POSTTEST 6

PEMPROGRAMAN BERORIENTASI OBJEK

Nama : Luthfiah Nur Alifah

NIM 2309106102

Kelas : C1’23

SS Program :

1. Menu Utama

```
========== APLIKASI JASA TITIP MIE GACOAN ==========
1. Tambah Reseller
2. Tambah Menu
3. Tambah Pelanggan
4. Buat Pesanan
5. Lihat Semua Data
6. Keluar
Pilih menu:
```

2. Menu Tambah Reseller

```
========== APLIKASI JASA TITIP MIE GACOAN ==========
1. Tambah Reseller
2. Tambah Menu
3. Tambah Pelanggan
4. Buat Pesanan
5. Lihat Semua Data
6. Keluar
Pilih menu: 1
Masukkan Nama: Luthfi
Masukkan Alamat: Jl.Cendana
Masukkan Telepon: 081234567890
Masukkan Email: luthfi@gmail.com
Reseller berhasil ditambahkan!
```

3. Tambah Menu

```
========== APLIKASI JASA TITIP MIE GACOAN ==========
1. Tambah Reseller
2. Tambah Menu
3. Tambah Pelanggan
4. Buat Pesanan
5. Lihat Semua Data
6. Keluar
Pilih menu: 2
Masukkan Nama Menu: Lumpia Udang
Masukkan Jenis Menu: Dimsum
Masukkan Harga: 15000
Masukkan Stok: 13
Menu berhasil ditambahkan!
```

4. Tambah Pelanggan

```
========== APLIKASI JASA TITIP MIE GACOAN ==========
1. Tambah Reseller
2. Tambah Menu
3. Tambah Pelanggan
4. Buat Pesanan
5. Lihat Semua Data
6. Keluar
Pilih menu: 3
Masukkan Nama: Nur
Masukkan Alamat: Jl.Antasari
Masukkan Telepon: 081212121212
Masukkan Email: nur@gmail.com
Pelanggan berhasil ditambahkan!
```

5. Buat Pesanan

```
========== APLIKASI JASA TITIP MIE GACOAN ==========
1. Tambah Reseller
2. Tambah Menu
3. Tambah Pelanggan
4. Buat Pesanan
5. Lihat Semua Data
6. Keluar
Pilih menu: 4

--- Daftar Pelanggan ---
Pelanggan - ID: 3, Nama: Nur, Alamat: Jl.Antasari, Telepon: 081212121212, Email: nur@gmail.com
Masukkan ID Pelanggan: 3

--- Daftar Menu ---
ID: 2, Nama: Lumpia Udang, Jenis: Dimsum, Harga: 15000.0, Stok: 13
Masukkan ID Menu: 2

--- Daftar Reseller ---
Reseller - ID: 1, Nama: Luthfi, Alamat: Jl.Cendana, Telepon: 081234567890, Email: luthfi@gmail.com
Masukkan ID Reseller: 1
Masukkan Jumlah: 1
Masukkan Status (contoh: Diproses, Dikirim, Selesai): Selesai
Pesanan berhasil dibuat!
Detail: Pelanggan Nur memesan 1 Lumpia Udang melalui reseller Luthfi
```

6. Lihat Semua Data

```
========== DATA JASA TITIP ==========

--- DAFTAR RESELLER ---
Reseller - ID: 1, Nama: Luthfi, Alamat: Jl.Cendana, Telepon: 081234567890, Email: luthfi@gmail.com

--- DAFTAR PELANGGAN ---
Pelanggan - ID: 3, Nama: Nur, Alamat: Jl.Antasari, Telepon: 081212121212, Email: nur@gmail.com

--- DAFTAR MENU ---
ID: 2, Nama: Lumpia Udang, Jenis: Dimsum, Harga: 15000.0, Stok: 12

--- DAFTAR PESANAN ---
ID Pesanan: 4, Pelanggan: 3, Menu: 2, Reseller: 1, Jumlah: 1, Status: Selesai, Tanggal: Wed Apr 16 22:45:11 SGT 2025
   Detail: Pelanggan: Nur, Menu: Lumpia Udang, Reseller: Luthfi
```

