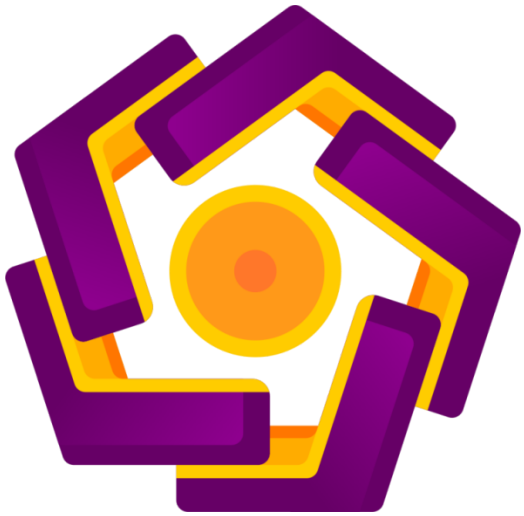
LEMBAR JAWAB UAS

MATA KULIAH PEMROGRAMAN



Nama :	Achmad Rizki Ramadhan
NIM :	22.11.4879
Dosen Pengampu :	Abd. Mizwar A. Rahim, M.Kom

S1-INFORMATIKA UNIVERSITAS AMIKOM YOGYAKARTA

2023

Soal Nomer 1

Encapsulation

Kodingan :

```
public class Pembayaran
{
    // Menyimpan jumlah pembayaran sebagai variabel private
    private double jumlah;
    // Menyimpan mata uang pembayaran sebagai variabel private
    private string mataUang;

    public double Jumlah
    {
        // getter untuk variabel jumlah
        get { return jumlah; }
        // setter untuk variabel jumlah
        set { jumlah = value; }
    }

    public string MataUang
    {
        // getter untuk variabel mataUang
        get { return mataUang; }
        // setter untuk variabel mataUang
        set { mataUang = value; }
    }

    public void ProsesPembayaran()
    {
        Console.WriteLine("Memproses pembayaran sejumlah {0}
{1}...", jumlah, mataUang);
        Console.WriteLine("Pembayaran berhasil diproses!");
    }
}
```

```
public class Program
{
    public static void Main(string[] args)
    {
        // Membuat objek pembayaran dari kelas Pembayaran
        Pembayaran pembayaran = new Pembayaran();

        Console.Write("Masukkan jumlah pembayaran: ");
        double jumlah = Convert.ToDouble(Console.ReadLine());

        Console.Write("Masukkan mata uang: ");
        string mataUang = Console.ReadLine();

        // Mengatur nilai jumlah pada objek pembayaran menggunakan
setter
        pembayaran.Jumlah = jumlah;

        // Mengatur nilai mataUang pada objek pembayaran
menggunakan setter
        pembayaran.MataUang = mataUang;

        // Memanggil metode ProsesPembayaran pada objek pembayaran
        pembayaran.ProsesPembayaran();

        Console.ReadLine();
    }
}
```

Hasil Program:

```
Masukkan jumlah pembayaran: 12000
Masukkan mata uang: idr
Memproses pembayaran sejumlah 12000 idr...
Pembayaran berhasil diproses!
```

Penjelasan Kodingan:

Kode tersebut merupakan kode yang mengimplementasikan salah satu konsep OOP yaitu Encapsulation yang mana pada class pembayaran menyediakan properti publik Jumlah dan MataUang yang memungkinkan akses kontrol terhadap variabel privat tersebut melalui metode get dan set.

