**LAPORAN PRAKTIKUM**

**PEMROGRAMAN OBJEK 2**

**MODUL 8**

**APLIKASI BERBASIS TEKS**

**DISUSUN OLEH :**

**FATHIR AHMAD NURPADLI - 2250081132**

Logo

Description automatically generated

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS SAINS DAN INFORMATIKA**

**UNIVERSITAS JENDERAL ACHMAD YANI**

**TAHUN 2024**

# DAFTAR ISI

[DAFTAR ISI i](#_Toc169542279)

[DAFTAR GAMBAR iv](#_Toc169542280)

[BAB I. HASIL PRAKTIKUM 1](#_Toc169542281)

[I.1 Program StandartInput 1](#_Toc169542282)

[I.1.A. Source Code 1](#_Toc169542283)

[I.1.B. Screenshot 1](#_Toc169542284)

[I.1.C. Analisa 2](#_Toc169542285)

[I.2 Program Input 2](#_Toc169542286)

[I.2.A. Source Code 2](#_Toc169542287)

[I.2.B. Screenshot 3](#_Toc169542288)

[I.2.C. Analisa 3](#_Toc169542289)

[I.3 Program Addition 3](#_Toc169542290)

[I.3.A. Source Code 3](#_Toc169542291)

[I.3.B. Screenshot 4](#_Toc169542292)

[I.3.C. Analisa 4](#_Toc169542293)

[I.4 Program GreetUser 5](#_Toc169542294)

[I.4.A. Source Code 5](#_Toc169542295)

[I.4.B. Screenshot 6](#_Toc169542296)

[I.4.C. Analisa 6](#_Toc169542297)

[I.5 Program ReadFile 6](#_Toc169542298)

[I.5.A. Source Code 6](#_Toc169542299)

[I.5.B. Screenshot 8](#_Toc169542300)

[I.5.C. Analisa 8](#_Toc169542301)

[I.6 Program WriteFile 9](#_Toc169542302)

[I.6.A. Source Code 9](#_Toc169542303)

[I.6.B. Screenshot 11](#_Toc169542304)

[I.6.C. Analisa 11](#_Toc169542305)

[I.7 Program FileReading 12](#_Toc169542306)

[I.7.A. Source Code 12](#_Toc169542307)

[I.7.B. Screenshot 13](#_Toc169542308)

[I.7.C. Analisa 13](#_Toc169542309)

[I.8 Program FileAndDirecoryInfo 13](#_Toc169542310)

[I.8.A. Source Code 13](#_Toc169542311)

[I.8.B. Screenshot 15](#_Toc169542312)

[I.8.C. Analisa 16](#_Toc169542313)

[I.9 Program CreateTextFile 16](#_Toc169542314)

[I.9.A. Source Code 16](#_Toc169542315)

[I.9.B. Screenshot 19](#_Toc169542316)

[I.9.C. Analisa 19](#_Toc169542317)

[I.10 Program ReadTextFile 20](#_Toc169542318)

[I.10.A. Source Code 20](#_Toc169542319)

[I.10.B. Screenshot 22](#_Toc169542320)

[I.10.C. Analisa 22](#_Toc169542321)

[BAB II. TUGAS PRAKTIKUM 23](#_Toc169542322)

[II.1 Tugas Akhir 23](#_Toc169542323)

[II.1.A. Source Code 23](#_Toc169542324)

[II.1.B. Screenshot 27](#_Toc169542325)

[II.1.C. Analisa 27](#_Toc169542326)

[BAB III. KESIMPULAN 29](#_Toc169542327)

# DAFTAR GAMBAR

[Gambar 1.1 Hasil Run Program StandartInput 2](#_Toc169542268)

[Gambar 2.1 Hasil Run dari Program Input 3](#_Toc169542269)

[Gambar 3.1 Hasil Run dari Program Addition 4](#_Toc169542270)

[Gambar 4.1 Hasil Run dari Program GreetUser 6](#_Toc169542271)

[Gambar 5.1 Hasil Run dari Program ReadFile 8](#_Toc169542272)

[Gambar 6.1 Hasil Run dari Program WriteFile 11](#_Toc169542273)

[Gambar 7.1 Hasil Run dari Program FileReading 13](#_Toc169542274)

[Gambar 8.1 Hasil Run dari Program FileAndDirectoryInfo 15](#_Toc169542275)

[Gambar 9.1 Hasil Run dari Program CreateTextFile 19](#_Toc169542276)

[Gambar 10.1 Hasil Run dari Program ReadTextFile 22](#_Toc169542277)

[Gambar 11.1 Hasil Run dari Program FileReadWriteGUI 27](#_Toc169542278)

# HASIL PRAKTIKUM

## Program StandartInput

### Source Code

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.IOException;

public class StandartInput {

public static void main(String[] args) {

System.out.println("Hi, What's your favorite character?");

char favChar;

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

try {

favChar = (char) br.read();

System.out.println(favChar + " is a good choice!");

} catch (IOException e) {

System.out.println("An error occurred while reading input.");

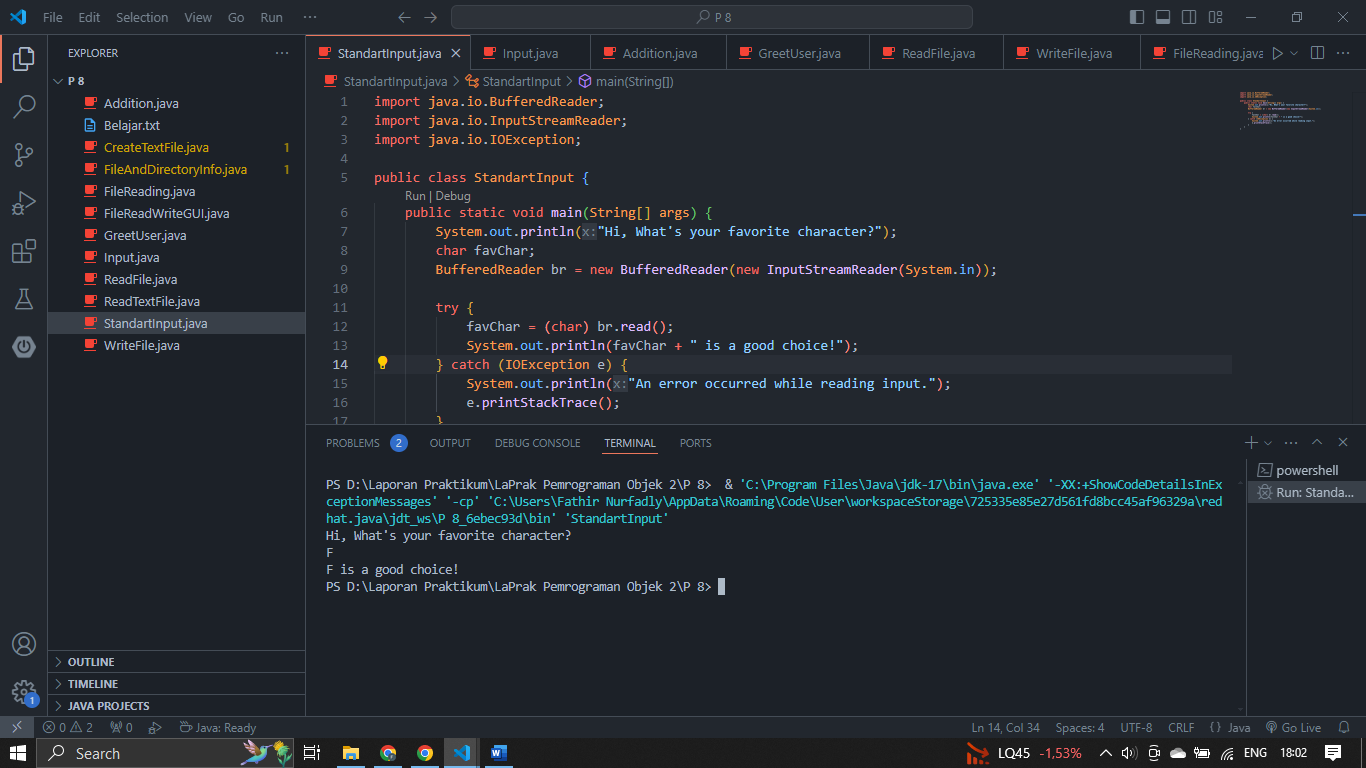
e.printStackTrace();

}

}

}

### Screenshot



Gambar 1.1 Hasil Run Program StandartInput

### Analisa

Program di atas adalah sebuah program Java yang meminta pengguna untuk memasukkan karakter favorit mereka melalui input standar (keyboard) dan kemudian menampilkan karakter tersebut sebagai respons. Program dimulai dengan mendefinisikan kelas `StandartInput` dan metode `main` yang merupakan titik awal eksekusi. Pada awalnya, program menampilkan pesan "Hi, What's your favorite character?" ke konsol. Kemudian, menggunakan `BufferedReader` yang dibungkus di sekitar `InputStreamReader` untuk membaca input dari pengguna. Program mencoba membaca satu karakter dengan metode `br.read()`, mengonversi hasil bacaan tersebut menjadi karakter, dan menampilkannya kembali dengan pesan yang mengakui pilihan pengguna. Jika terjadi kesalahan selama pembacaan input, blok `catch` akan menangkap `IOException` dan menampilkan pesan kesalahan beserta jejak stack trace untuk debugging.