7. Keluar

```
========== APLIKASI JASA TITIP MIE GACOAN ==========
1. Tambah Reseller
2. Tambah Menu
3. Tambah Pelanggan
4. Buat Pesanan
5. Lihat Semua Data
6. Keluar
Pilih menu: 6
Terima kasih telah menggunakan aplikasi JasaTitip!
```

Source Code

```
import java.util.ArrayList;

import java.util.Date;

import java.util.Scanner;


// Interface untuk notifikasi dengan minimal 2 method

interface Notifikasi {

    void kirimNotifikasi(String pesan); //
Method untuk mengirim notifikasi

    void kirimLaporan(String laporan); //
Method untuk mengirim laporan

}


abstract class Orang {

    protected final int id;

    protected String nama;

    protected final String alamat, telepon,
email;


    public Orang(int id, String nama, String
alamat, String telepon, String email) {

        this.id = id;

        this.nama = nama;

        this.alamat = alamat;

        this.telepon = telepon;

        this.email = email;

    }


    public int getId() { return id; }

    public String getNama() { return nama; }

    public void setNama(String nama) {
this.nama = nama; }


    // Abstract method yang harus
diimplementasikan oleh kelas turunan
```

```
    public abstract void kirimNotifikasi();


       @Override

    public String toString() {

        return "ID: " + id + ", Nama: " + nama
+ ", Alamat: " + alamat + ", Telepon: " +
telepon + ", Email: " + email;

    }

}


// Implementasi interface pada class Reseller

class Reseller extends Orang implements
Notifikasi {

    public Reseller(int id, String nama,
String alamat, String telepon, String email) {

        super(id, nama, alamat, telepon,
email);

    }


    public void kirimPromo() {

        System.out.println("Promo telah
dikirim ke reseller: " + getNama());

    }


    @Override

    public void kirimNotifikasi() {

        System.out.println("Notifikasi dikirim
ke Reseller: " + getNama() + " melalui email:
" + email);

    }


    // Implementasi method dari interface
Notifikasi

    @Override

    public void kirimNotifikasi(String pesan)
{

        System.out.println("Notifikasi ke
Reseller " + getNama() + ": " + pesan);
```

```
    }


    @Override
    public void kirimLaporan(String laporan) {
        System.out.println("Laporan untuk
Reseller " + getNama() + ": " + laporan);
    }


    @Override
    public String toString() {
        return "Reseller - " +
super.toString();
    }
}


// Implementasi interface pada class Pelanggan
class Pelanggan extends Orang implements
Notifikasi {
    public Pelanggan(int id, String nama,
String alamat, String telepon, String email) {
        super(id, nama, alamat, telepon,
email);
    }


    public void beriUlasan() {
        System.out.println("Pelanggan " +
getNama() + " memberikan ulasan.");
    }


    @Override
    public void kirimNotifikasi() {
        System.out.println("Notifikasi dikirim
ke Pelanggan: " + getNama() + " melalui SMS: "
+ telepon);
    }


    // Implementasi method dari interface
```

```
Notifikasi

    @Override

    public void kirimNotifikasi(String pesan)
{

        System.out.println("Notifikasi ke
Pelanggan " + getNama() + ": " + pesan);

    }



    @Override

    public void kirimLaporan(String laporan) {

        System.out.println("Laporan untuk
Pelanggan " + getNama() + ": " + laporan);

    }



    @Override

    public String toString() {

        return "Pelanggan - " +
super.toString();

    }

}



final class Menu {  // Menambahkan keyword
final pada class Menu

    private final int id;

    private final String nama, jenis;

    private double harga;

    private int stok;



    public Menu(int id, String nama, String
jenis, double harga, int stok) {

        this.id = id;

        this.nama = nama;

        this.jenis = jenis;



        // Validasi langsung tanpa memanggil
method

        if (harga <= 0) {
```

```
            throw new
IllegalArgumentException("Harga harus lebih
dari 0!");

        }

        this.harga = harga;



        if (stok < 0) {

            throw new
IllegalArgumentException("Stok tidak boleh
negatif!");

        }

        this.stok = stok;

    }



    public int getId() { return id; }

    public String getNama() { return nama; }

    public String getJenis() { return jenis; }

    public double getHarga() { return harga; }



    public void setHarga(double harga) {

        if (harga <= 0) {

            throw new
IllegalArgumentException("Harga harus lebih
dari 0!");

        }

        this.harga = harga;

    }



    public int getStok() { return stok; }



    public void setStok(int stok) {

        if (stok < 0) {