Inheritance

Kodingan

```
namespace Inheritance;

// class induk (superclass)
public class Pembayaran
{
    // Menyimpan jumlah pembayaran
    private double jumlah;
    // Menyimpan mata uang pembayaran
    private string mataUang;

    public double Jumlah
    {
        // Getter untuk mendapatkan nilai jumlah
        get { return jumlah; }
        // Setter untuk mengatur nilai jumlah
        set { jumlah = value; }
    }

    public string MataUang
    {
        // Getter untuk mendapatkan nilai mataUang
        get { return mataUang; }
        // Setter untuk mengatur nilai mataUang
        set { mataUang = value; }
    }

    // Metode virtual yang akan di override oleh subclass
    public virtual void ProsesPembayaran()
    {
        Console.WriteLine("Memproses pembayaran sejumlah {0}
{1}...", jumlah, mataUang);
        Console.WriteLine("Pembayaran berhasil diproses!");
    }
}
```

```
// class anak (subclass)
public class PembayaranKartuKredit : Pembayaran
{
    // Menyimpan nomor kartu kredit
    private string nomorKartu;
    // Menyimpan nama pemegang kartu kredit
    private string namaPemegangKartu;
    // Menyimpan tanggal kadaluarsa kartu kredit
    private string tanggalKadaluarsa;


    public string NomorKartu
    {
        // Getter untuk mendapatkan nilai nomorKartu
        get { return nomorKartu; }
        // Setter untuk mengatur nilai nomorKartu
        set { nomorKartu = value; }
    }

    public string NamaPemegangKartu
    {
        // Getter untuk mendapatkan nilai namaPemegangKartu
        get { return namaPemegangKartu; }
        // Setter untuk mengatur nilai namaPemegangKartu
        set { namaPemegangKartu = value; }
    }

    public string TanggalKadaluarsa
    {
        // Getter untuk mendapatkan nilai tanggalKadaluarsa
        get { return tanggalKadaluarsa; }
        // Setter untuk mengatur nilai tanggalKadaluarsa
        set { tanggalKadaluarsa = value; }
    }
```

```
    // Metode override yang menggantikan metode virtual pada
superclass
    public override void ProsesPembayaran()
    {
        Console.WriteLine("Memproses pembayaran kartu kredit
dengan nomor {0} atas nama {1}...", nomorKartu,
namaPemegangKartu);
        Console.WriteLine("Pembayaran kartu kredit berhasil
diproses!");
    }
}

public class Program
{
    public static void Main(string[] args)
    {
        // Membuat objek pembayaran
        Pembayaran pembayaran = new Pembayaran();

        Console.Write("Masukkan jumlah pembayaran: ");
        double jumlah = Convert.ToDouble(Console.ReadLine());

        Console.Write("Masukkan mata uang: ");
        string mataUang = Console.ReadLine();

        // Mengatur nilai jumlah pada objek pembayaran
        pembayaran.Jumlah = jumlah;
        // Mengatur nilai mataUang pada objek pembayaran
        pembayaran.MataUang = mataUang;

        // Memanggil metode ProsesPembayaran pada objek pembayaran
        pembayaran.ProsesPembayaran();

        Console.WriteLine();
```

```
        // Membuat objek pembayaran kartu kredit
        PembayaranKartuKredit pembayaranKartuKredit = new
PembayaranKartuKredit();
        Console.Write("Masukkan jumlah pembayaran kartu kredit:
");
        double jumlahKartuKredit =
Convert.ToDouble(Console.ReadLine());
        Console.Write("Masukkan mata uang kartu kredit: ");
        string mataUangKartuKredit = Console.ReadLine();
        Console.Write("Masukkan nomor kartu kredit: ");
        string nomorKartuKredit = Console.ReadLine();
        Console.Write("Masukkan nama pemegang kartu kredit: ");
        string namaPemegangKartuKredit = Console.ReadLine();
        Console.Write("Masukkan tanggal kadaluarsa kartu kredit:
");
        string tanggalKadaluarsaKartuKredit = Console.ReadLine();
         // Mengatur nilai jumlah pada objek pembayaran kartu
kredit
        pembayaranKartuKredit.Jumlah = jumlahKartuKredit;
        // Mengatur nilai mataUang pada objek pembayaran kartu
kredit
        pembayaranKartuKredit.MataUang = mataUangKartuKredit;
        // Mengatur nilai nomorKartu pada objek pembayaran kartu
kredit
        pembayaranKartuKredit.NomorKartu = nomorKartuKredit;
        // Mengatur nilai namaPemegangKartu pada objek pembayaran
kartu kredit
        pembayaranKartuKredit.NamaPemegangKartu =
namaPemegangKartuKredit;
        // Mengatur nilai tanggalKadaluarsa pada objek pembayaran
kartu kredit
        pembayaranKartuKredit.TanggalKadaluarsa =
tanggalKadaluarsaKartuKredit;
                // Memanggil metode ProsesPembayaran pada objek
pembayaran kartu kredit
        pembayaranKartuKredit.ProsesPembayaran();
        Console.ReadLine();
    }
}
```