## Program Input

### Source Code

import java.io.IOException;

public class Input {

public static void main(String[] args) {

try {

System.out.print("Masukkan Karakter: ");

char i = (char) System.in.read();

System.out.println("Anda memasukkan " + i);

} catch (IOException e) {

System.out.println("Terjadi kesalahan saat membaca input.");

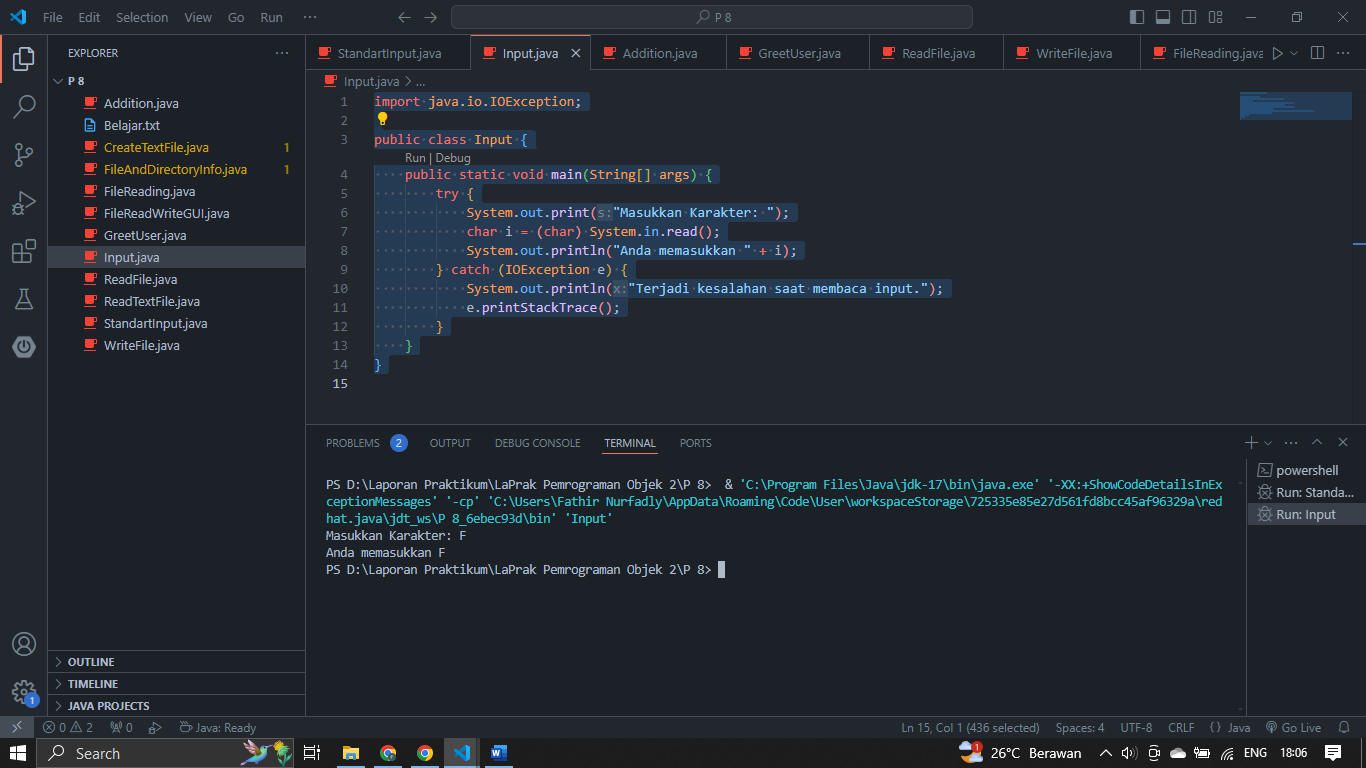
e.printStackTrace();

}

}

}

### Screenshot



Gambar 2.1 Hasil Run dari Program Input

### Analisa

Program di atas meminta pengguna untuk memasukkan satu karakter melalui input standar (keyboard) dan kemudian menampilkan karakter yang dimasukkan. Program dimulai dengan mendefinisikan kelas `Input` dan metode `main`, yang merupakan titik awal eksekusi. Program menampilkan pesan "Masukkan Karakter: " ke konsol untuk meminta input dari pengguna. Selanjutnya, program menggunakan `System.in.read()` untuk membaca satu karakter dari input, mengonversinya menjadi tipe `char`, dan menyimpannya dalam variabel `i`. Karakter yang dimasukkan kemudian ditampilkan kembali ke konsol dengan pesan "Anda memasukkan " diikuti oleh karakter tersebut. Jika terjadi kesalahan selama pembacaan input, blok `catch` akan menangkap `IOException` dan menampilkan pesan kesalahan beserta jejak stack trace untuk membantu debugging.

## Program Addition

### Source Code

import java.util.Scanner;

public class Addition {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

System.out.print("Masukkan nilai integer pertama = ");

int num1 = input.nextInt();

System.out.print("Masukkan nilai integer kedua = ");

int num2 = input.nextInt();

int sum = num1 + num2;

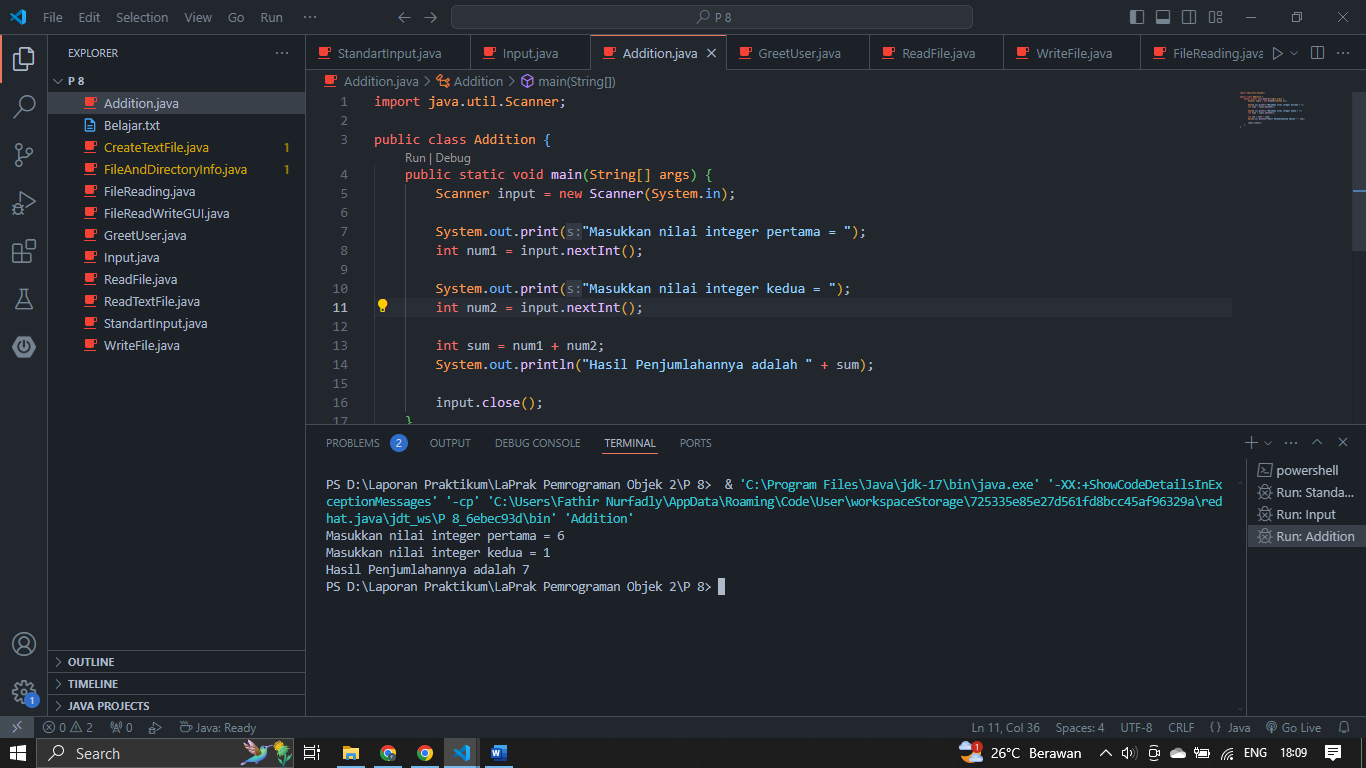
System.out.println("Hasil Penjumlahannya adalah " + sum);

input.close();