            throw new
IllegalArgumentException("Stok tidak boleh
negatif!");

        }

        this.stok = stok;
```

```
    }


    // Menambahkan keyword final pada method
toString
    @Override
    public final String toString() {
        return "ID: " + id + ", Nama: " + nama
+ ", Jenis: " + jenis + ", Harga: " + harga +
", Stok: " + stok;
    }
}


class Pesanan {
    private final int id;
    private final int idPelanggan;
    private final int idMenu;
    private final int idReseller;
    private final int jumlah;
    private final String status;
    private final String tanggal;


    public Pesanan(int id, int idPelanggan,
int idMenu, int idReseller, int jumlah, String
status, String tanggal) {
        this.id = id;
        this.idPelanggan = idPelanggan;
        this.idMenu = idMenu;
        this.idReseller = idReseller;


        if (jumlah <= 0) {
            throw new
IllegalArgumentException("Jumlah harus lebih
dari 0!");
        }
        this.jumlah = jumlah;
```

```
        if (status == null ||
status.trim().isEmpty()) {

            throw new
IllegalArgumentException("Status tidak boleh
kosong!");

        }

        this.status = status;


        this.tanggal = tanggal;

    }



    public int getId() { return id; }

    public int getIdPelanggan() { return
idPelanggan; }

    public int getIdMenu() { return idMenu; }

    public int getIdReseller() { return
idReseller; }

    public int getJumlah() { return jumlah; }

    public String getStatus() { return status;
}

    public String getTanggal() { return
tanggal; }



    @Override

    public String toString() {

        return "ID Pesanan: " + id + ",
Pelanggan: " + idPelanggan + ", Menu: " +
idMenu +

               ", Reseller: " + idReseller +
", Jumlah: " + jumlah + ", Status: " + status
+

               ", Tanggal: " + tanggal;

    }

}



public class JasaTitipApp {

    static Scanner scanner = new
Scanner(System.in);
```

```
    static ArrayList<Reseller> resellerList =
new ArrayList<>();

    static ArrayList<Menu> menuList = new
ArrayList<>();

    static ArrayList<Pelanggan> pelangganList
= new ArrayList<>();

    static ArrayList<Pesanan> pesananList =
new ArrayList<>();

    static int nextId = 1;



    // Variable static baru untuk tracking
statistik

    private static int totalPesanan = 0;

    private static double totalPendapatan = 0;



    // Method static baru untuk menambahkan
pendapatan

    public static void tambahPendapatan(double
jumlah) {

        totalPendapatan += jumlah;

        totalPesanan++;

    }



    // Method static untuk mendapatkan
statistik

    public static String getStatistik() {

        return "Total pesanan: " +
totalPesanan + ", Total pendapatan: Rp " +
totalPendapatan;

    }



    public static void main(String[] args) {

        boolean running = true;

        while (running) {

            tampilkanMenuUtama();



            int pilihan;

            try {
```

```
                pilihan =
Integer.parseInt(scanner.nextLine());

            } catch (NumberFormatException e)
{

                System.out.println("Input
tidak valid! Masukkan angka.");

                continue;

            }



            try {

                switch (pilihan) {

                    case 1 ->
tambahReseller();

                    case 2 -> tambahMenu();

                    case 3 ->
tambahPelanggan();

                    case 4 -> buatPesanan();

                    case 5 ->
lihatSemuaData();

                    case 6 ->
lihatStatistik(); // Menu baru untuk melihat
statistik

                    case 7 -> {

                        System.out.println("Te
rima kasih telah menggunakan aplikasi
JasaTitip!");

                        scanner.close();

                        running = false;

                    }

                    default ->
System.out.println("Pilihan tidak tersedia.
Silakan pilih 1-7.");

                }

            } catch (Exception e) {

                System.out.println("Terjadi
kesalahan: " + e.getMessage());

            }

        }

    }
```

```
    static void tampilkanMenuUtama() {
        System.out.println("\n==========
APLIKASI JASA TITIP MIE GACOAN ==========");
        System.out.println("1. Tambah
Reseller");
        System.out.println("2. Tambah Menu");
        System.out.println("3. Tambah
Pelanggan");
        System.out.println("4. Buat Pesanan");
        System.out.println("5. Lihat Semua
Data");
        System.out.println("6. Lihat
Statistik"); // Menu baru
        System.out.println("7. Keluar");
        System.out.print("Pilih menu: ");
    }

