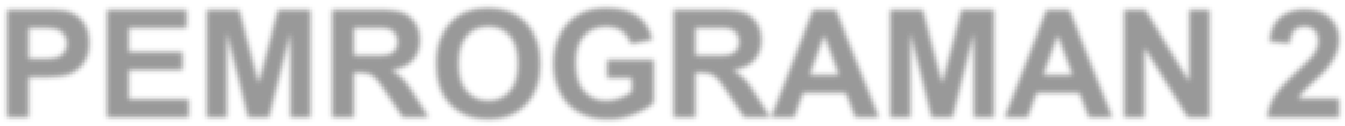
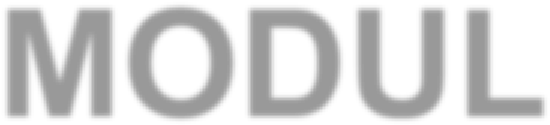
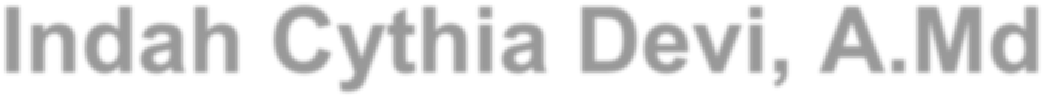
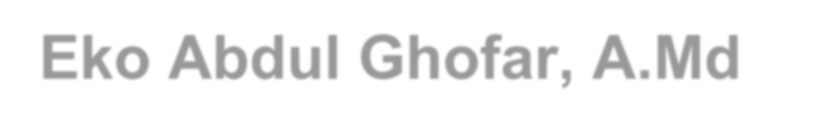
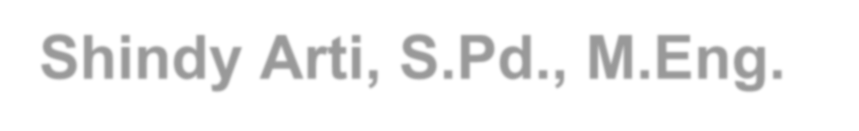
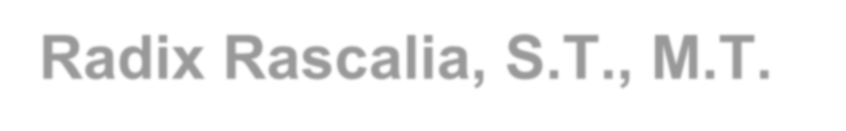
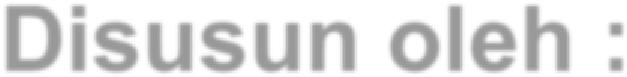
**MODUL**

**PEMROGRAMAN 2**



**Disusun oleh :**

**Radix Rascalia, S.T., M.T.**



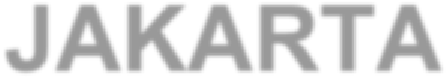
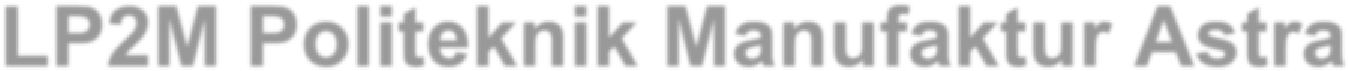
|  |
| --- |
|  |
|  |
|  |