}

}

### Screenshot



Gambar 3.1 Hasil Run dari Program Addition

### Analisa

Program di atas melakukan operasi penjumlahan dua bilangan bulat yang dimasukkan oleh pengguna. Program dimulai dengan mendefinisikan kelas `Addition` dan metode `main`, yang merupakan titik awal eksekusi. Di dalam metode `main`, sebuah objek `Scanner` bernama `input` dibuat untuk membaca input dari pengguna. Program meminta pengguna untuk memasukkan dua bilangan bulat dengan menampilkan pesan "Masukkan nilai integer pertama =" dan "Masukkan nilai integer kedua =". Nilai-nilai yang dimasukkan oleh pengguna disimpan dalam variabel `num1` dan `num2` menggunakan metode `input.nextInt()`. Kemudian, program menghitung jumlah kedua bilangan tersebut dan menyimpannya dalam variabel `sum`. Hasil penjumlahan tersebut ditampilkan ke konsol dengan pesan "Hasil Penjumlahannya adalah " diikuti oleh nilai `sum`. Terakhir, `input.close()` dipanggil untuk menutup objek `Scanner` dan membebaskan sumber daya yang digunakan.

## Program GreetUser

### Source Code

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.IOException;

public class GreetUser {

public static void main(String[] args) {

System.out.println("Hi, what's your name?");

String name;

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

try {

name = br.readLine();

System.out.println("Nice to meet you, " + name + "! :)");

} catch (IOException e) {

System.out.println("An error occurred while reading your input.");

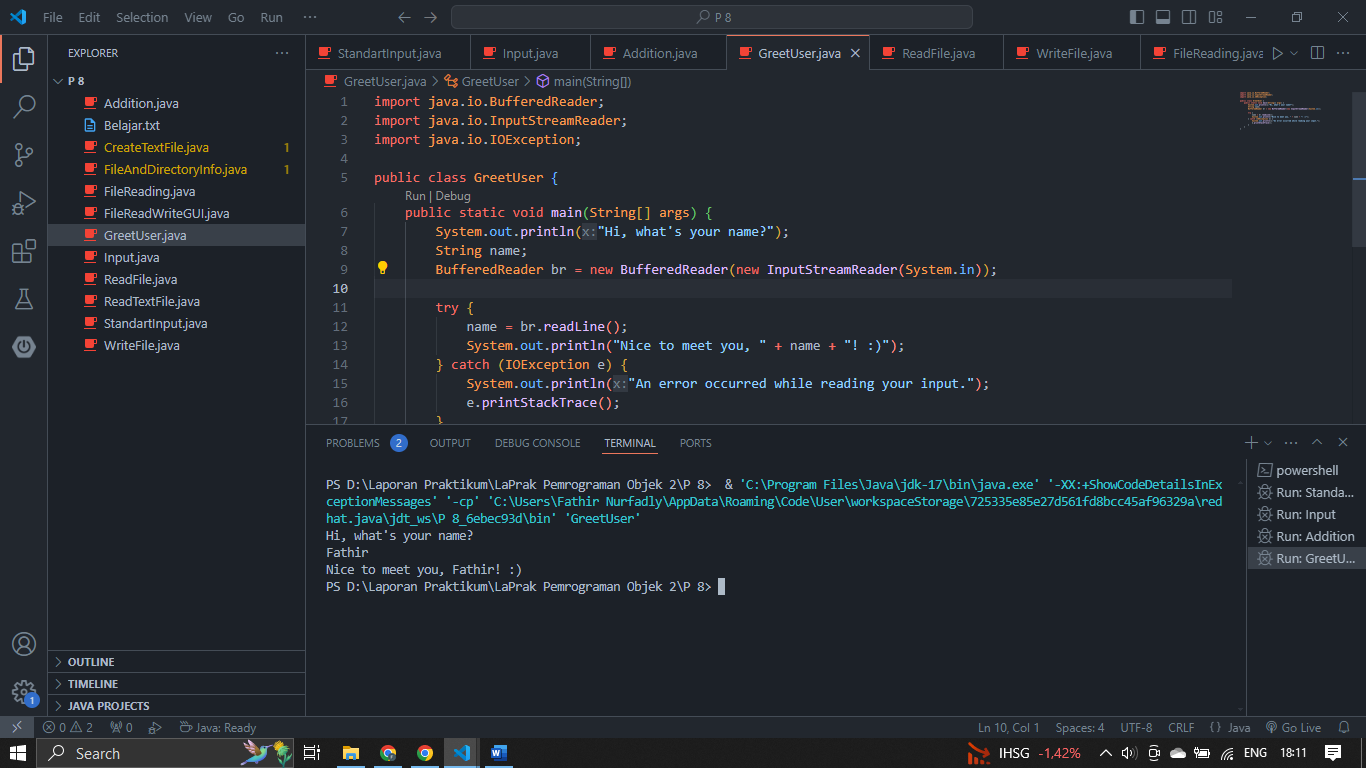
e.printStackTrace();

}

}

}

### Screenshot



Gambar 4.1 Hasil Run dari Program GreetUser

### Analisa

Program di atas meminta pengguna untuk memasukkan nama mereka dan kemudian menampilkan sapaan yang dipersonalisasi. Program dimulai dengan mendefinisikan kelas `GreetUser` dan metode `main`, yang merupakan titik awal eksekusi. Program menampilkan pesan "Hi, what's your name?" ke konsol untuk meminta input dari pengguna. Selanjutnya, sebuah objek `BufferedReader` bernama `br` dibuat, yang dibungkus di sekitar `InputStreamReader` untuk membaca input dari pengguna melalui konsol. Program mencoba membaca satu baris teks yang dimasukkan oleh pengguna menggunakan metode `br.readLine()`, dan menyimpannya dalam variabel `name`. Nama yang dimasukkan kemudian ditampilkan kembali ke konsol dengan pesan "Nice to meet you, " diikuti oleh nama pengguna dan simbol senyum ":)". Jika terjadi kesalahan selama pembacaan input, blok `catch` akan menangkap `IOException` dan menampilkan pesan kesalahan beserta jejak stack trace untuk membantu debugging.

## Program ReadFile

### Source Code

import java.io.BufferedReader;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.InputStreamReader;

public class ReadFile {

public static void main(String[] args) {

System.out.println("What is the name of the file to read from?");

String filename = null;

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

try {

filename = br.readLine();

} catch (IOException e) {

System.out.println("An error occurred while reading your input.");

e.printStackTrace();

return; // Exit if the filename cannot be read

}

System.out.println("Now reading from " + filename + "...");

try (FileInputStream fis = new FileInputStream(filename)) {

int temp;

while ((temp = fis.read()) != -1) {

char data = (char) temp;

System.out.print(data);

}

} catch (FileNotFoundException ex) {

System.out.println("File not found.");

} catch (IOException ex) {

System.out.println("Problem in reading from the file.");