    // Method baru untuk melihat statistik
    static void lihatStatistik() {
        System.out.println("\n==========
STATISTIK APLIKASI ==========");
        System.out.println(getStatistik());
    }

    static void tambahReseller() {
        try {
            System.out.print("Masukkan Nama:
");
            String nama = scanner.nextLine();
            if (nama.trim().isEmpty()) {
                throw new
IllegalArgumentException("Nama tidak boleh
kosong!");
            }

            System.out.print("Masukkan Alamat:
");
```

```
            String alamat =
scanner.nextLine();

            if (alamat.trim().isEmpty()) {

                throw new
IllegalArgumentException("Alamat tidak boleh
kosong!");

            }



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

            String telepon =
scanner.nextLine();

            if (telepon.trim().isEmpty()) {

                throw new
IllegalArgumentException("Telepon tidak boleh
kosong!");

            }



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

            String email = scanner.nextLine();

            if (email.trim().isEmpty() ||
!email.contains("@")) {

                throw new
IllegalArgumentException("Format email tidak
valid!");

            }



            Reseller newReseller = new
Reseller(nextId++, nama, alamat, telepon,
email);

            resellerList.add(newReseller);



            // Menggunakan interface method

            newReseller.kirimNotifikasi("Selam
at bergabung sebagai reseller Mie Gacoan!");

            newReseller.kirimLaporan("Akun
anda telah dibuat pada " + new
Date().toString());
```

```
            System.out.println("Reseller
berhasil ditambahkan!");

        } catch (IllegalArgumentException e) {

            System.out.println("Error: " +
e.getMessage());

        } catch (Exception e) {

            System.out.println("Terjadi
kesalahan: " + e.getMessage());

        }

    }



    static void tambahPelanggan() {

        try {

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

            String nama = scanner.nextLine();

            if (nama.trim().isEmpty()) {

                throw new
IllegalArgumentException("Nama tidak boleh
kosong!");

            }



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

            String alamat =
scanner.nextLine();

            if (alamat.trim().isEmpty()) {

                throw new
IllegalArgumentException("Alamat tidak boleh
kosong!");

            }



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

            String telepon =
scanner.nextLine();

            if (telepon.trim().isEmpty()) {

                throw new
IllegalArgumentException("Telepon tidak boleh
```

```
kosong!");
            }


            System.out.print("Masukkan Email:
");
            String email = scanner.nextLine();
            if (email.trim().isEmpty() ||
!email.contains("@")) {
                throw new
IllegalArgumentException("Format email tidak
valid!");
            }


            Pelanggan newPelanggan = new
Pelanggan(nextId++, nama, alamat, telepon,
email);
            pelangganList.add(newPelanggan);


            // Menggunakan interface method
            newPelanggan.kirimNotifikasi("Sela
mat bergabung sebagai pelanggan Mie Gacoan!");
            newPelanggan.kirimLaporan("Akun
anda telah dibuat pada " + new
Date().toString());


            System.out.println("Pelanggan
berhasil ditambahkan!");
        } catch (IllegalArgumentException e) {
            System.out.println("Error: " +
e.getMessage());
        } catch (Exception e) {
            System.out.println("Terjadi
kesalahan: " + e.getMessage());
        }
    }


    static void tambahMenu() {
        try {
            System.out.print("Masukkan Nama
```

```
Menu: ");

            String nama = scanner.nextLine();

            if (nama.trim().isEmpty()) {

                throw new
IllegalArgumentException("Nama menu tidak
boleh kosong!");

            }



            System.out.print("Masukkan Jenis
Menu: ");

            String jenis = scanner.nextLine();

            if (jenis.trim().isEmpty()) {

                throw new
IllegalArgumentException("Jenis menu tidak
boleh kosong!");

            }



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

            String hargaInput =
scanner.nextLine();

            hargaInput =
hargaInput.replace(',', '.');

            double harga;

            try {

                harga =
Double.parseDouble(hargaInput);

                if (harga <= 0) {

                    throw new
IllegalArgumentException("Harga harus lebih
dari 0!");

                }