 **Shindy Arti, S.Pd., M.Eng.**

**Eko Abdul Ghofar, A.Md**

**Indah Cythia Devi, A.Md**

**LP2M Politeknik Manufaktur Astra**



**JAKARTA**



**MODUL Mata Kuliah**

**Penulis :**

**Radix Rascalia, S.T., M.T.**

**Shindy Arti, S.Pd., M.Eng.**

**Eko Abdul Ghofar,A.Md**

**Indah Cynthia Devi, A.Md**

**Penerbit :**

**LP2M Politeknik Manufaktur Astra**

**Jl. Gaya Motor Raya No.8 Sunter II Jakarta 14330**

**Telepon: (021) 6519555 Fax: (021) 6519821**

**Email: sekretariat@polman.astra.ac.id**

**KATA PENGANTAR**

Modul praktikum ini dibuat sebagai pedoman mahasiswa dalam melakukan kegiatan praktikum mata kuliah Pemrograman 2 di Politeknik Manufaktur Astra. Modul praktikum ini terdiri dari 12 topik praktikum yang membahas tentang dasar pemrograman C#, OOP, Windows Form, Kontrol Windows Form, koneksi program C# dengan basis data menggunakan .NET Framework, Store Procedure dan LINQ. Topik praktikum lainnya membahas mengenai pembuatan laporan winodws form, asinkronous dan WPF, deployment aplikasi windows form dan versioning dan studi kasus. Modul praktikum ini diharapkan dapat membantu mahasiswa/i dalam melaksanakan praktikum dengan lebih baik, terarah, dan terencana. Pada setiap topik telah ditetapkan tujuan pelaksanaan praktikum dan semua kegiatan yang harus dilakukan oleh mahasiswa/i serta teori singkat untuk memperdalam pemahaman mahasiswa/i mengenai materi yang dibahas.

Penyusun menyakini bahwa dalam pembuatan Modul Praktikum mata kuliah

Pemrograman 2 ini masih jauh dari sempurna. Oleh karena itu penyusun mengharapkan kritik dan saran yang membangun guna penyempurnaan modul praktikum ini dimasa yang akan datang. Akhir kata, penyusun mengucapkan banyak terima kasih kepada semua pihak yang telah membantu baik secara langsung maupun tidak langsung.

Jakarta, Maret 2022

Tim Penulis

**DAFTAR ISI**

KATA PENGANTAR ................................................................................................................ ii

DAFTAR ISI .............................................................................................................................. iii

[M01 Pengenalan Pemrograman 2 ............................................................................................... 1 1. TUJUAN 1](#_Toc15163)

[2. DURASI WAKTU 1](#_Toc15164)

[3. DASAR TEORI 1](#_Toc15165)

[4. PERCOBAAN 10](#_Toc15166)

[5. LATIHAN 20](#_Toc15167)

**M02 Object Oriented Program C#**

**1. TUJUAN** CPMK :

Sub-CPMK : Mahasiswa mampu menerapkan program dengan bahasa pemrograman

C#

## 2. DURASI WAKTU

2 pertemuan x 170 menit

## 3. DASAR TEORI

### Object Oriented Programming

Object Oriented Programming adalah suatu metode pemrograman yang berorientasi kepada objek. Tujuannya mengatur kompleksitas dengan pemodelan yang sudah ada untuk mempermudah pengembangan program.

Dalam paradigma OOP, solusi dibangun dari 1 objek atau lebih yang saling bekerja sama untuk menyelesaikan masalah. Objek merupakan benda (nyata dan tidak nyata). Setiap objek memiliki nilai yang melekat pada object tersebut disebut dengan properti. Setiap objek memiliki kemampuan untuk melakukan suatu aksi. Aksi yang dapat dilakukan oleh sebuah objek disebut dengan method.

### Class

Termasuk types, namun tidak termasuk tipe data simple. Sebuah struktur data yang berisi data dna fungsi. Merupakan struktur sintaksis bernama yang menggambarkan behaviour dan attributesnya. Dapat menambahkan method dalam class. Merupakan blue print dari sebuah objek. Class terdiri dari attributes (data/field) dna behaviour(methods).

Class statement

Attribute (opsional)  menyimpan informasi tambahan deklaratif, contoh :

[webmethod]

Modifiers (optional)  kode pengubah akses, contoh private

Class identifier  nama class

Base list (optional)  daftar yang berisi base class dna implementasi interface, dipisahkan koma

Class body  deklarasi isi/anggota class

### Object

Merupakan hasil cetakan dari sebuah class. Untuk membuat sebuah objek class, menggunakan keyword new.

Objek merupakan bentuk nyata (instance) dari kelas. Dalam kelas sudah terdapat informasi berupa properti dan tingkah laku object. Properti berupa variabel kelas yang digunakan untuk menyimpan data dan tingkah laku berupa fungsi yang dapat dijalankan.



Identity : object dapat dibedakan satu dengan yang lain

Behavior : object memiliki perilaku tertentu

State (attribute) : keadaan dari suatu object

### Object Instance

Proses membuat instan/objek dari class tertentu.

Syntax :

NamaClass namaObject = new NamaClass();

### Constructor

* Method yang memiliki nama sama dengan class dan tidak memiliki nilai kembali.
* Digunakan untuk menginisialisasi data dari object yang kita dibuat.