Hasil Program:

```
Masukkan jumlah pembayaran: 120000
Masukkan mata uang: idr
Memproses pembayaran sejumlah 120000 idr...
Pembayaran berhasil diproses!

Masukkan jumlah pembayaran kartu kredit: 50000
Masukkan mata uang kartu kredit: idr
Masukkan nomor kartu kredit: 123456
Masukkan nama pemegang kartu kredit: rizki
Masukkan tanggal kadaluarsa kartu kredit: 1223
Memproses pembayaran kartu kredit dengan nomor 123456 atas nama rizki...
Pembayaran kartu kredit berhasil diproses!
```

Penjelasan Program:

Dalam kode tersebut, konsep inheritance diimplementasikan dengan menggunakan kelas induk `Pembayaran` dan kelas anak `PembayaranKartuKredit`. Kelas anak mewarisi atribut dan metode dari kelas induk, sehingga memiliki akses terhadap properti `Jumlah` dan `MataUang`, serta dapat mengganti implementasi metode `ProsesPembayaran()`. Hal ini memungkinkan kelas anak untuk memiliki perilaku tambahan yang spesifik, sementara masih mempertahankan atribut dan perilaku yang didefinisikan dalam kelas induk.

Polymorphism

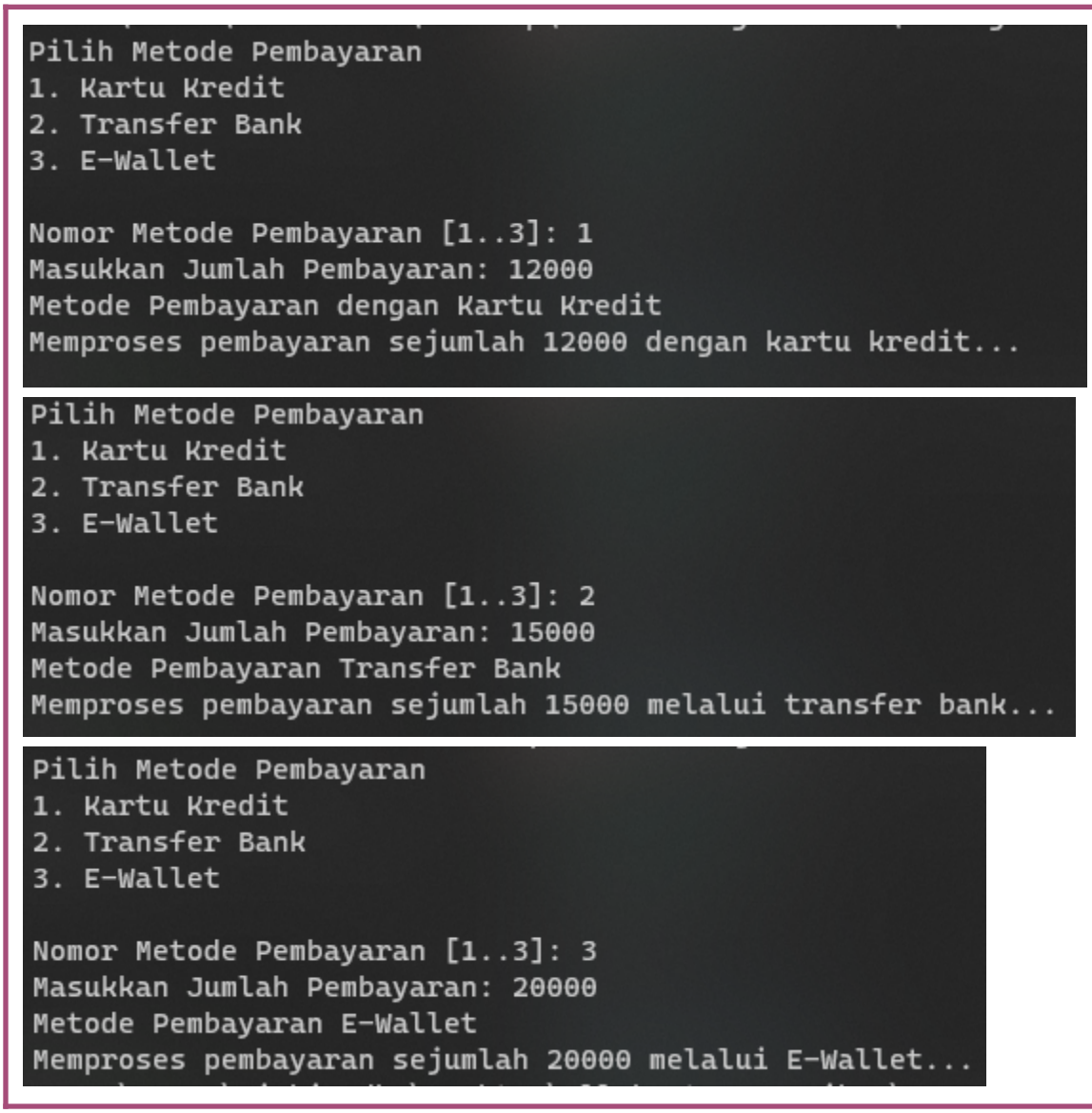
Kodingan

```
using System;
namespace Polymorphism
{
    internal class Program
    {
        static void Main(string[] args)
        {
            // deklarasi variabel pembayaran dengan tipe data
Pembayaran (superclass)
            Pembayaran pembayaran;

            Console.WriteLine("Pilih Metode Pembayaran");
            Console.WriteLine("1. Kartu Kredit");
            Console.WriteLine("2. Transfer Bank");
            Console.WriteLine("3. E-Wallet\n");

            Console.Write("Nomor Metode Pembayaran [1..3]: ");
            int nomorMetode = Convert.ToInt32(Console.ReadLine());

            Console.Write("Masukkan Jumlah Pembayaran: ");
            double jumlahPembayaran =
Convert.ToDouble(Console.ReadLine());
            if (nomorMetode == 1){
                // inisialisasi variabel pembayaran dengan objek
PembayaranKartuKredit (subclass)
                pembayaran = new PembayaranKartuKredit();
            }else if (nomorMetode == 2){
                // inisialisasi variabel pembayaran dengan objek
PembayaranTransferBank (subclass)
                pembayaran = new PembayaranTransferBank();
            }else{
                // inisialisasi variabel pembayaran dengan objek