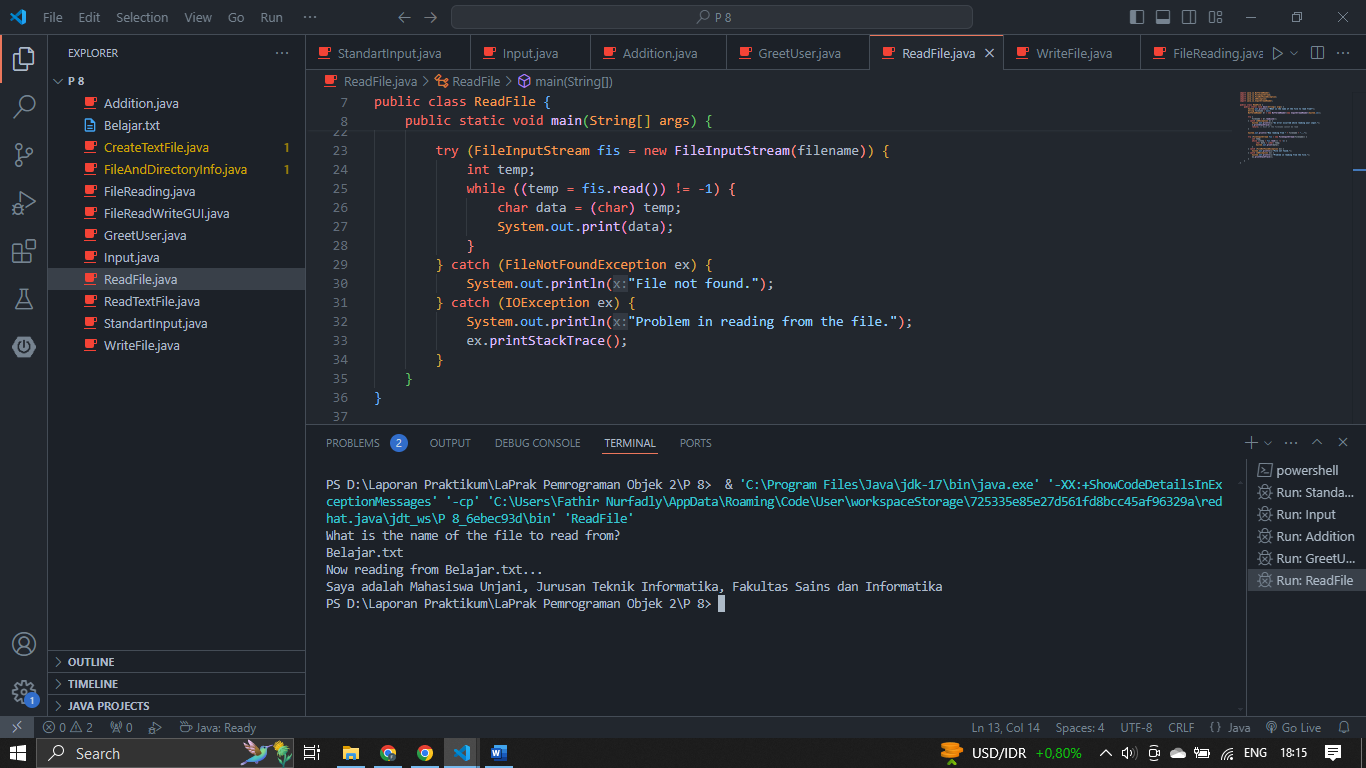
ex.printStackTrace();

}

}

}

### Screenshot



Gambar 5.1 Hasil Run dari Program ReadFile

### Analisa

Program di atas membaca dan menampilkan konten dari sebuah file teks yang namanya dimasukkan oleh pengguna. Program dimulai dengan mendefinisikan kelas `ReadFile` dan metode `main`, yang merupakan titik awal eksekusi. Pertama, program menampilkan pesan "What is the name of the file to read from?" ke konsol untuk meminta input dari pengguna mengenai nama file yang akan dibaca. Sebuah objek `BufferedReader` dibuat untuk membaca input dari pengguna melalui konsol, dan nama file yang dimasukkan disimpan dalam variabel `filename`. Jika terjadi kesalahan saat membaca nama file, blok `catch` akan menangkap `IOException`, menampilkan pesan kesalahan, mencetak jejak stack trace, dan menghentikan eksekusi program. Selanjutnya, program mencoba membuka dan membaca file yang namanya diberikan oleh pengguna menggunakan objek `FileInputStream` dalam blok `try-with-resources` untuk memastikan bahwa stream ditutup secara otomatis. Program membaca file byte demi byte, mengonversi setiap byte menjadi karakter, dan menampilkannya ke konsol hingga mencapai akhir file. Jika file tidak ditemukan, blok `catch` akan menangkap `FileNotFoundException` dan menampilkan pesan "File not found." Jika terjadi kesalahan saat membaca file, blok `catch` lainnya akan menangkap `IOException`, menampilkan pesan kesalahan, dan mencetak jejak stack trace untuk membantu debugging.

## Program WriteFile

### Source Code

import java.io.\*;

public class WriteFile {

public WriteFile() {

}

public static void main(String args[]) throws IOException {

System.out.println("What is the name of the file to be written to?");

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String filename = br.readLine();

System.out.println("Enter data to write to " + filename + "... (end input with 'g$')");

FileOutputStream fos = null;

try {

fos = new FileOutputStream(filename);

} catch (FileNotFoundException ex) {

System.out.println("File cannot be opened for writing.");

return;

}

try {

boolean done = false;

int data;

do {

data = br.read();

if ((char) data == 'g') {

data = br.read();

if ((char) data == '$') {

done = true;

} else {

fos.write('g');

fos.write(data);

}

} else {

fos.write(data);

}

} while (!done);

} catch (IOException ex) {

System.out.println("Problem in reading or writing to the file.");

} finally {

if (fos != null) {

fos.close();

