daftar petak yang tersedia di papan, lengkap dengan nama kota, harga beli, dan status kepemilikan.

4.8.11 Tampilan Kartu Properti Pemain

```
void tampilkanKartuPemain(int id) {
    printf("\n\U0001F4C7 Kartu Properti Pemain %d (%s):\n", id + 1,
pemain_karakter[id]);
    int punya = 0;
    for (int i = 0; i < MAX_PETAK; i++) {
        if (kartu_kepemilikan[id][i]) {
            printf("- %s (Level: %d)\n", papan[i]->nama, papan[i]->level);
            punya = 1;
        }
    }
    if (!punya) printf("(Belum punya properti)\n");
```

```
}
```

Fungsi ini mencetak daftar properti yang dimiliki oleh pemain tertentu beserta tingkat bangunan yang dimiliki di atasnya.

4.8.12 Pencarian Petak Berdasarkan Nama

Program menyediakan fitur pencarian petak berdasarkan nama kota yang dimasukkan oleh pemain. Fungsi cariPetakBerdasarkanNama() akan membandingkan input dengan nama-nama petak pada papan tanpa membedakan huruf kapital, sehingga memudahkan pemain dalam mencari petak tertentu. Jika petak ditemukan, informasi lengkap mengenai harga, pemilik, dan level akan ditampilkan. Berikut potongan kode programnya:

```
void cariPetakBerdasarkanNama() {
    char cari[30];
    printf("Masukkan nama kota: ");
    scanf(" %[^"]", cari);
    for (int i = 0; i < MAX_PETAK; i++) {
        if (samaIgnoreCase(papan[i]->nama, cari)) {
            printf("Petak %d: %s, Harga: %d, Pemilik: %s, Level: %d\n",
                   i, papan[i]->nama, papan[i]->harga,
                   getNamaPemilik(papan[i]->pemilik), papan[i]->level);
            return;
        }
    }
    printf("Petak tidak ditemukan.\n");
}
```

Fungsi `samaIgnoreCase()` digunakan untuk membandingkan string tanpa memperhatikan kapital huruf, memastikan pencarian tetap fleksibel terhadap input pemain.

BAB V

PENUTUP

5.1 Kesimpulan

Proyek pengembangan aplikasi simulasi permainan Monopoli berbasis C++ ini telah berhasil menunjukkan penerapan berbagai konsep penting dalam pemrograman dan struktur data. Melalui pendekatan berorientasi objek dan pemanfaatan struktur seperti *linked list*, *queue*, *hash table*, hingga *enum*, simulasi permainan dapat dijalankan secara interaktif dan sesuai dengan aturan permainan asli Monopoli.

Fitur-fitur utama seperti sistem giliran pemain, pembelian dan pengelolaan properti, kartu acak (Kesempatan dan Dana Umum), serta mekanisme penjara dan kebangkrutan telah berhasil diimplementasikan dengan baik. Proses perancangan papan menggunakan struktur data *circular linked list* juga menjadi pendekatan yang efektif untuk merepresentasikan papan permainan melingkar.

Selain itu, proyek ini memberikan kontribusi signifikan dalam meningkatkan pemahaman mahasiswa terhadap integrasi antara logika permainan, struktur data, algoritma, dan konsep OOP. Implementasi berbasis teks (CLI) telah mempermudah pengujian serta memperkuat fondasi dalam pengembangan sistem game sederhana. Simulasi juga mampu mencatat data transaksi, giliran, hingga statistik akhir yang berguna untuk evaluasi permainan.

Secara keseluruhan, proyek ini tidak hanya berhasil memenuhi tujuan akademik Ujian Akhir Semester, namun juga memberikan nilai praktis sebagai prototipe dasar pengembangan game berbasis console menggunakan bahasa C++.

5.2 Saran

1. Pengembangan Tampilan Grafis (GUI):
   Untuk meningkatkan pengalaman pengguna, aplikasi ini dapat dikembangkan lebih lanjut dengan antarmuka grafis berbasis desktop, seperti menggunakan library `SFML`, `Qt`, atau `SDL`.
2. Peningkatan Kompleksitas AI:
   Pengembangan pemain komputer (bot) dengan kecerdasan buatan sederhana dapat menjadi tantangan menarik untuk mendalami algoritma pengambilan keputusan.
3. Penambahan Fitur Multiplayer Online:
   Implementasi sistem permainan daring berbasis jaringan akan membuka peluang kolaboratif antar pemain dari perangkat yang berbeda.
4. Optimasi Struktur Data dan Memori:
   Penggunaan struktur data dapat ditingkatkan agar lebih efisien secara waktu dan ruang, termasuk pengelolaan memori dinamis yang lebih hati-hati untuk menghindari *memory leak*.
5. Pengujian Unit dan Dokumentasi Otomatis:
   Menambahkan unit testing pada tiap fungsi utama dan membuat dokumentasi kode secara otomatis akan meningkatkan keandalan dan keberlanjutan proyek ke depannya.
6. Penambahan Sistem Simpan dan Lanjutkan Permainan:
   Fitur untuk menyimpan sesi permainan dan melanjutkannya kembali sangat berguna agar pemain tidak harus memulai ulang dari awal.

Dengan berbagai pengembangan dan evaluasi berkala, proyek simulasi game Monopoli ini berpotensi untuk dikembangkan menjadi aplikasi edukatif maupun hiburan yang lebih lengkap dan profesional.

DAFTAR PUSTAKA

DDeitel, H. M., & Deitel, P. J. (2016). *C++ how to program* (10th ed.). Pearson Education.

Malik, D. S. (2010). *C++ programming: From problem analysis to program design* (5th ed.). Cengage Learning.

Weiss, M. A. (2014). *Data structures and algorithm analysis in C++* (4th ed.). Pearson.

Wahid, A. (2022). *Struktur data dan algoritma menggunakan C++*. Andi Publisher.

Raymond, E. S. (2003). *The art of Unix programming*. Addison-Wesley

Hasbro. (2023). *Monopoly official rules*. https://www.hasbro.com/common/instruct/monopoly.pdf

GeeksforGeeks. (n.d.). *Data structures in C++*. Retrieved June 2025, from https://www.geeksforgeeks.org/data-structures/

TutorialsPoint. (n.d.). *C++ programming language.* Retrieved June 2025, from https://www.tutorialspoint.com/cplusplus/index.htm

cppreference.com. (n.d.). *C++ reference.* Retrieved June 2025, from https://en.cppreference.com/

GameDev.net. (n.d.). *Creating a board game simulation in C++*. Retrieved June 2025, from https://www.gamedev.net/