PembayaranEWallet (subclass)
                pembayaran = new PembayaranEWallet();
            }
```

```
            // memanggil method Show() dari objek pembayaran
            pembayaran.Show();
            // memanggil method ProcessPayment() dari objek
pembayaran
            pembayaran.ProcessPayment(jumlahPembayaran);
            Console.ReadKey();
        }
    }

    //public class Pembayaran (superclass)
    public class Pembayaran
    {
        //public virtual void Show() (method virtual)
        //virtual adalah method yang dapat di override oleh
subclass
        public virtual void Show()
        {
            Console.WriteLine("Metode Pembayaran Tersedia");
        }

        //public virtual void ProcessPayment(double jumlah)
(method virtual)
        public virtual void ProcessPayment(double jumlah)
        {
            Console.WriteLine("Silahkan Pilih Metode Pembayaran");
        }
    }

    //public class PembayaranKartuKredit : Pembayaran (subclass)
    public class PembayaranKartuKredit : Pembayaran
    {
        //public override void Show() (method override)
        //overide adalah method yang memiliki nama, parameter, dan
tipe data yang sama dengan method yang ada di superclass
        public override void Show()
        {
            Console.WriteLine("Metode Pembayaran dengan Kartu
Kredit");
```

```
        }
        public override void ProcessPayment(double jumlah)
        {
            Console.WriteLine("Memproses pembayaran sejumlah {0}
dengan kartu kredit...", jumlah);
        }
    }
    public class PembayaranTransferBank : Pembayaran
    {
        public override void Show()
        {
            Console.WriteLine("Metode Pembayaran Transfer Bank");
        }

        public override void ProcessPayment(double jumlah)
        {
            Console.WriteLine("Memproses pembayaran sejumlah {0}
melalui transfer bank...", jumlah);
        }
    }

    public class PembayaranEWallet : Pembayaran
    {
        public override void Show()
        {
            Console.WriteLine("Metode Pembayaran E-Wallet");
        }

        public override void ProcessPayment(double jumlah)
        {
            Console.WriteLine("Memproses pembayaran sejumlah {0}
melalui E-Wallet...", jumlah);
        }
    }
}
```

