SISTEM INFORMASI

PEMROGRAMAN PENGELOLAAN FILE DALAM JAVA

UJIAN KOMPETENSI

REKAYASA PERANGKAT LUNAK

SMK TI BALI GLOBAL
DENPASAR

Disusun Oleh :

Nama : Theo Darmawan
NIS : 6196
Kelas : XII RPL 1

SEKOLAH MENENGAH KEJURUAN TEKNOLOGI INFORMASI

SMK TI BALI GLOBAL DENPASAR

2023

KATA PENGANTAR

Puji syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya sehingga kami dapat menyelesaikan tugas yang berjudul "Pemrograman Pengelolaan File dalam Java" ini dengan baik dan tepat waktu. Tugas ini kami susun sebagai bagian dari pembelajaran dalam bidang pemrograman Java, khususnya dalam pengelolaan file yang mencakup proses pembuatan, pembacaan, penulisan, serta penghapusan file secara programatis. Dengan memahami konsep ini, diharapkan kami dapat mengembangkan sistem yang lebih efisien dalam menangani data berbasis file.

Kami mengucapkan terima kasih kepada semua pihak yang telah membantu dalam penyusunan tugas ini, baik dalam bentuk dukungan materi maupun bimbingan teknis. Semoga tugas ini dapat menjadi referensi yang bermanfaat bagi rekan-rekan yang ingin memahami lebih dalam tentang pengelolaan file dalam Java. Kami menyadari bahwa tugas ini masih memiliki kekurangan. Oleh karena itu, kritik dan saran yang membangun sangat kami harapkan untuk penyempurnaan di masa mendatang.

Denpasar, 1 Februari 2025

Penyusun Tugas

DAFTAR ISI

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam dunia pemrograman, pengelolaan file merupakan aspek penting dalam menyimpan, membaca, dan memanipulasi data secara persisten. Java sebagai salah satu bahasa pemrograman yang banyak digunakan dalam pengembangan perangkat lunak menyediakan berbagai kelas dan metode untuk menangani operasi file dengan efisien.

Seiring dengan meningkatnya kebutuhan akan sistem yang dapat mengelola data dalam bentuk file teks maupun biner, pemahaman tentang cara kerja file dalam Java menjadi krusial bagi pengembang perangkat lunak. Dengan menggunakan kelas seperti File, FileReader, FileWriter, BufferedReader, dan BufferedWriter, programmer dapat melakukan berbagai operasi seperti membaca, menulis, dan menghapus data dari file dengan mudah dan terstruktur.

Namun, dalam implementasinya, masih terdapat tantangan seperti penanganan error saat mengakses file, efisiensi dalam membaca dan menulis data dalam jumlah besar, serta keamanan data dalam file. Oleh karena itu, diperlukan pemahaman mendalam tentang konsep dan teknik pengelolaan file dalam Java agar sistem yang dibangun dapat berjalan dengan baik, efisien, dan sesuai dengan kebutuhan.

1.2 Rumusan Masalah

a. Bagaimana cara mengimplementasikan operasi file dalam Java untuk membaca dan menulis data secara efisien?
b. Bagaimana penggunaan kelas dan metode dalam Java dapat mempermudah pengelolaan file secara otomatis?
c. Bagaimana teknik penanganan error (exception handling) dapat diterapkan untuk menghindari kesalahan dalam pengelolaan file?
d. Bagaimana cara memastikan keamanan dan integritas data saat membaca dan menulis file dalam Java?

1.3 Ruang Lingkup

a. Implementasi operasi file dalam Java, termasuk pembuatan, pembacaan, penulisan, dan penghapusan file.
b. Penggunaan kelas bawaan Java seperti File, FileWriter, FileReader, BufferedReader, dan BufferedWriter untuk mengelola file.
c. Penerapan mekanisme penanganan kesalahan (exception handling) dalam operasi file untuk memastikan keamanan dan kestabilan program.
d. Penyusunan struktur program yang modular untuk pengelolaan file agar dapat digunakan dalam berbagai skenario pengolahan data.
e. Analisis manfaat penggunaan file dalam pengelolaan data berbasis teks dibandingkan dengan sistem basis data.

1.4 Tujuan

a. Merancang sistem pengelolaan file dalam Java yang dapat membaca dan menulis data secara efisien.
b. Meningkatkan akurasi dan keandalan dalam penyimpanan serta pemrosesan data berbasis file.
c. Menyediakan solusi berbasis Java yang mempermudah akses dan manipulasi data dalam file secara sistematis.