            } catch (NumberFormatException e)
{

                throw new
IllegalArgumentException("Format harga tidak
valid!");

            }



            System.out.print("Masukkan Stok:
```

```
");
            int stok;
            try {
                stok =
Integer.parseInt(scanner.nextLine());
                if (stok < 0) {
                    throw new
IllegalArgumentException("Stok tidak boleh
negatif!");
                }
            } catch (NumberFormatException e)
{
                throw new
IllegalArgumentException("Format stok tidak
valid!");
            }


            menuList.add(new Menu(nextId++,
nama, jenis, harga, stok));
            System.out.println("Menu berhasil
ditambahkan!");
        } catch (IllegalArgumentException e) {
            System.out.println("Error: " +
e.getMessage());
        } catch (Exception e) {
            System.out.println("Terjadi
kesalahan: " + e.getMessage());
        }
    }


    static void buatPesanan() {
        try {
            if (pelangganList.isEmpty()) {
                throw new
IllegalStateException("Belum ada pelanggan!
Tambahkan pelanggan terlebih dahulu.");
            }
            if (menuList.isEmpty()) {
                throw new
```

```
IllegalStateException("Belum ada menu!
Tambahkan menu terlebih dahulu.");

            }

            if (resellerList.isEmpty()) {

                throw new
IllegalStateException("Belum ada reseller!
Tambahkan reseller terlebih dahulu.");

            }



            System.out.println("\n--- Daftar
Pelanggan ---");

            for (Pelanggan p : pelangganList)
{

                System.out.println(p);

            }



            System.out.print("Masukkan ID
Pelanggan: ");

            int idPelanggan;

            try {

                idPelanggan =
Integer.parseInt(scanner.nextLine());

            } catch (NumberFormatException e)
{

                throw new
IllegalArgumentException("Format ID pelanggan
tidak valid!");

            }



            Pelanggan pelangganDipilih = null;

            for (Pelanggan p : pelangganList)
{

                if (p.getId() == idPelanggan)
{

                    pelangganDipilih = p;

                    break;

                }

            }

            if (pelangganDipilih == null) {
```

```
                throw new
IllegalArgumentException("ID Pelanggan tidak
ditemukan!");
            }


            System.out.println("\n--- Daftar
Menu ---");
            for (Menu m : menuList) {
                System.out.println(m);
            }


            System.out.print("Masukkan ID
Menu: ");
            int idMenu;
            try {
                idMenu =
Integer.parseInt(scanner.nextLine());
            } catch (NumberFormatException e)
{
                throw new
IllegalArgumentException("Format ID menu tidak
valid!");
            }


            Menu menuDipilih = null;
            for (Menu m : menuList) {
                if (m.getId() == idMenu) {
                    menuDipilih = m;
                    break;
                }
            }
            if (menuDipilih == null) {
                throw new
IllegalArgumentException("ID Menu tidak
ditemukan!");
            }


            System.out.println("\n--- Daftar
```

```
Reseller ---");
            for (Reseller r : resellerList) {
                System.out.println(r);
            }

            System.out.print("Masukkan ID
Reseller: ");
            int idReseller;
            try {
                idReseller =
Integer.parseInt(scanner.nextLine());
            } catch (NumberFormatException e)
{
                throw new
IllegalArgumentException("Format ID reseller
tidak valid!");
            }

            Reseller resellerDipilih = null;
            for (Reseller r : resellerList) {
                if (r.getId() == idReseller) {
                    resellerDipilih = r;
                    break;
                }
            }
            if (resellerDipilih == null) {
                throw new
IllegalArgumentException("ID Reseller tidak
ditemukan!");
            }

            System.out.print("Masukkan Jumlah:
");
            int jumlah;
            try {
                jumlah =
Integer.parseInt(scanner.nextLine());
                if (jumlah <= 0) {
```

```
                    throw new
IllegalArgumentException("Jumlah harus lebih
dari 0!");

                }

            } catch (NumberFormatException e)
{

                throw new
IllegalArgumentException("Format jumlah tidak
valid!");

            }



            if (jumlah >
menuDipilih.getStok()) {

                throw new