Struktur penulisan constructor:



Attributes (opsional)  menyimpan informasi tambahn deklaratif, contoh :

[WebMethod]

Modifiers (optional)  kode pengubah akses, contoh: private

Identifier  nama yang sama dengan nama class

Formal parameter list (opsional)  parameter opsional melalui constructor

Initializer (opsional)  kode yang dipanggil sebelum menjalankan body

Construct body  blok yang berisi statement yang menginisialisasi object

**class Customer1**

{

static void Main()

{

Customer customer = new Customer("Paul");  Console.WriteLine("The customer's name is {0}", customer.Name);

}

}

**class Customer**

{

private string name;

public Customer(string name)

{ this.Name = name;

}

public string Name { get { return name; } }

}

### Destructor

Digunakan untuk menghapus suatu object dan melakukan pembersihan, seperti memutuskan koneksi dari internet atau menutup file.

Destructor tidak bersifat statis, tidak diwariskan, tidak mengambil parameter, dan tidak menggunakan acces modifier.

Deklarasi destructor menggunakan lambang tilde(~)

**class Customer**

{ private string name;

public Customer(string name)

{ this.Name = name;

}

public string Name

{ get { return name; } }

**~Customer()**

{

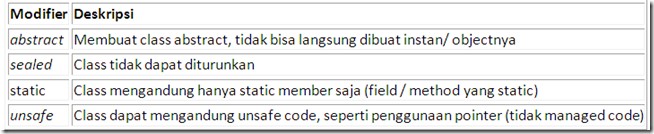
Console.WriteLine("In Customer's destructor.");

} }

### Encapsulation

* Merupakan proses pengelompokkan data dan fungsionalitas dengan pemberian hak akses, sehingga hanya dapat diakses melalui method accessor.
* Merupakan kombinasi data dan fungsionalitas dalam sebuah unit tunggal sebagai bentuk untuk menyembunyikan detail informasi.
* Memudahkan penggunaan sebuah objek dari suatu kelas karena kita tidak perlu mengetahui segala hal secara rinci.

### Modifier Class



### Access Modifier

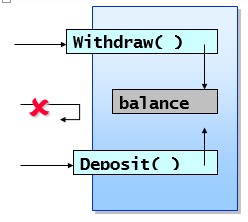
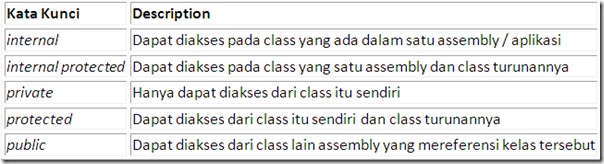
•

Methods den

gan akses public, diakses dari luar class.

•

Data dengan akses private, diakses dari luar class.



### Property

* Seperti field, kode dari sebuah object yang bersangkutan.
* Secara internal, menggunakan method accessor untuk mendapatkan dan mengatur data mereka.
* Dapat memberikan kode tipe data untuk membatasi data yang ditulis maupun dibaca dari sebuah property.

class Customer1 class Customer

{

{  private string name;

static void Main() public string Name

{ {

Customer customer = new Customer(); get{  customer.Name = "Nancy";

return name;

System.Console.WriteLine("Th e customer's name is {0}", }

customer.Name); set

} {

name = value;

}

}

}

}

### Get Accessor

* Penggunaan get{} pada property mirip penggunaan method getValue() / getter method pada java
* Get pasti mengembalikan nilai jadi harus ada return valuenya.

### Set Accessor

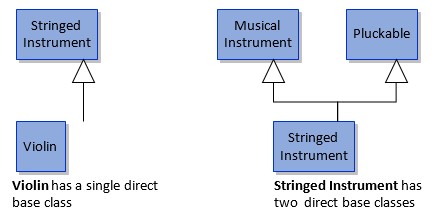
* Penggunaan set{} pada property mirip dengan setValue() / setter method pada java
* Set digunakan untuk memberi nilai ke dalam class variable.
* Pada C# jika memberi nilai pada property maka set{} otomatis dijalankan.

### Inheritance

* Penggunaan set{} pada property mirip dengan setValue() / setter method pada java
* Set digunakan untuk memberi nilai ke dalam class variable.
* Pada C# jika memberi nilai pada property maka set{} otomatis dijalankan.