BAB II
PEMBAHASAN

2.1 Spesifikasi Perangkat

Terdapat perangkat yang dapat digunakan. Dalam proses pengerjaan sistem ini, perangkat yang digunakan meliputi:

Tabel 2.1 Spesifikasi Perangkat

No	Perangkat	Spesifikasi
1	Laptop	• OS: Windows 10 (64 bit) • Processor: 11th Gen Intel® Core™ i5-11400H • RAM: 4,00 GB

2.2 Text Editor

Dalam proses pengembangan sistem ini, digunakan Visual Studio Code (VS Code). VS Code adalah editor kode sumber yang dikembangkan oleh Microsoft, yang mendukung berbagai bahasa pemrograman termasuk Java. VS Code memiliki fitur IntelliSense untuk membantu penulisan kode, debugging tools, serta dukungan untuk ekstensi yang memperluas fungsionalitasnya.

Gambar 2.1 Visual Studio Code

2.3 Bahasa Pemrograman

Bahasa pemrograman yang digunakan dalam pembuatan sistem informasi SKKPd ini ialah Java.

Gambar 2.2 Java

Java adalah bahasa pemrograman berorientasi objek yang digunakan untuk membangun berbagai jenis aplikasi, mulai dari perangkat lunak desktop, aplikasi web, hingga sistem berbasis Android. Java dirancang agar dapat berjalan di berbagai platform menggunakan konsep "Write Once, Run Anywhere" (WORA), yang berarti kode Java dapat dijalankan di sistem operasi apa pun yang memiliki Java Virtual Machine (JVM).

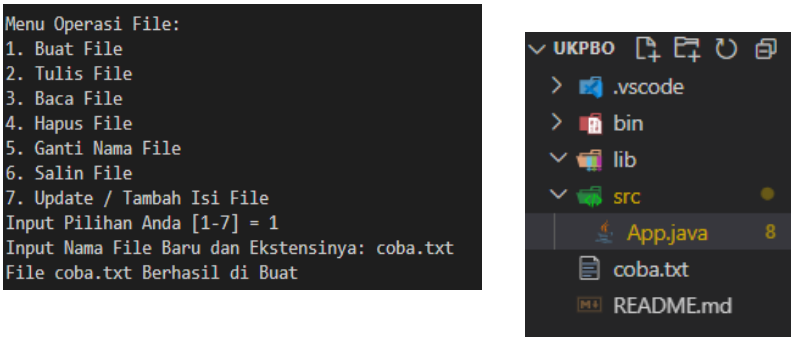
Java sering digunakan dalam pengembangan aplikasi berbasis enterprise, sistem backend, serta pengelolaan data dalam skala besar. Dengan fitur seperti garbage collection, multithreading, dan keamanan tinggi, Java menjadi pilihan utama untuk pengembangan perangkat lunak yang stabil dan efisien.

2.4 Penjelasan Kode Program

```
Menu Operasi File:
1. Buat File
2. Tulis File
3. Baca File
4. Hapus File
5. Ganti Nama File
6. Salin File
7. Update / Tambah Isi File
Input Pilihan Anda [1-7] =
```

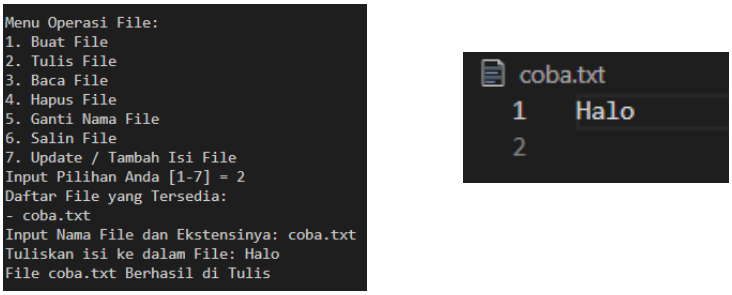
Gambar 2.3 Tampilan Output Program

Program ini memiliki beberapa fitur utama yang memungkinkan pengguna untuk mengelola file dengan mudah. Berikut adalah penjelasan dari masing-masing fitur:



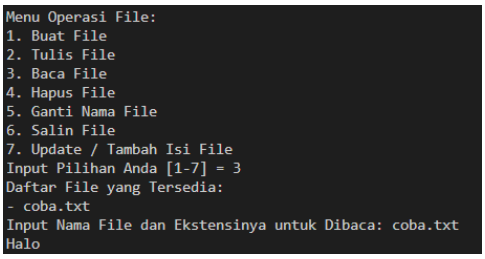
Gambar 2.4 Fitur Membuat File

Membuat File BaruPengguna dapat membuat file baru dengan mengetikkan nama file yang diinginkan. Jika file sudah ada, program akan memberikan pilihan untuk membaca atau menghapusnya.