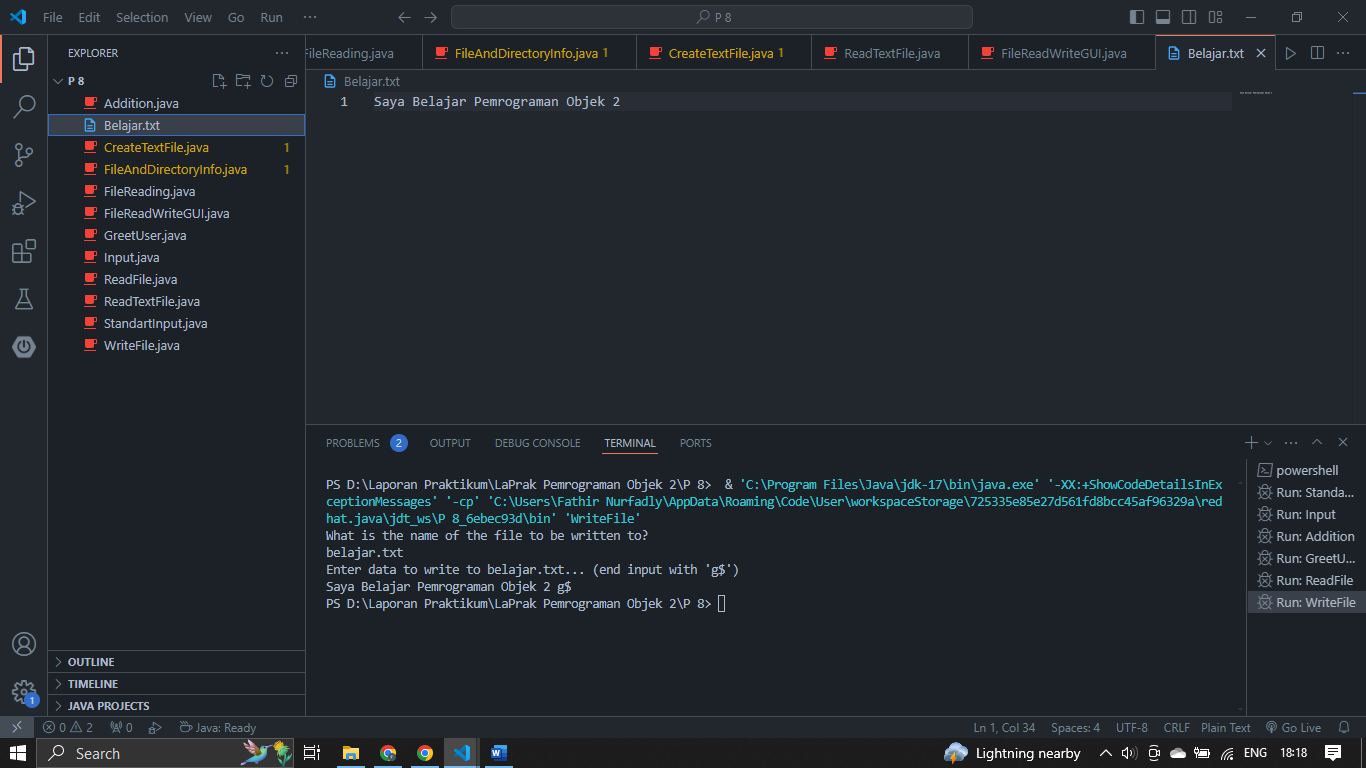
}

}

}

}

### Screenshot



Gambar 6.1 Hasil Run dari Program WriteFile

### Analisa

Program di atas memungkinkan pengguna untuk menulis data ke dalam sebuah file yang nama filenya dimasukkan oleh pengguna melalui konsol. Program dimulai dengan mendefinisikan kelas `WriteFile` dan metode `main`, yang merupakan titik awal eksekusi. Pertama, program menampilkan pesan "What is the name of the file to be written to?" untuk meminta pengguna memasukkan nama file yang akan ditulis. Sebuah objek `BufferedReader` dibuat untuk membaca input dari pengguna melalui konsol, dan nama file yang dimasukkan disimpan dalam variabel `filename`. Program kemudian menampilkan pesan "Enter data to write to " diikuti oleh nama file dan pesan untuk mengakhiri input dengan 'g$'. Selanjutnya, program mencoba membuka file tersebut untuk ditulis menggunakan objek `FileOutputStream`. Jika file tidak dapat dibuka, program menangkap `FileNotFoundException`, menampilkan pesan "File cannot be opened for writing.", dan menghentikan eksekusi dengan menggunakan `return`. Selama proses penulisan data, program membaca input dari pengguna menggunakan `br.read()` untuk membaca karakter demi karakter. Jika karakter yang dibaca adalah 'g$' (yaitu tanda akhir input), program selesai menulis ke file. Jika tidak, program menulis karakter tersebut ke dalam file menggunakan `fos.write(data)`. Jika terjadi kesalahan selama proses membaca atau menulis file, program menangkap `IOException` dan menampilkan pesan "Problem in reading or writing to the file." Pada akhirnya, dalam blok `finally`, program memastikan bahwa objek `FileOutputStream` ditutup untuk membebaskan sumber daya yang digunakan.

## Program FileReading

### Source Code

import java.io.\*;

public class FileReading {

public static void main(String[] args) {

StringBuilder fileContent = new StringBuilder();

try (BufferedReader fileInput = new BufferedReader(new FileReader("D:/Laporan Praktikum/LaPrak Pemrograman Objek 2/P 8/Belajar.txt"))) {

String line;

while ((line = fileInput.readLine()) != null) {

fileContent.append(line).append("\n");

}

} catch (FileNotFoundException fnfe) {

System.out.println("File not found");

System.exit(0);

} catch (IOException ioe) {

System.out.println("Error reading file");

System.exit(0);

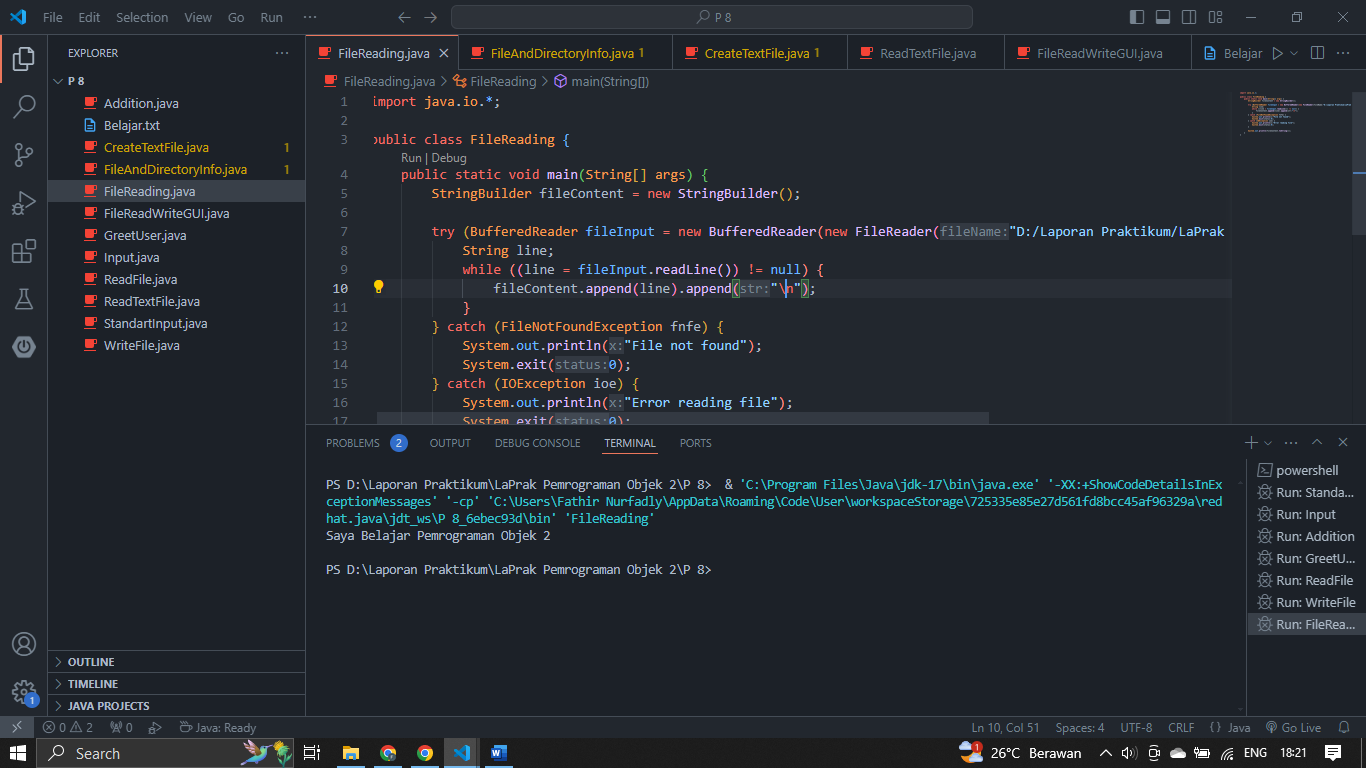
}

System.out.println(fileContent.toString());

}

}

### Screenshot



Gambar 7.1 Hasil Run dari Program FileReading

### Analisa

Program di atas membaca konten dari sebuah file teks yang terletak di path yang telah ditentukan dan kemudian menampilkannya ke konsol. Program dimulai dengan mendefinisikan kelas `FileReading` dan metode `main`, yang merupakan titik awal eksekusi. Sebuah objek `StringBuilder` dengan nama `fileContent` dibuat untuk menyimpan konten file yang akan dibaca. Selanjutnya, program menggunakan blok `try-with-resources` untuk membuka dan membaca file menggunakan objek `BufferedReader` yang dibuat dengan `FileReader`. Selama proses pembacaan, setiap baris dari file dibaca menggunakan `fileInput.readLine()` dan ditambahkan ke dalam `StringBuilder` `fileContent`, dengan menambahkan karakter newline (`\n`) setiap kali baris dibaca. Jika file tidak ditemukan, program menangkap `FileNotFoundException`, menampilkan pesan "File not found", dan keluar dari program menggunakan `System.exit(0)`. Jika terjadi kesalahan dalam proses membaca file, program menangkap `IOException`, menampilkan pesan "Error reading file", dan juga keluar dari program. Setelah selesai membaca file, konten yang ada dalam `StringBuilder` `fileContent` ditampilkan ke konsol menggunakan `System.out.println(fileContent.toString())`.

## Program FileAndDirecoryInfo

### Source Code

import java.io.IOException;

import java.nio.file.DirectoryIteratorException;

import java.nio.file.DirectoryStream;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.util.Scanner;

public class FileAndDirectoryInfo {

public static void main(String[] args) throws IOException {

Scanner input = new Scanner(System.in);

System.out.println("Enter file or directory name:");

Path path = Paths.get(input.nextLine());

if (Files.exists(path)) {

System.out.printf("%n%s exists%n", path.getFileName());

System.out.printf("%s a directory%n", Files.isDirectory(path) ? "Is" : "Is not");

System.out.printf("%s an absolute path%n", path.isAbsolute() ? "Is" : "Is not");

System.out.printf("Last modified: %s%n", Files.getLastModifiedTime(path));

System.out.printf("Size: %s bytes%n", Files.size(path));

System.out.printf("Path: %s%n", path);

System.out.printf("Absolute path: %s%n", path.toAbsolutePath());

if (Files.isDirectory(path)) {

System.out.printf("%nDirectory contents:%n");

try (DirectoryStream<Path> directoryStream = Files.newDirectoryStream(path)) {

for (Path p : directoryStream) {

System.out.println(p.getFileName());

}

} catch (IOException | DirectoryIteratorException e) {

System.err.println("Error reading directory contents: " + e.getMessage());