### Single & Multiple Inheritance

* Single inheritance : berasal dari satu base class
* Multiple inheritance : berasal dari dua atau lebih base class



### Polymorphism

* Menimpa (override) suatu method, yang berasal dari parent class (super class) dimana object tersebut diturunkan, sehingga memiliki kelakuan yang berbeda.
* Terbagi menjadi dua, yaitu: overloading dan overriding

### Overriding Method

* Kemampuan suatu subkelas untuk memodifikasi attribute dan method milik kelas induknya.
* Override merupakan pendefinisian ulang suatu metode oleh subclass.
* Syarat Override yaitu nama method, return type, dan parameter harus sama. Jika tidak sama maka bukan dianggap sebagai override tetapi method yang baru pada subclass.
* Perbedaan utama antara Overloading dengan Overriding adalah pada pemakaian parameter.

class Over

{

public int satu, dua;

public void set\_input(int a, int b)

{

satu = a; dua = b;

}

public virtual void show()

{

Console.WriteLine("Perkalian");

Console.WriteLine("Hasil perkalian {0} dan {1} = {2}", satu, dua, satu \* dua);

}

}

class Ride : Over

{

public override void show()

{ Console.WriteLine("Pembagian");

Console.WriteLine("Hasil pembagian {0} dan {1} = {2}", satu, dua, satu / dua);

}

}

### Overloading Method

* Suatu keadaan dimana beberapa method memiliki nama yang sama tetapi parameternya berbeda.
* Deklarasi method overloading berbeda jumlah atau tipe data parameternya.
* Membuat banyak method dengan nama yang sama, namun berbeda parameter.

class ch03\_12

{

static void Main()

{

Console.WriteLine("Here's the area of the square: {0}",

Math2.Area(10));

Console.WriteLine("Here's the area of the rectangle: {0}", Math2.Area(10, 5));

}

}

public class Math2

{

// This one's for squares public static double Area(double side)

{

return side \* side;

}

// This one's for rectangles

public static double Area(double length, double height)

{

return length \* height;

}

}

### Abstract

* Syntax yang menunjukkan bahwa semua class yang diturunkan dari class yang mempunyai abstract method harus mengimplementasikan method tersebut.
* Abstract method tidak mempunyai implementation.
* Tidak bisa membuat instan dari abstract class abstract public class Window

{

abstract public void Open();

}

public class Menu : Window

{ private string openingMessage = "Displaying items...";

public override void Open()

{

System.Console.WriteLine(openingMessage);

}

}

### Interface

* Hampir sama dengan abstract class, namun abstract class hanya bisa digunakan di class paling atas ‘top-hierarchy’ sedangkan interface dapat digunakan di setiap class yang diinginkan.
* Hanya berisi operasi, tidak ada implementasi. • Tidak bisa membuat instantiasi dari interface.
* Setiap class dapat mengimplementasikan lebih dari satu interface.

## 4. PERCOBAAN

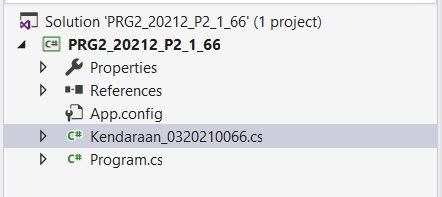
### PERCOBAAN 1

**Tugas:**

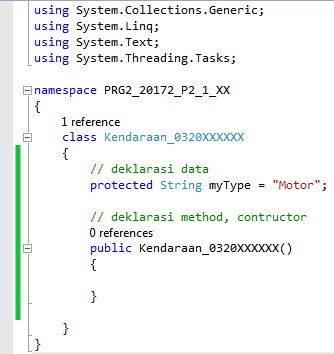
Buatlah program sesuai dengan tutorial berikut. Perhatikan, setiap penamanaan file menggunakan digit akhir NIM, hanya pada contoh tidak menggunakan NIM.

**Bagian I:** Program OOP C# sederhana

1. Buat project aplikasi console dengan nama PRG2\_20202\_P2\_1\_XX (XX adalah 2 digit terakhir NIM Anda).
2. Klik kanan PRG2\_20202\_P2\_1\_XX di solution explorer, Add >> Class
3. Beri nama kelas dengan Kendaraan\_0320XXXXXX. Maka akan terbuat sebuah file Kendaraan\_0320XXXXXX.cs di solution explorer.