IllegalArgumentException("Stok tidak
mencukupi! Stok tersedia: " +
menuDipilih.getStok());

            }



            System.out.print("Masukkan Status
(contoh: Diproses, Dikirim, Selesai): ");

            String status =
scanner.nextLine();

            if (status.trim().isEmpty()) {

                throw new
IllegalArgumentException("Status tidak boleh
kosong!");

            }



            menuDipilih.setStok(menuDipilih.ge
tStok() - jumlah);



            // Menambahkan data pendapatan
(static)

            tambahPendapatan(menuDipilih.getHa
rga() * jumlah);



            pesananList.add(new
Pesanan(nextId++, idPelanggan, idMenu,
idReseller, jumlah, status, new
Date().toString()));
```

```
            System.out.println("Pesanan
berhasil dibuat!");

            System.out.println("Detail:
Pelanggan " + pelangganDipilih.getNama() +

                               " memesan " +
jumlah + " " + menuDipilih.getNama() +

                               " melalui
reseller " + resellerDipilih.getNama());



            // Menggunakan method original

            pelangganDipilih.kirimNotifikasi()
;

            resellerDipilih.kirimNotifikasi();



            // Menggunakan method dari
interface

            String pesanPelanggan = "Pesanan "
+ menuDipilih.getNama() + " sejumlah " +
jumlah + " sedang " + status;

            String pesanReseller = "Ada
pesanan baru untuk diproses: " +
menuDipilih.getNama() + " sejumlah " + jumlah;



            pelangganDipilih.kirimNotifikasi(p
esanPelanggan);

            resellerDipilih.kirimNotifikasi(pe
sanReseller);



            String laporanPelanggan = "Total
biaya: Rp " + (menuDipilih.getHarga() *
jumlah);

            pelangganDipilih.kirimLaporan(lapo
ranPelanggan);



        } catch (IllegalArgumentException |
IllegalStateException e) {

            System.out.println("Error: " +
e.getMessage());

        } catch (Exception e) {

            System.out.println("Terjadi
kesalahan: " + e.getMessage());
```

```
        }
    }


    static void lihatSemuaData() {
        boolean adaData = false;


        System.out.println("\n========== DATA
JASA TITIP ==========");


        System.out.println("\n--- DAFTAR
RESELLER ---");
        if (resellerList.isEmpty()) {
            System.out.println("Belum ada data
reseller.");
        } else {
            adaData = true;
            for (Reseller r : resellerList) {
                System.out.println(r);
            }
        }


        System.out.println("\n--- DAFTAR
PELANGGAN ---");
        if (pelangganList.isEmpty()) {
            System.out.println("Belum ada data
pelanggan.");
        } else {
            adaData = true;
            for (Pelanggan p : pelangganList)
{
                System.out.println(p);
            }
        }


        System.out.println("\n--- DAFTAR MENU
---");
```

```
        if (menuList.isEmpty()) {
            System.out.println("Belum ada data
menu.");
        } else {
            adaData = true;
            for (Menu m : menuList) {
                System.out.println(m);
            }
        }


        System.out.println("\n--- DAFTAR
PESANAN ---");
        if (pesananList.isEmpty()) {
            System.out.println("Belum ada data
pesanan.");
        } else {
            adaData = true;
            for (Pesanan p : pesananList) {
                String namaPelanggan =
"Pelanggan tidak ditemukan";
                String namaMenu = "Menu tidak
ditemukan";
                String namaReseller =
"Reseller tidak ditemukan";


                for (Pelanggan pel :
pelangganList) {
                    if (pel.getId() ==
p.getIdPelanggan()) {
                        namaPelanggan =
pel.getNama();
                        break;
                    }
                }


                for (Menu m : menuList) {
                    if (m.getId() ==
p.getIdMenu()) {
```

```
                        namaMenu =
m.getNama();
                        break;
                    }
                }

                for (Reseller r :
resellerList) {
                    if (r.getId() ==
p.getIdReseller()) {
                        namaReseller =
r.getNama();
                        break;
                    }
                }

                System.out.println(p);
                System.out.println("   Detail:
Pelanggan: " + namaPelanggan +
                                   ", Menu: " +
namaMenu +
                                   ", Reseller:
" + namaReseller);
            }
        }

        if (!adaData) {
            System.out.println("\nBelum ada
data yang tersimpan di aplikasi.");
        }
    }
}
```