}

}

} else {

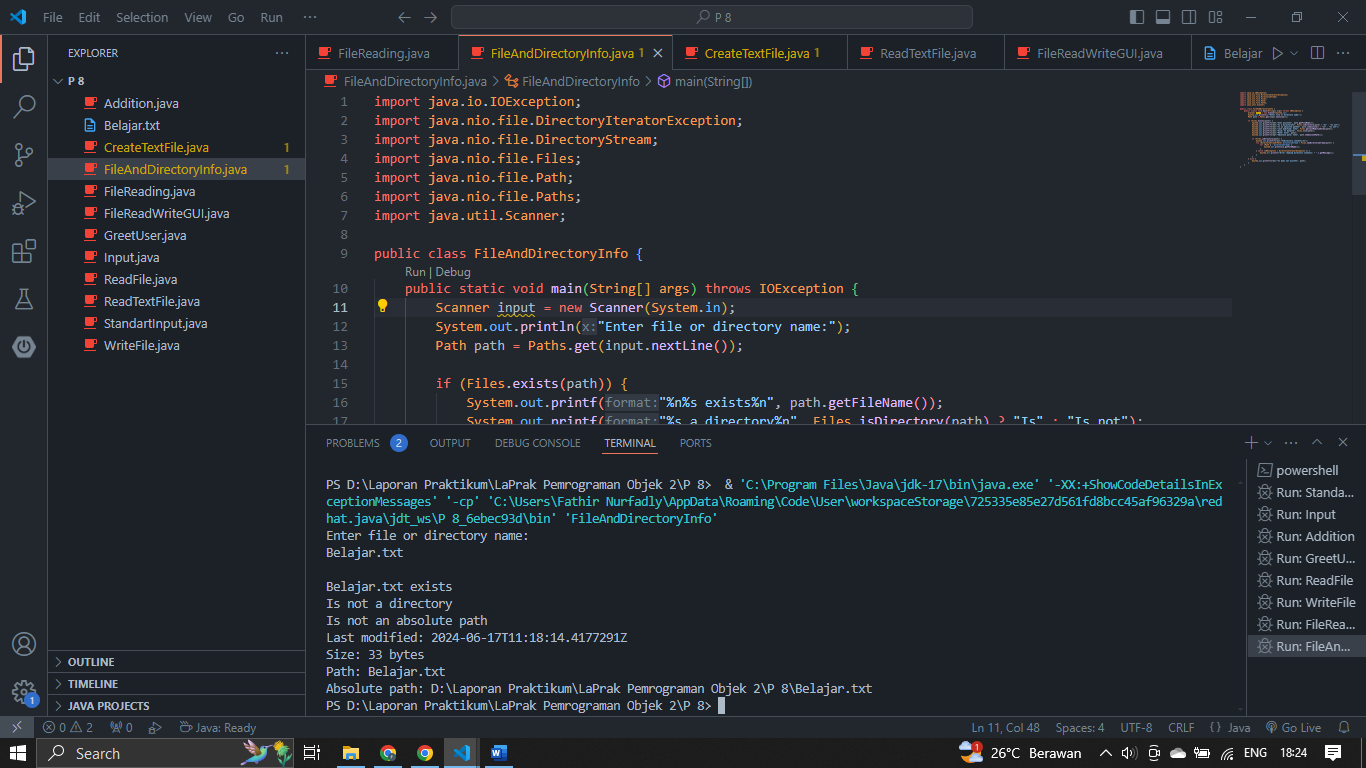
System.out.printf("%s does not exist%n", path);

}

}

}

### Screenshot



Gambar 8.1 Hasil Run dari Program FileAndDirectoryInfo

### Analisa

Program di atas memungkinkan pengguna untuk memasukkan nama file atau direktori, dan kemudian memberikan informasi detail tentang entitas tersebut. Program dimulai dengan mendefinisikan kelas `FileAndDirectoryInfo` dan metode `main`, yang merupakan titik awal eksekusi. Sebuah objek `Scanner` dibuat untuk menerima input dari pengguna melalui konsol dengan pesan "Enter file or directory name:". Nama yang dimasukkan oleh pengguna diambil dan diubah menjadi objek `Path` menggunakan `Paths.get(input.nextLine())`. Program kemudian memeriksa apakah entitas yang ditunjukkan oleh `Path` tersebut ada atau tidak dengan menggunakan `Files.exists(path)`. Jika entitas tersebut ada, program mencetak informasi sebagai berikut: nama file/direktori, apakah itu direktori atau bukan, apakah itu path absolut atau tidak, waktu modifikasi terakhir, ukuran file (dalam byte), path, dan path absolut. Jika entitas yang dimasukkan adalah direktori, program juga mencetak isi direktori tersebut dengan menggunakan `DirectoryStream` dan mencetak nama setiap entitas di dalamnya. Jika terjadi kesalahan dalam membaca direktori, program menangkap `IOException` atau `DirectoryIteratorException` dan mencetak pesan kesalahan yang sesuai. Jika entitas yang dimasukkan tidak ada, program mencetak pesan bahwa entitas tersebut tidak ada.

## Program CreateTextFile

### Source Code

import java.io.FileNotFoundException;

import java.lang.SecurityException;

import java.util.Formatter;

import java.util.FormatterClosedException;

import java.util.NoSuchElementException;

import java.util.Scanner;

public class CreateTextFile {

private static Formatter output;

public static void main(String[] args) {

openFile();

addRecords();

closeFile();

}

// open file

public static void openFile() {

try {

output = new Formatter("clients.txt"); // open the file

} catch (SecurityException securityException) {

System.err.println("Write permission denied. Terminating.");

System.exit(1); // terminate the program

} catch (FileNotFoundException fileNotFoundException) {

System.err.println("Error opening file. Terminating.");

System.exit(1); // terminate the program

}

}

// add records to file

public static void addRecords() {

Scanner input = new Scanner(System.in);

System.out.printf("%s%n%s%n? ", "Enter account number, first name, last name and balance", "Enter end-of-file indicator to end input.");

while (input.hasNext()) {

try {

// output new record to file; assumes valid input

output.format("%d %s %s %.2f%n", input.nextInt(), input.next(), input.next(), input.nextDouble());

} catch (FormatterClosedException formatterClosedException) {

System.err.println("Error writing to file. Terminating.");

break;

} catch (NoSuchElementException noSuchElementException) {

System.err.println("Invalid input. Please try again.");

input.nextLine(); // discard input so user can try again

}

System.out.print("? ");

}

}

// close file

public static void closeFile() {

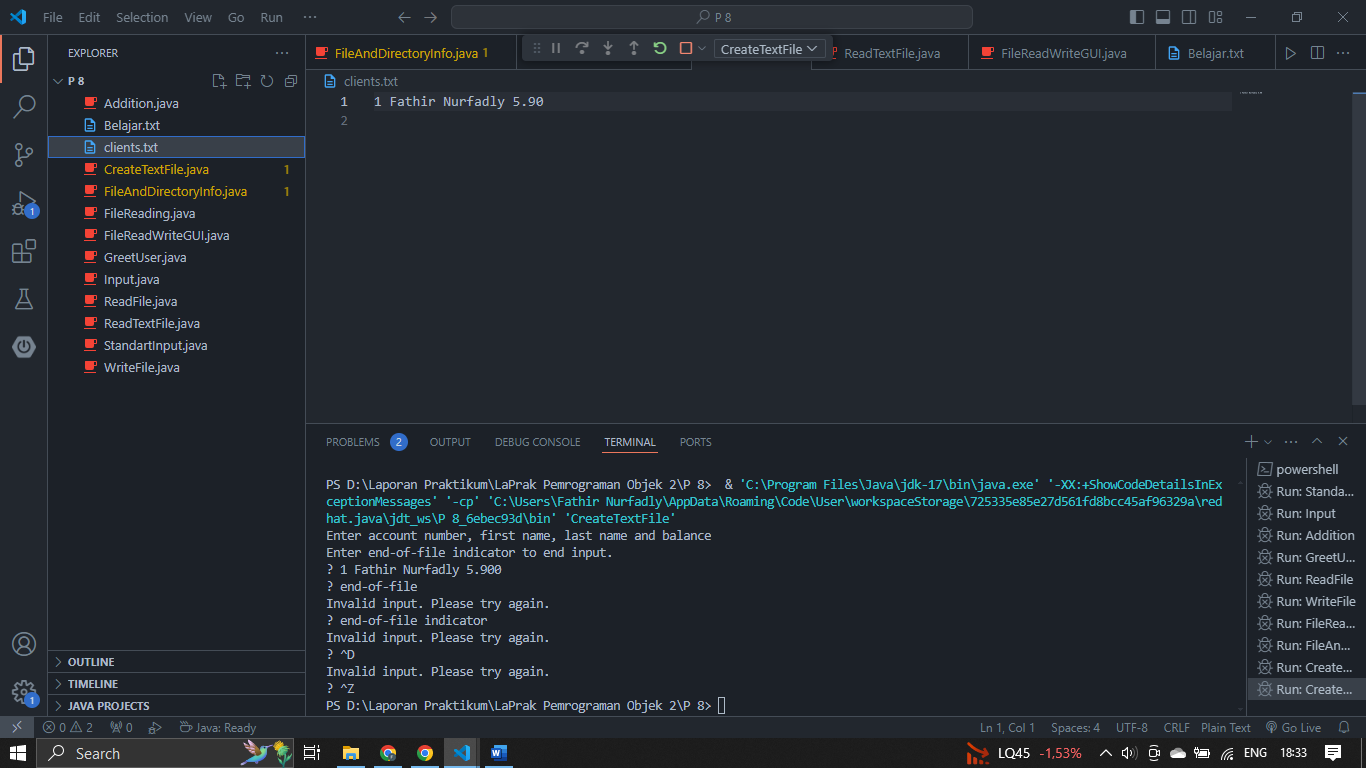
if (output != null)

output.close();