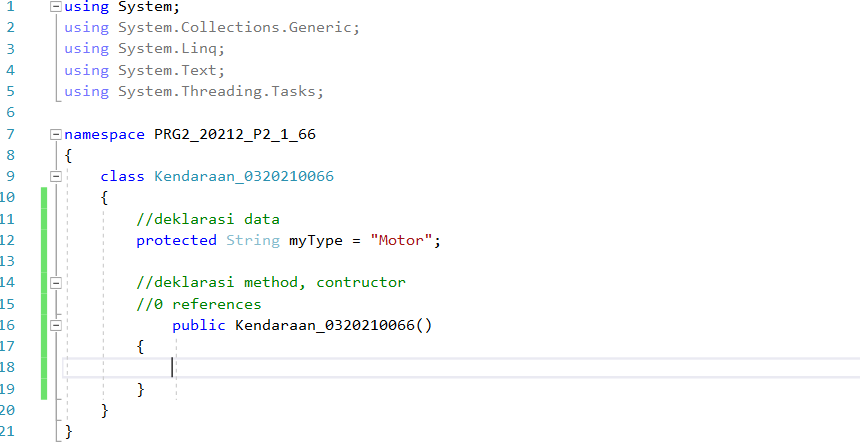
1. Ketikkan program pada Kendaraan\_0320XXXXXX.cs sehingga berbentuk sebagai berikut:



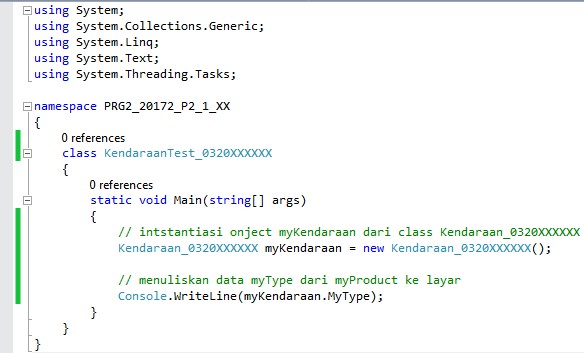
PRG2\_2020

2

\_P2\_1\_XX



1. Build lah project tersebut (CTRL+Shift+B)
2. Rename Program.cs menjadi KendaraanTest\_0320XXXXXX.cs di solution explorer
3. Buka file KendaraanTest\_0320XXXXXX.cs dan ketikan kode berikut :

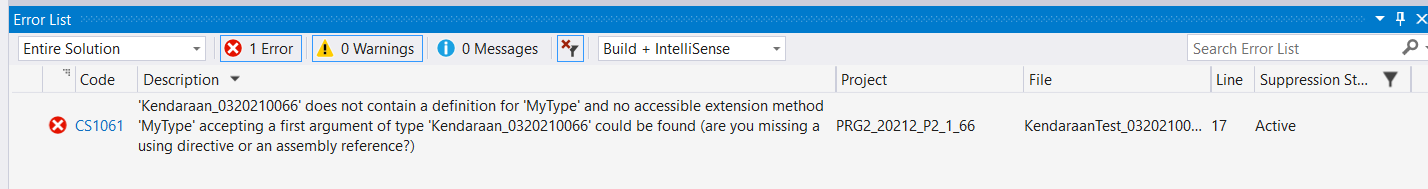


PRG2\_2020

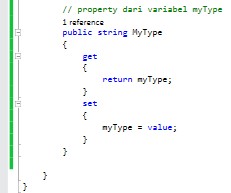
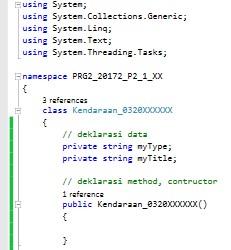
2

\_P2\_1\_XX

1. Build lah project tersebut.
2. Perhatikan error message yang muncul ketika build



1. **Jelaskan mengapa pesan kesalahan tersebut muncul? Apakah istilah dalam OOP yang mendefinisikan fenomena seperti ini?**
2. Ubahlah kode pada kelas Kendaraan\_0320XXXXXX sehingga seperti berikut: (perhatikan: variable/data MyType diubah menjadi myType, dan dibuat menjadi property)