Gambar 2.5 Fitur Menulis File

Menulis ke dalam FileSetelah file dibuat, pengguna dapat menambahkan isi ke dalamnya. Misalnya, jika pengguna menulis "Halo", maka program akan menyimpan teks tersebut dalam file.

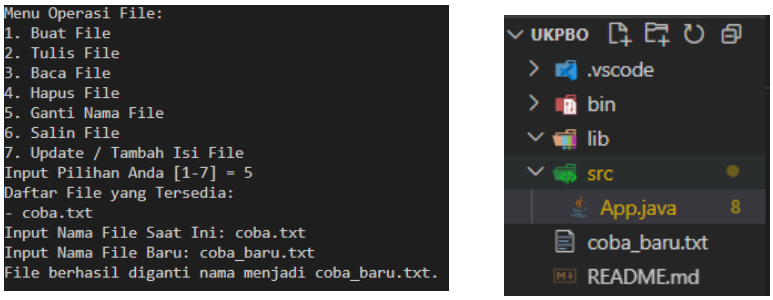


Gambar 2.6 Fitur Membaca File

Membaca Isi FileProgram dapat membaca isi dari file yang telah dibuat dan menampilkan kontennya di layar.

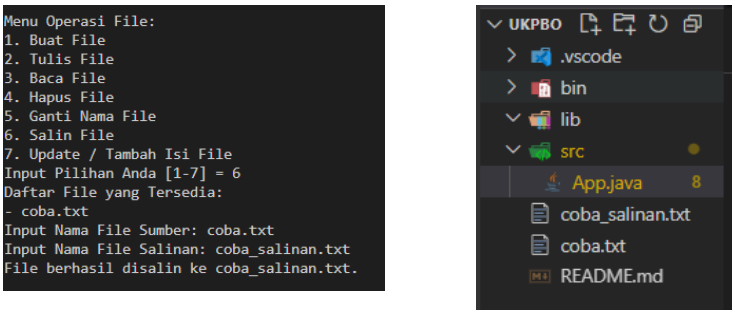
Gambar 2.7 Fitur Menghapus File

Menghapus FileJika pengguna ingin menghapus file tertentu, mereka cukup memasukkan nama file yang ingin dihapus, dan program akan menghapusnya jika file tersebut ditemukan.



Gambar 2.8 Fitur Mengganti Nama File

Mengganti Nama FilePengguna dapat mengganti nama file lama dengan nama baru untuk memudahkan pengelolaan dokumen.



Gambar 2.9 Fitur Salin File

Menyalin FileProgram dapat membuat salinan dari suatu file, sehingga pengguna memiliki backup dari file yang mereka butuhkan.

Gambar 2.10 Fitur Menambah Isi File

Menambah Isi ke dalam FilePengguna dapat menambahkan teks baru ke dalam file yang sudah ada. Misalnya, jika file sebelumnya berisi "Halo", kemudian ditambahkan kata "hai", maka isi file menjadi: “Halo” dan “hai”.