Hasil Program:



Penjelasan Program:

Pada kode tersebut, terdapat penggunaan polimorfisme. Polimorfisme adalah konsep dalam OOP di mana objek dari kelas anak dapat dianggap sebagai objek dari kelas induk, sehingga dapat digunakan secara umum tanpa memperhatikan jenis spesifiknya. Dalam kode tersebut, deklarasi variabel `pembayaran` dengan tipe data `Pembayaran` (superclass) memungkinkan untuk merujuk pada objek dari kelas anak seperti `PembayaranKartuKredit`, `PembayaranTransferBank`, dan `PembayaranEWallet`. Melalui pemanggilan metode `Show()` dan `ProcessPayment()`, metode yang dijalankan adalah metode yang telah di-override oleh kelas anak, sehingga menghasilkan perilaku yang sesuai dengan jenis objek yang digunakan.

Abstraction

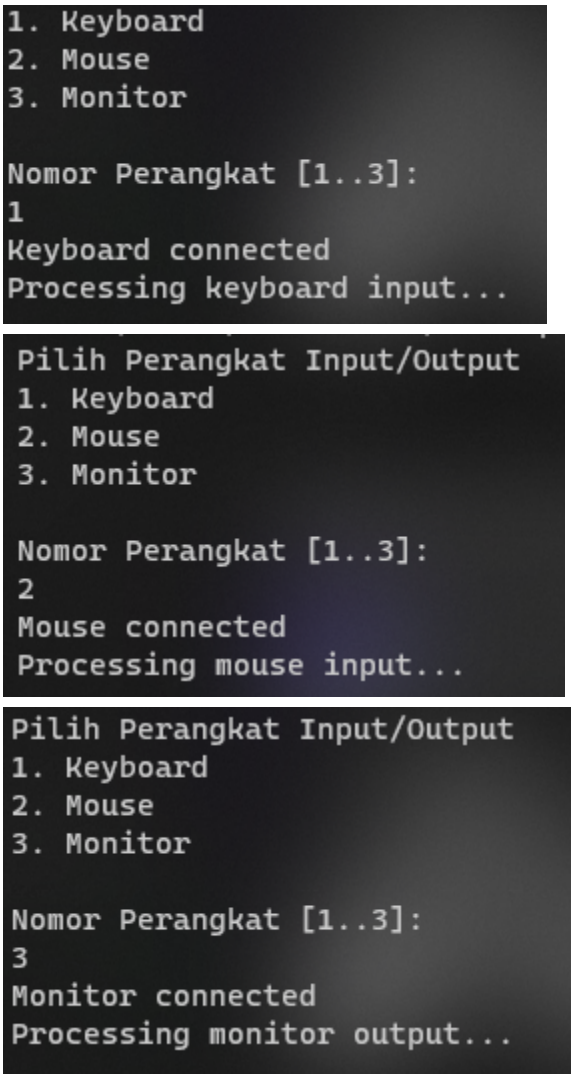
Kodingan

```
using System;

namespace Abstraction
{
    internal class Program
    {
        static void Main(string[] args)
        {
            // deklarasi variabel device dengan tipe data IODevice
(superclass)
            IODevice device;

            Console.WriteLine("Pilih Perangkat Input/Output");
            Console.WriteLine("1. Keyboard");
            Console.WriteLine("2. Mouse");
            Console.WriteLine("3. Monitor\n");

            Console.WriteLine("Nomor Perangkat [1..3]: ");
            int deviceNumber =
Convert.ToInt32(Console.ReadLine());

            if (deviceNumber == 1)
            {
                // inisialisasi variabel device dengan objek
Keyboard (subclass)
                device = new Keyboard();
            }
            else if (deviceNumber == 2)
            {
                // inisialisasi variabel device dengan objek Mouse
(subclass)
                device = new Mouse();
            }
```

```
            else
            {
                // inisialisasi variabel device dengan objek
Monitor (subclass)
                device = new Monitor();
            }
            // memanggil method Show() dari objek device
            device.Show();
            // memanggil method ProcessInput() dari objek device
            device.ProcessInput();
            Console.ReadKey();
        }
    }

    //public class IODevice (superclass)
    //abstract adalah class yang tidak dapat di instansiasi (tidak
dapat dibuat objeknya)
    //class ini hanya dapat diwarisi oleh subclass
    public abstract class IODevice
    {
        public abstract void Show();
        public abstract void ProcessInput();
    }

    //public class Keyboard : IODevice (subclass)
    public class Keyboard : IODevice
    {
        //public override void Show() (method override)
        //override adalah method yang menggantikan method virtual
dari superclass
        public override void Show()
        {
            Console.WriteLine("Keyboard connected");
        }

        public override void ProcessInput()
        {
            Console.WriteLine("Processing keyboard input...");
        }
```

```
    }

    //public class Mouse : IODevice (subclass)
    public class Mouse : IODevice
    {
        public override void Show()
        {
            Console.WriteLine("Mouse connected");
        }

        public override void ProcessInput()
        {
            Console.WriteLine("Processing mouse input...");
        }
    }

    //public class Monitor : IODevice (subclass)
    public class Monitor : IODevice
    {
        public override void Show()
        {
            Console.WriteLine("Monitor connected");
        }

        public override void ProcessInput()
        {
            Console.WriteLine("Processing monitor output...");
        }
    }
}
```