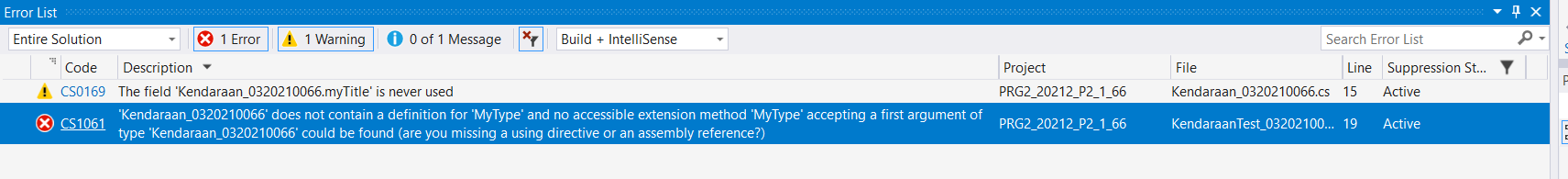


PRG2\_2020

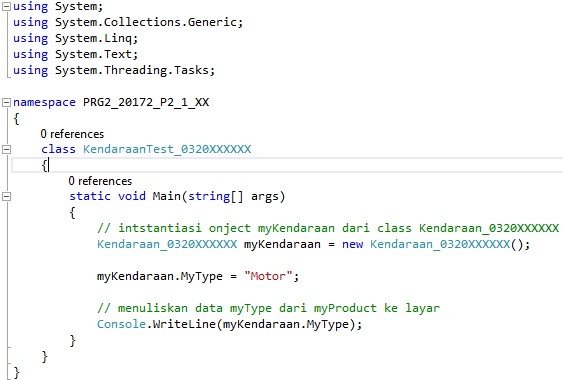
2

\_P2\_1\_XX

1. Bulid lah project tersebut.
2. Pilih menu Debug >> Start Without Debugging (CTRL+F5)
3. **Perhatikan hasil yang muncul di layar, mengapa demikian ?**



1. Tambahkan kode pada KendaraanTest\_0320XXXXXX.cs sehingga berbentuk sbb:



PRG2\_2020

2

\_P2\_1\_XX

1. Build lah project tersebut.
2. Tekan CTRL+F5
3. Perhatikan hasil yang muncul di layar

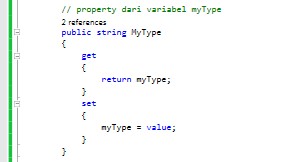
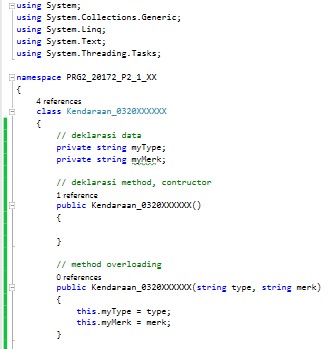


1. **Jelaskan mengapa yang muncul dilayar adalah kata tersebut?**

### Percobaan 2

**Bagian II:** Motor & Mobil Class

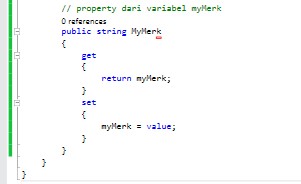
1. Ubahlah Kendaraan\_0320XXXXXX.cs seperti berikut:



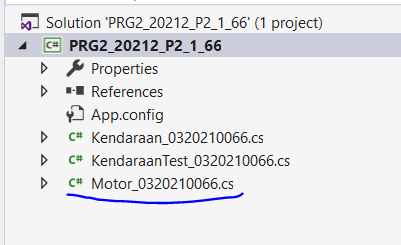
PRG2\_2020

2

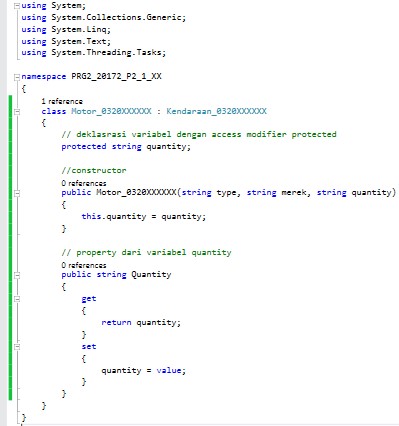
\_P2\_1\_XX



1. Tambahkan kelas Motor\_0320XXXXXX (Add >> Add Class), sehingga berbentuk Motor\_0320XXXXXX.cs



1. Ketikkan kode berikut:

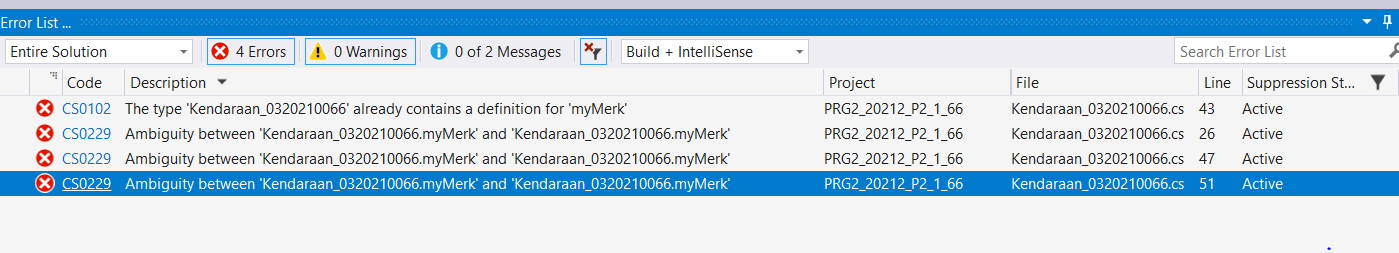


PRG2\_2020

2

\_P2\_1\_XX

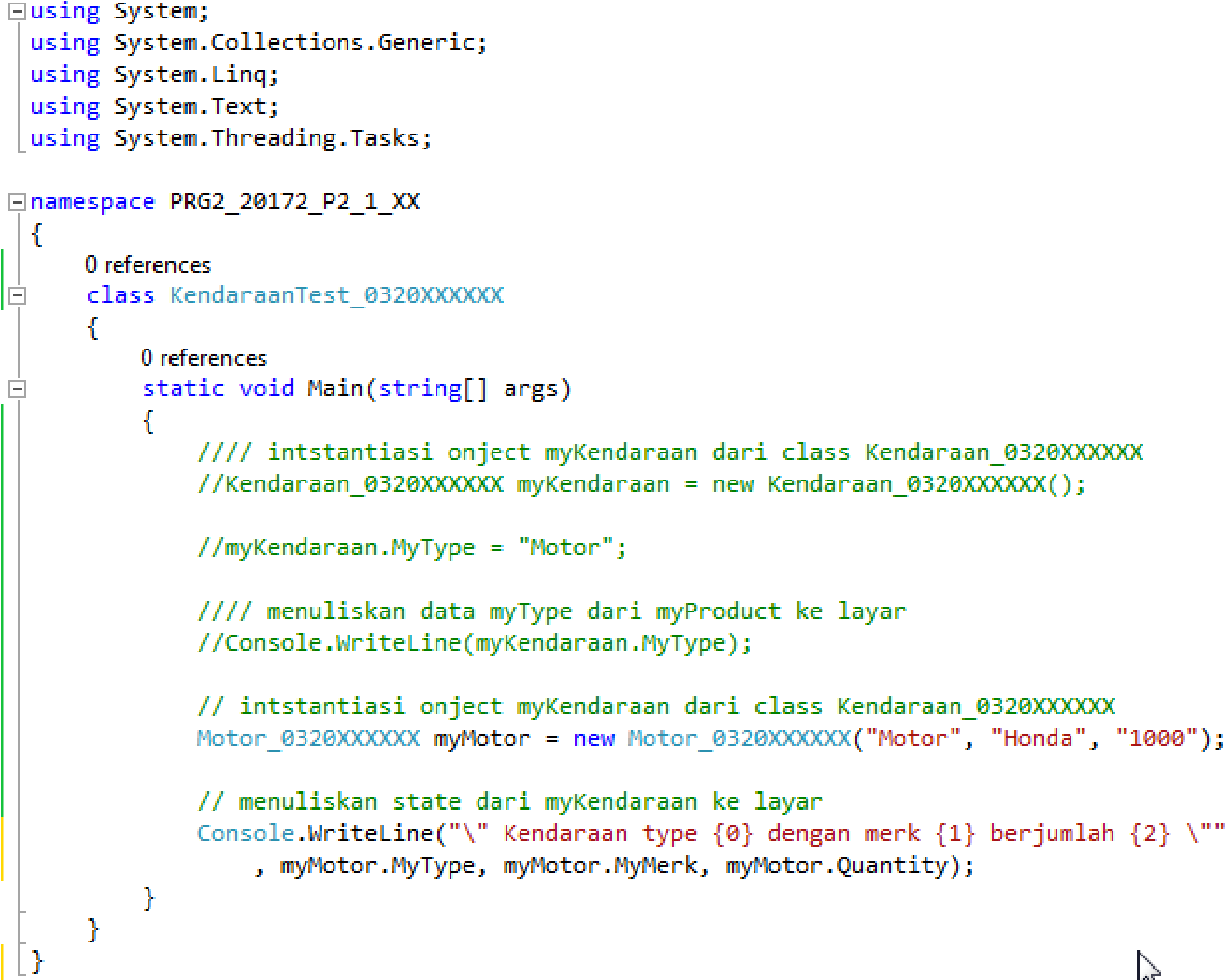
1. Build lah project tersebut.