2.5 Pembuatan Kode Program

```
import java.io.*;
import java.nio.file.*;
import java.util.*;

public class App {

    public static void main(String[] args) throws IOException {
        App menu = new App();
        menu.utama();
    }

    public void utama() throws IOException {
        Scanner input = new Scanner(System.in);
        System.out.println("Menu Operasi File:");
        System.out.println("1. Buat File");
        System.out.println("2. Tulis File");
        System.out.println("3. Baca File");
        System.out.println("4. Hapus File");
        System.out.println("5. Ganti Nama File");
        System.out.println("6. Salin File");
        System.out.println("7. Update / Tambah Isi File");
        System.out.print("Input Pilihan Anda [1-7] = ");

        String n = input.nextLine();

        switch (n) {
            case "1":
                buatFile();
                break;
            case "2":
                tampilkanDaftarFile();
                tulisFile();
                break;
            case "3":
                tampilkanDaftarFile();
                bacaFile();
                break;
```

```
                case "4":
                    tampilkanDaftarFile();
                    hapusFile();
                    break;
                case "5":
                    tampilkanDaftarFile();
                    gantiNamaFile();
                    break;
                case "6":
                    tampilkanDaftarFile();
                    salinFile();
                    break;
                case "7":
                    tampilkanDaftarFile();
                    updateFile();
                    break;
                default:
                    System.out.println("Pilihan tidak valid.");
            }

            utama();
        }

        public void tampilkanDaftarFile() {
            File folder = new File(".");
            File[] files = folder.listFiles((dir, name) ->
name.endsWith(".txt"));
            if (files != null && files.length > 0) {
                System.out.println("Daftar File yang Tersedia:");
                for (File file : files) {
                    System.out.println("- " + file.getName());
                }
            } else {
                System.out.println("Anda belum membuat file.\n");
            }
        }

        public void buatFile() throws IOException {
            Scanner input = new Scanner(System.in);
            System.out.print("Input Nama File Baru dan Ekstensinya: ");
            String namaFile = input.nextLine();

            File file = new File(namaFile);

            if (!file.exists()) {
                try {
                    Formatter formatter = new Formatter(namaFile);
                    System.out.println("File " + namaFile + " Berhasil di
Buat\n");
                    formatter.close();
                } catch (Exception err) {
                    System.out.println("Anda Gagal Membuat File " +
namaFile);
                    System.out.println("Error: " + err.getMessage() +
```

```
"\n");
            }
        } else {
            System.out.println("File Sudah Terdapat! Silahkan
melakukan Baca, atau Hapus pada File " + namaFile + ".\n");
        }
    }

    public void tulisFile() throws IOException {
        Scanner input = new Scanner(System.in);
        System.out.print("Input Nama File dan Ekstensinya: ");
        String namaFile = input.nextLine();

        File file = new File(namaFile);

        if (file.exists()) {
            try {
                FileWriter fileWriter = new FileWriter(namaFile,
true);
                System.out.print("Tuliskan isi ke dalam File: ");
                String isiFile = input.nextLine();

                fileWriter.append(isiFile).append("\n");
                System.out.println("File " + namaFile + " Berhasil di
Tulis\n");
                fileWriter.close();
            } catch (Exception err) {
                System.out.println("Error Menulis File " + namaFile);
                System.out.println("Error: " + err.getMessage() +
"\n");
            }
        } else {
            System.out.println("File " + namaFile + " Tidak di
Temukan!\n");
        }
    }



    public void bacaFile() throws IOException {
        Scanner input = new Scanner(System.in);
        System.out.print("Input Nama File dan Ekstensinya untuk
Dibaca: ");
        String namaFile = input.nextLine();

        File file = new File(namaFile);

        if (file.exists()) {
            try (BufferedReader br = new BufferedReader(new
FileReader(namaFile))) {
                String line;
                while ((line = br.readLine()) != null) {
                    System.out.println(line);
                }
```

```
            } catch (Exception err) {
                System.out.println("Error Membaca File " + namaFile);
                System.out.println("Error: " + err.getMessage() +
"\n");
            }
        } else {
            System.out.println("File " + namaFile + " Tidak di
Temukan!\n");
        }
    }

    public void hapusFile() throws IOException {
        Scanner input = new Scanner(System.in);
        System.out.print("Input Nama File dan Ekstensinya yang Akan
Dihapus: ");
        String namaFile = input.nextLine();

        File file = new File(namaFile);
        if (file.exists()) {
            if (file.delete()) {
                System.out.println("File " + namaFile + " berhasil
dihapus.\n");
            } else {
                System.out.println("Gagal menghapus file " + namaFile
+ ".\n");
            }
        } else {
            System.out.println("File " + namaFile + " tidak
ditemukan!\n");
        }
    }

    public void gantiNamaFile() throws IOException {
        Scanner input = new Scanner(System.in);
        System.out.print("Input Nama File Saat Ini: ");
        String namaFileLama = input.nextLine();

        File fileLama = new File(namaFileLama);
        if (fileLama.exists()) {
            System.out.print("Input Nama File Baru: ");
            String namaFileBaru = input.nextLine();
            File fileBaru = new File(namaFileBaru);

            if (fileLama.renameTo(fileBaru)) {
                System.out.println("File berhasil diganti nama
menjadi " + namaFileBaru + ".\n");
            } else {
                System.out.println("Gagal mengganti nama file.\n");
            }
        } else {
            System.out.println("File " + namaFileLama + " tidak
ditemukan!\n");
        }
    }
```

```
    public void salinFile() throws IOException {
        Scanner input = new Scanner(System.in);
        System.out.print("Input Nama File Sumber: ");
        String namaFileSumber = input.nextLine();

        File fileSumber = new File(namaFileSumber);
        if (fileSumber.exists()) {
            System.out.print("Input Nama File Salinan: ");
            String namaFileTujuan = input.nextLine();
            Files.copy(fileSumber.toPath(),
Paths.get(namaFileTujuan), StandardCopyOption.REPLACE_EXISTING);
            System.out.println("File berhasil disalin ke " +
namaFileTujuan + ".\n");
        } else {
            System.out.println("File " + namaFileSumber + " tidak
ditemukan!\n");
        }
    }

    public void updateFile() throws IOException {
        Scanner input = new Scanner(System.in);
        System.out.print("Input Nama File yang Akan Diupdate: ");
        String namaFile = input.nextLine();

        File file = new File(namaFile);

        if (file.exists()) {
            System.out.println("\nIsi File Sebelumnya:");
            try (BufferedReader br = new BufferedReader(new
FileReader(namaFile))) {
                String line;
                while ((line = br.readLine()) != null) {
                    System.out.println(line);
                }
            } catch (Exception err) {
                System.out.println("Error Membaca File: " +
err.getMessage() + "\n");
            }

            System.out.print("\nApakah Anda ingin menimpa isi file
(append) atau menambah (tambah) isi baru? (ketik 'append' /
'tambah'): ");
            String pilihan = input.nextLine();

            boolean isAppend = pilihan.equalsIgnoreCase("tambah");

            try (FileWriter fileWriter = new FileWriter(namaFile,
isAppend)) {
                System.out.print("Tuliskan isi baru untuk file
tersebut: ");
                String isiBaru = input.nextLine();

                fileWriter.write(isiBaru + "\n");
```

```
                System.out.println("Isi File " + namaFile + "
berhasil di-update.\n");
            } catch (Exception err) {
                System.out.println("Error Menulis File: " +
err.getMessage() + "\n");
            }

        } else {
            System.out.println("File " + namaFile + " tidak
ditemukan!\n");
        }
    }
}
```