Hasil Program:



Penjelasan Program:

Pada kode tersebut, terdapat penggunaan abstraksi. Abstraksi adalah konsep dalam OOP di mana sebuah kelas abstrak memiliki satu atau lebih metode abstrak yang tidak memiliki implementasi. Kelas abstrak ini tidak dapat diinstansiasi langsung, tetapi hanya dapat diwarisi oleh kelas-kelas anak yang memberikan implementasi khusus untuk metode abstrak tersebut. Dalam kode tersebut, kelas `IODevice` (superclass) adalah kelas abstrak yang memiliki metode abstrak `Show()` dan `ProcessInput()`. Kelas anak seperti `Keyboard`, `Mouse`, dan `Monitor` mewarisi kelas `IODevice` dan memberikan implementasi spesifik untuk metode-metode tersebut.

Soal Nomer 2 : Aplikasi untuk proses pemesanan tiket

Kodingan

```
public abstract class Pemesanan
{
    private string nama; // field nama
    private string nik; // field nik
    private string nomorTelepon; // field nomorTelepon

    // property nama
    public string Nama
    {
        get { return nama; }
        set { nama = value; }
    }

    // property nik
    public string NIK
    {
        get { return nik; }
        set { nik = value; }
    }

    // property nomorTelepon
    public string NomorTelepon
    {
        get { return nomorTelepon; }
        set { nomorTelepon = value; }
    }

    // method abstrak ProsesPemesanan yang akan diimplementasikan
di kelas turunan
    public abstract void ProsesPemesanan();
}

// kelas PemesananTiket yang merupakan turunan dari kelas
Pemesanan
```