}

}

### Screenshot



Gambar 9.1 Hasil Run dari Program CreateTextFile

### Analisa

Program di atas bertujuan untuk membuat dan mengisi sebuah file teks `clients.txt` dengan data yang dimasukkan oleh pengguna melalui konsol. Program dimulai dengan mendefinisikan kelas `CreateTextFile` dan metode `main`, yang merupakan titik awal eksekusi. Metode `main` memanggil metode `openFile` untuk membuka file `clients.txt`, `addRecords` untuk menambahkan catatan ke file, dan `closeFile` untuk menutup file setelah selesai ditulis. Metode `openFile` mencoba membuka file menggunakan objek `Formatter`, menangkap `SecurityException` jika izin menulis ditolak atau `FileNotFoundException` jika file tidak ditemukan, dan mengakhiri program jika terjadi kesalahan. Metode `addRecords` menggunakan objek `Scanner` untuk menerima input dari pengguna melalui konsol. Program meminta pengguna untuk memasukkan nomor akun, nama depan, nama belakang, dan saldo dengan pesan yang sesuai. Selama input masih tersedia (`input.hasNext()`), program mencoba menulis data ke file menggunakan `output.format()` dengan format tertentu. Jika terjadi `FormatterClosedException`, program mencetak pesan kesalahan dan menghentikan penulisan. Jika terjadi `NoSuchElementException`, program mencetak pesan bahwa input tidak valid dan mengabaikan input yang salah dengan memanggil `input.nextLine()`. Metode `closeFile` memastikan bahwa file ditutup dengan memeriksa apakah `output` tidak null sebelum memanggil `output.close()`.

## Program ReadTextFile

### Source Code

import java.io.IOException;

import java.lang.IllegalStateException;

import java.nio.file.Paths;

import java.util.NoSuchElementException;

import java.util.Scanner;

public class ReadTextFile{

private static Scanner input;

public static void main (String [] args){

openFile();

readRecords();

closeFile();

}

//openFile client.txt

public static void openFile(){

try{

input = new Scanner(Paths.get("client.txt"));

}catch (IOException ioException){

System.err.println("Error opening file. Terminating.");

System.exit(1);

}

}

//readRecords from file

public static void readRecords(){

System.out.printf("%-10s%-12s%-12s%10s%n", "Account", "First Name", "Last Name", "Balance");

try{

while(input.hasNext()) //while there is more to read

{

// display record contens

System.out.printf("%-10d%-12s%-12s%10.2f%n", input.nextInt(), input.next(), input.next(), input.nextDouble());

}

}catch(NoSuchElementException elementException){

System.err.println("File Improperly formed. Terminating");

}catch (IllegalStateException stateException){

System.err.println("Error reading from file. Terminating.");

} //end method readRecords

}

// close file and terminate application

public static void closeFile(){

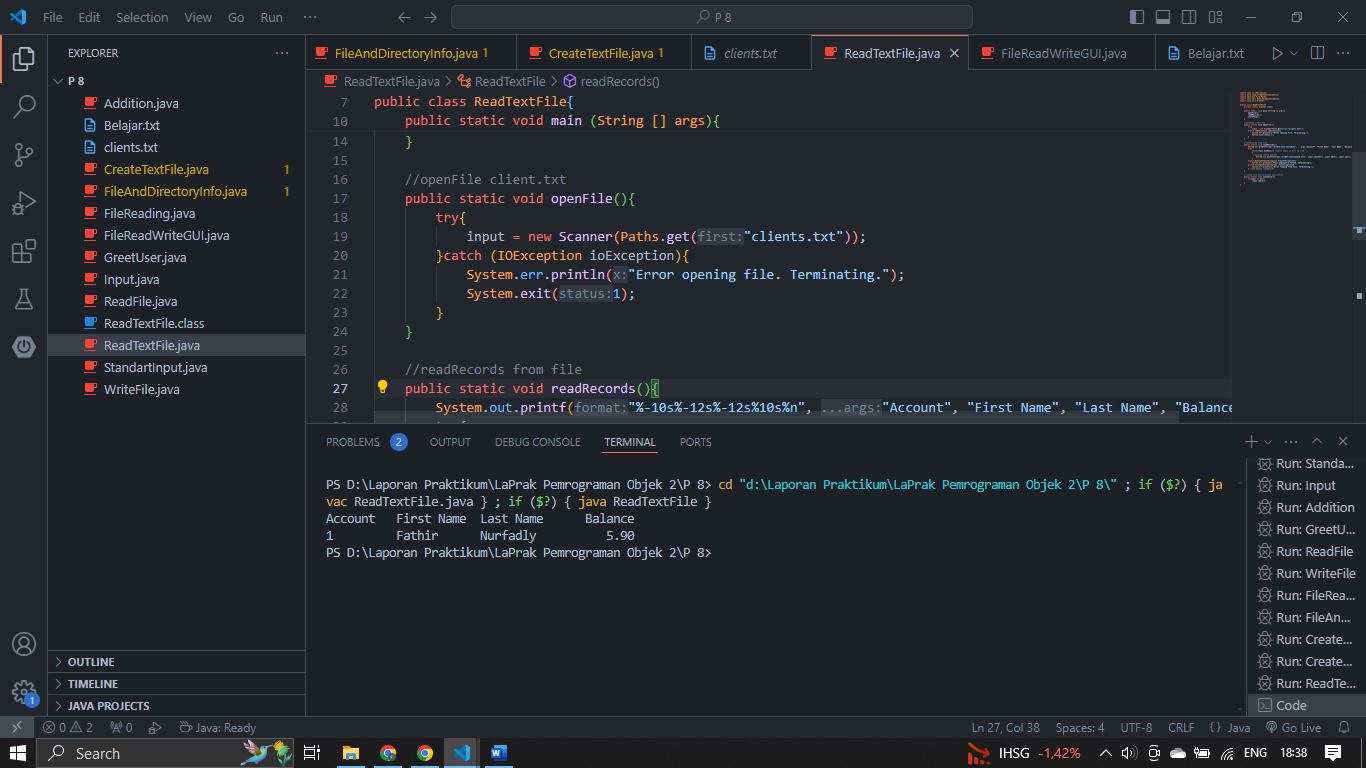
if(input != null)

input.close();

}

}

### Screenshot



Gambar 10.1 Hasil Run dari Program ReadTextFile

### Analisa

Program di atas bertujuan untuk membaca dan menampilkan konten dari file teks `client.txt`, yang berisi catatan akun dengan format yang telah ditentukan. Program dimulai dengan mendefinisikan kelas `ReadTextFile` dan metode `main`, yang merupakan titik awal eksekusi. Metode `main` memanggil `openFile` untuk membuka file `client.txt`, `readRecords` untuk membaca dan menampilkan catatan dari file, dan `closeFile` untuk menutup file setelah selesai dibaca. Metode `openFile` menggunakan `Scanner` untuk membaca file yang diberikan melalui `Paths.get("client.txt")`. Jika terjadi `IOException`, program mencetak pesan kesalahan "Error opening file. Terminating." dan menghentikan eksekusi program. Metode `readRecords` mencetak header tabel dengan format yang telah ditentukan (`%-10s%-12s%-12s%10s%n`) untuk nomor akun, nama depan, nama belakang, dan saldo. Selama ada data yang dapat dibaca (`input.hasNext()`), program mencoba membaca dan mencetak setiap baris data dengan format yang sesuai (`%-10d%-12s%-12s%10.2f%n`). Jika terjadi `NoSuchElementException`, program mencetak pesan "File Improperly formed. Terminating." Jika terjadi `IllegalStateException`, program mencetak pesan "Error reading from file. Terminating." Metode `closeFile` memastikan bahwa file ditutup dengan memeriksa apakah `input` tidak null sebelum memanggil `input.close()`.