BAB III
PENUTUP

3.1 Kesimpulan

Dari perancangan sistem berbasis Java untuk pengelolaan file, diperoleh solusi yang dapat mempermudah proses pembuatan, pembacaan, pembaruan, dan penghapusan data dalam file secara lebih efisien. Sistem ini memungkinkan pengguna untuk mengelola file teks dengan lebih terstruktur serta meningkatkan akurasi dalam penyimpanan informasi.

Dengan implementasi metode seperti FileWriter, FileReader, BufferedReader, dan BufferedWriter, sistem mampu menangani berbagai operasi file dengan baik. Penggunaan konsep Exception Handling juga memastikan bahwa kesalahan dalam pengelolaan file dapat diminimalisir. Selain itu, struktur program yang berbasis Object-Oriented Programming (OOP) membantu dalam modularisasi kode, sehingga lebih mudah untuk dikembangkan dan dipelihara.

3.2 Saran

Untuk meningkatkan sistem pengelolaan file berbasis Java ini, beberapa saran berikut dapat dipertimbangkan:

a. Penambahan fitur antarmuka grafis (GUI), agar pengguna dapat mengelola file dengan lebih interaktif dan tidak hanya melalui terminal atau command line.
b. Pengembangan sistem berbasis database, untuk menyimpan informasi dalam skala yang lebih besar dan memungkinkan pencarian serta manipulasi data secara lebih cepat dan terstruktur.
c. Integrasi dengan sistem penyimpanan cloud, agar file yang dikelola dapat diakses dari berbagai perangkat secara lebih fleksibel.
d. Peningkatan mekanisme keamanan, seperti enkripsi file, guna memastikan data yang disimpan tetap aman dari akses yang tidak sah.

Dengan pengembangan lebih lanjut, sistem ini dapat semakin optimal dalam membantu pengguna mengelola file secara efisien dan aman.