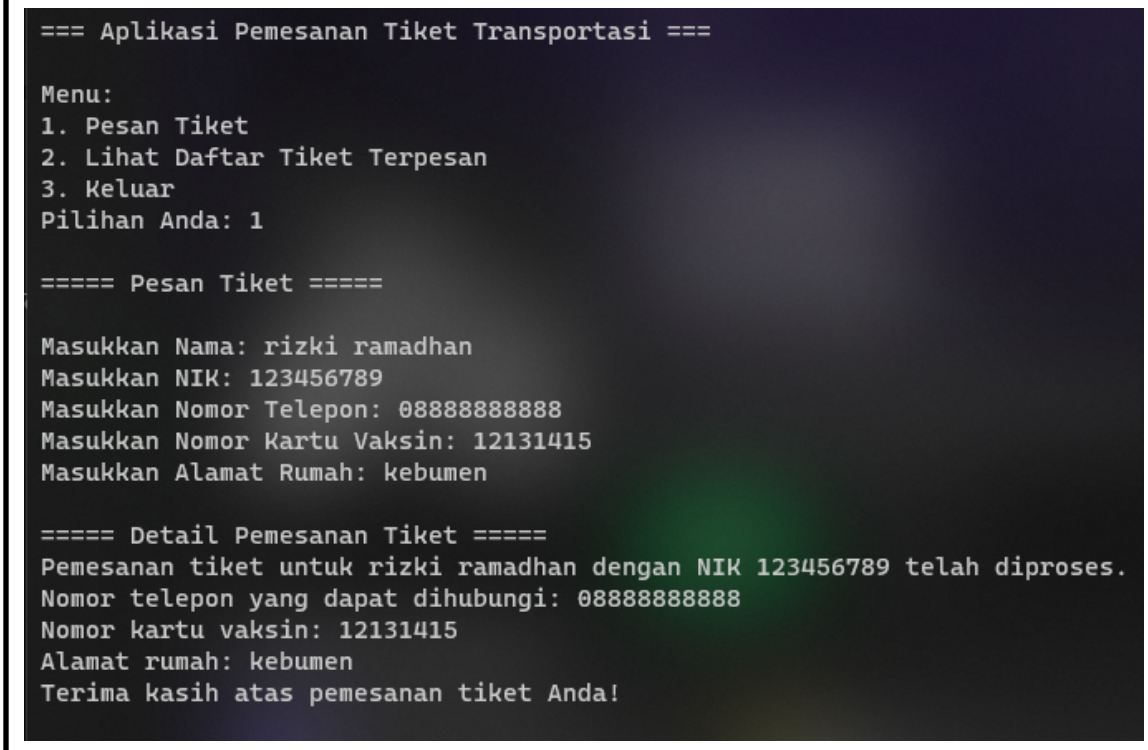
```
public class PemesananTiket : Pemesanan
{
    private string nomorKartuVaksin; // field nomorKartuVaksin
    private string alamatRumah; // field alamatRumah

    // property nomorKartuVaksin
    public string NomorKartuVaksin
    {
        get { return nomorKartuVaksin; }
        set { nomorKartuVaksin = value; }
    }

    // property alamatRumah
    public string AlamatRumah
    {
        get { return alamatRumah; }
        set { alamatRumah = value; }
    }

    // implementasi method ProsesPemesanan yang diwarisi dari
kelas Pemesanan
    public override void ProsesPemesanan()
    {
        Console.WriteLine();
        Console.WriteLine("===== Detail Pemesanan Tiket =====");
        Console.WriteLine("Pemesanan tiket untuk {0} dengan NIK
{1} telah diproses.", Nama, NIK);
        Console.WriteLine("Nomor telepon yang dapat dihubungi:
{0}", NomorTelepon);
        Console.WriteLine("Nomor kartu vaksin: {0}",
nomorKartuVaksin);
        Console.WriteLine("Alamat rumah: {0}", alamatRumah);
        Console.WriteLine("Terima kasih atas pemesanan tiket
Anda!");
        Console.WriteLine();
    }
}
```