1. Ubah kode pada kelas KendaraanTest\_0320XXXXXX Anda sehingga berbentuk :
2. Build lah project tersebut.
3. Pilihmenu Debug >> Start Without Debugging
4. Perhartikan hasil yang muncul di layar
5. **Jelaskan mengapa type dan merk dari motor tersebut tidak muncul?**

**Pakah istilah konsep OOP yang mendefinisikan fenomena seperti ini?**

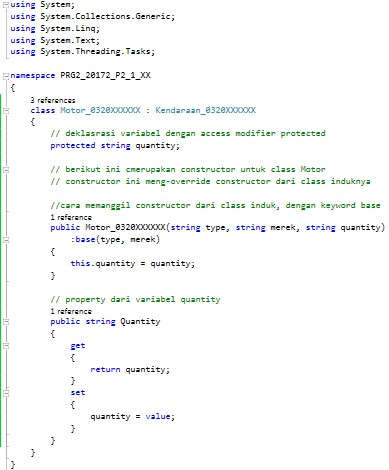
1. Ubahlah Motor\_0320XXXXXX.cs Anda sehingga sbb:



PRG2\_2020

2

\_P2\_1\_XX

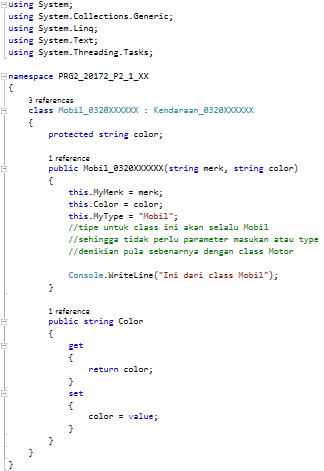


PRG2\_2020

2

\_P2\_1\_XX

1. Build lah project tersebut.
2. Pilih menu Debug >> Start Without Debugging
3. Perhatikan hasil yang muncul di layar
4. Tambahkan kelas Mobil\_0320XXXXXX, sehingga terbentuk file Mobil\_0320XXXXXX.cs
5. Ketikan kode berikut:



PRG2\_2020

2

\_P2\_1\_XX

1. Ubahlah KendaraanTest\_0320XXXXXX.cs Anda sehingga menjadi sbb:

PRG2\_2020

2

\_P2\_1\_XX



1. Build lah project tersebut.
2. Pilih menu Debug >> Start Without Debugging
3. Perhatikan hasil yang muncul di layar

## 5. LATIHAN

### Latihan 1

1. Buatlah sebuah program sederhana yang mencakup keseluruhan konsep OOP seperti pada percobaan
2. Beri nama project Anda dengan format PRG2\_21222\_P2\_1\_XXX.zip (XX adalah 3 digit terakhir NIM Anda)
3. Perhatikan, tidak boleh ada tema yang sama! **Catatan:**

* Buat folder PRG2\_21222\_P2\_XXX (XX adalah 3 digit terakhir dari NIM Anda)
* Masukkan semua folder hasil kerja Anda ke dalam folder tersbut.
* Kumpulkan file dan folder project tersebut setelah dikompresi dalam bentuk zip.