# TUGAS PRAKTIKUM

## Tugas Akhir

### Source Code

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.\*;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

public class FileReadWriteGUI extends JFrame {

private JTextArea textArea;

private JButton readButton;

private JButton writeButton;

private JTextField filePathField;

public FileReadWriteGUI() {

super("File Read/Write Program");

setLayout(new BorderLayout());

textArea = new JTextArea(20, 50);

textArea.setLineWrap(true);

textArea.setWrapStyleWord(true);

JScrollPane scrollPane = new JScrollPane(textArea);

JPanel panel = new JPanel();

panel.setLayout(new FlowLayout());

JLabel filePathLabel = new JLabel("File Path:");

filePathField = new JTextField(30);

readButton = new JButton("Read File");

writeButton = new JButton("Write File");

panel.add(filePathLabel);

panel.add(filePathField);

panel.add(readButton);

panel.add(writeButton);

add(panel, BorderLayout.NORTH);

add(scrollPane, BorderLayout.CENTER);

readButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

readFile();

}

});

writeButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

writeFile();

}

});

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

pack();

setLocationRelativeTo(null);

setVisible(true);

}

private void readFile() {

String filePath = filePathField.getText();

Path path = Paths.get(filePath);

if (Files.exists(path)) {

try (BufferedReader reader = Files.newBufferedReader(path)) {

textArea.setText("");

String line;

while ((line = reader.readLine()) != null) {

textArea.append(line + "\n");

}

} catch (IOException ex) {

JOptionPane.showMessageDialog(this, "Error reading file: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

} else {

JOptionPane.showMessageDialog(this, "File not found", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void writeFile() {

String filePath = filePathField.getText();

Path path = Paths.get(filePath);

try (BufferedWriter writer = Files.newBufferedWriter(path)) {

writer.write(textArea.getText());

JOptionPane.showMessageDialog(this, "File written successfully", "Success", JOptionPane.INFORMATION\_MESSAGE);

} catch (IOException ex) {

JOptionPane.showMessageDialog(this, "Error writing file: " + ex.getMessage(), "Error", JOptionPane.ERROR\_MESSAGE);

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

public void run() {

new FileReadWriteGUI();

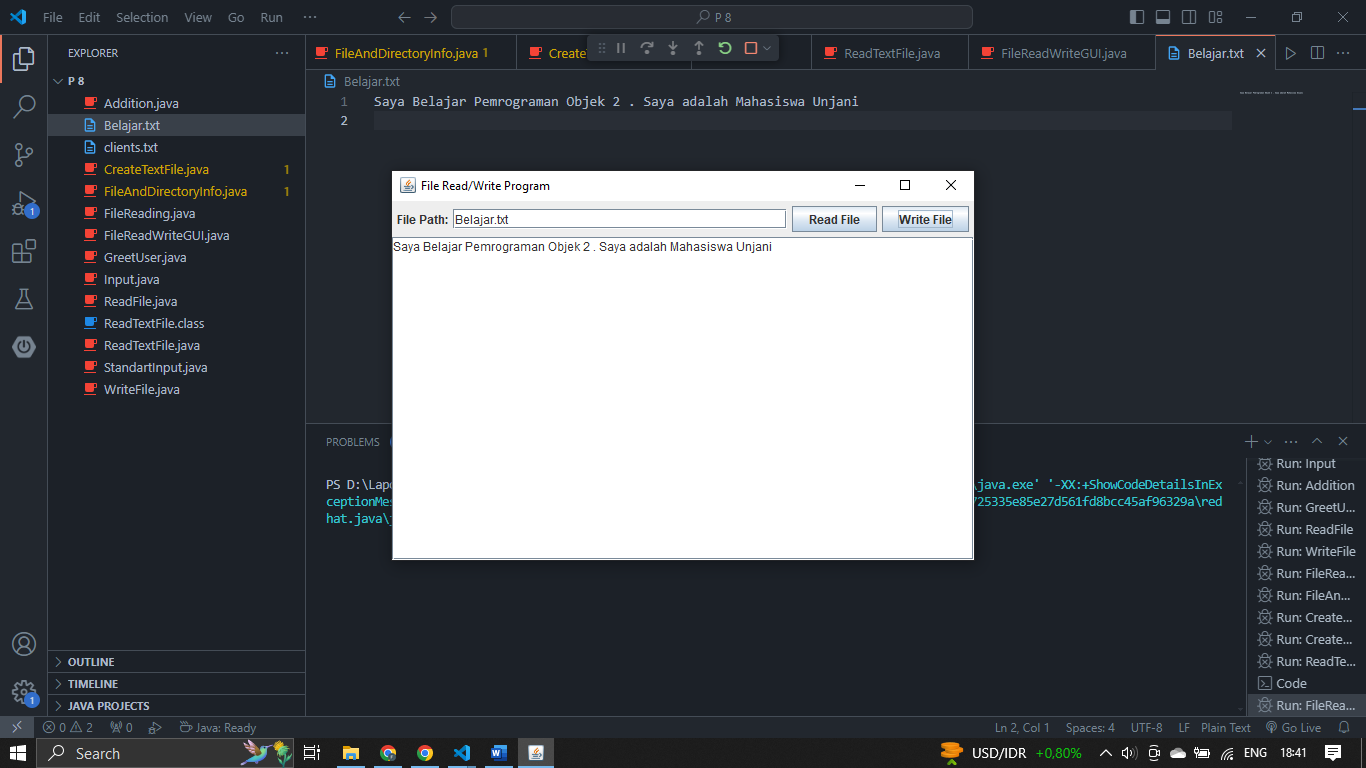
}

});

}

}

### Screenshot



Gambar 11.1 Hasil Run dari Program FileReadWriteGUI

### Analisa

Program di atas merupakan aplikasi GUI sederhana menggunakan Java Swing untuk melakukan operasi membaca dan menulis file teks. Kelas ini mewarisi `JFrame` dan memiliki komponen-komponen seperti `JTextArea` untuk menampilkan teks file, `JButton` untuk membaca dan menulis file, serta `JTextField` untuk memasukkan path file yang akan dioperasikan. Saat aplikasi dijalankan, ia menampilkan sebuah jendela dengan judul "File Read/Write Program", yang memiliki `JPanel` untuk mengatur komponen-komponen GUI di bagian atas (NORTH) dan `JScrollPane` untuk `JTextArea` di bagian tengah (CENTER). Tombol "Read File" dan "Write File" memiliki event listener yang menghubungkan mereka dengan metode `readFile()` dan `writeFile()` masing-masing, yang diimplementasikan untuk membaca dan menulis file. Metode `readFile()` menggunakan `Files.newBufferedReader(path)` untuk membaca setiap baris dari file yang ditunjukkan oleh path yang dimasukkan oleh pengguna, sementara `writeFile()` menggunakan `Files.newBufferedWriter(path)` untuk menulis konten dari `JTextArea` ke dalam file yang ditunjukkan oleh path yang dimasukkan oleh pengguna. Saat terjadi kesalahan dalam operasi file, seperti file tidak ditemukan atau kesalahan I/O, program menampilkan dialog pesan error menggunakan `JOptionPane`. Program ini diakhiri dengan memanggil `setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE)` untuk menutup aplikasi saat jendela utama ditutup, dan `setVisible(true)` untuk menampilkan jendela ke pengguna setelah semua komponen diatur dengan benar.

# KESIMPULAN

Aplikasi berbasis teks pada Java meliputi penggunaan berbagai teknik untuk membaca, menulis, dan memanipulasi file teks menggunakan kelas-kelas seperti `File`, `FileReader`, `FileWriter`, `BufferedReader`, dan `BufferedWriter`. Aplikasi semacam ini biasanya berfokus pada interaksi dengan pengguna melalui konsol atau antarmuka pengguna sederhana (GUI), dengan tujuan melakukan operasi dasar seperti membaca data dari file, menuliskan data ke dalam file, serta melakukan operasi manipulasi dan analisis teks. Pentingnya pengelolaan eksepsi (exceptions handling) dalam pengoperasian file menjadi bagian krusial untuk mengatasi kemungkinan kesalahan saat membaca atau menulis file, sehingga aplikasi dapat memberikan respons yang sesuai kepada pengguna dalam hal kegagalan operasi file. Teknik-teknik ini menjadi dasar penting dalam pengembangan aplikasi yang membutuhkan manajemen data berbasis teks, baik itu untuk keperluan pengolahan data sederhana maupun sebagai bagian dari sistem yang lebih kompleks dalam lingkungan pengembangan perangkat lunak dengan Java.