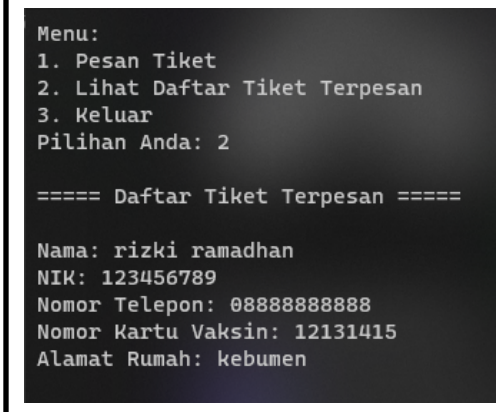
```
public class Program
{
    // list daftarPemesanan untuk menyimpan objek Pemesanan
    static List<Pemesanan> daftarPemesanan = new
List<Pemesanan>();

    public static void Main(string[] args)
    {
        Console.WriteLine();
        Console.WriteLine("=== Aplikasi Pemesanan Tiket
Transportasi ===");
        Console.WriteLine();

        bool isRunning = true;
        while (isRunning)
        {
            Console.WriteLine("Menu:");
            Console.WriteLine("1. Pesan Tiket");
            Console.WriteLine("2. Lihat Daftar Tiket Terpesan");
            Console.WriteLine("3. Keluar");
            Console.Write("Pilihan Anda: ");
            string pilihan = Console.ReadLine();

            switch (pilihan)
            {
                case "1":
                    PesanTiket();
                    break;
                case "2":
                    LihatDaftarTiketTerpesan();
                    break;
                case "3":
                    isRunning = false;
                    break;
                default:
                    Console.WriteLine("Pilihan tidak valid.
Silakan coba lagi.");
                    break;
            }
```

```
            Console.WriteLine();
        }
    }

    public static void PesanTiket()
    {
        Console.WriteLine();
        Console.WriteLine("===== Pesan Tiket =====");
        Console.WriteLine();

        Console.Write("Masukkan Nama: ");
        string nama = Console.ReadLine();

        Console.Write("Masukkan NIK: ");
        string nik = Console.ReadLine();

        Console.Write("Masukkan Nomor Telepon: ");
        string nomorTelepon = Console.ReadLine();

        Console.Write("Masukkan Nomor Kartu Vaksin: ");
        string nomorKartuVaksin = Console.ReadLine();

        Console.Write("Masukkan Alamat Rumah: ");
        string alamatRumah = Console.ReadLine();

        // membuat objek PemesananTiket dan menyimpannya di
variabel pemesanan
        Pemesanan pemesanan = new PemesananTiket
        {
            Nama = nama,
            NIK = nik,
            NomorTelepon = nomorTelepon,
            NomorKartuVaksin = nomorKartuVaksin,
            AlamatRumah = alamatRumah
        };


        daftarPemesanan.Add(pemesanan);
        pemesanan.ProsesPemesanan();
```

```
        Console.WriteLine();
    }

    public static void LihatDaftarTiketTerpesan()
    {

        Console.WriteLine();
        Console.WriteLine("===== Daftar Tiket Terpesan =====");
        Console.WriteLine();

        // mengecek apakah daftarPemesanan memiliki isi atau tidak
        if (daftarPemesanan.Count > 0){
            foreach (Pemesanan pemesanan in daftarPemesanan)
            {
                Console.WriteLine("Nama: {0}", pemesanan.Nama);
                Console.WriteLine("NIK: {0}", pemesanan.NIK);
                Console.WriteLine("Nomor Telepon: {0}",
pemesanan.NomorTelepon);

                // mengecek apakah pemesanan merupakan objek
PemesananTiket
                if (pemesanan is PemesananTiket pemesananTiket)
                {
                    Console.WriteLine("Nomor Kartu Vaksin: {0}",
pemesananTiket.NomorKartuVaksin);
                    Console.WriteLine("Alamat Rumah: {0}",
pemesananTiket.AlamatRumah);
                }
                Console.WriteLine();
            }
        }
        else{
            Console.WriteLine("Belum ada tiket yang terpesan.");
        }

        Console.WriteLine();
    }
}
```

Hasil Program





Penjelasan

Program di atas adalah sebuah aplikasi pemesanan tiket transportasi yang mengimplementasikan konsep abstraksi. Kelas abstrak `Pemesanan` memiliki beberapa field seperti `nama`, `nik`, dan `nomorTelepon`, serta memiliki metode abstrak `ProsesPemesanan()`. Kelas `PemesananTiket` merupakan turunan dari `Pemesanan` yang memiliki field tambahan seperti `nomorKartuVaksin` dan `alamatRumah`, serta mengimplementasikan metode `ProsesPemesanan()` dengan tampilan informasi pemesanan tiket. Pada kelas `Program`, terdapat daftar pemesanan yang disimpan dalam `List<Pemesanan> daftarPemesanan`. Aplikasi ini memiliki menu untuk pesan tiket, lihat daftar tiket terpesan, dan keluar. Metode `PesanTiket()` digunakan untuk membuat objek `PemesananTiket` dan menambahkannya ke daftar pemesanan. Metode `LihatDaftarTiketTerpesan()` digunakan untuk menampilkan informasi dari setiap objek pemesanan dalam daftar pemesanan.