Part01

- **Problem**:
  - Define a class `Car` with properties `Id`, `Brand`, and `Price`.
  - Write multiple constructors:
    1. Default constructor.
    2. Constructor with one parameter (Id).
    3. Constructor with two parameters (Id, Brand).
    4. Constructor with all three parameters.
  - Demonstrate the constructors by creating objects.
- **Question**: Why does defining a custom constructor suppress the default constructor in C#?

- **Problem**:
  - Write a class `Calculator` with overloaded `Sum()` methods to:
    1. Add two integers.
    2. Add three integers.
    3. Add two doubles.
  - Write a program to test each overload.
- **Question**: How does method overloading improve code readability and reusability?

- **Problem**:
  - Create a base class `Parent` with properties `X` and `Y`, and a constructor to initialize them.
  - Create a derived class `Child` with an additional property `Z`, and chain its constructor to the base class.
  - Demonstrate constructor chaining by creating an instance of `Child`.
- **Question**: What is the purpose of constructor chaining in inheritance?

- **Problem**:
  - Define a method `Product()` in the `Parent` class to return `X * Y`.
  - In the `Child` class:
    1. Override the `Product()` method using the `new` keyword.
    2. Override it using the `override` keyword.
  - Demonstrate the difference in behavior using an instance of `Child`.
- **Question**: How does `new` differ from `override` in method overriding?

- **Problem**:
  - Override the `ToString()` method in `Parent` to return `(X, Y)` and in `Child` to return `(X, Y, Z)`.
  - Demonstrate polymorphism by printing instances of both `Parent` and `Child`.
- **Question**: Why is `ToString()` often overridden in custom classes?

- **Problem**:
  - Define an interface `IShape` with:
    1. A property `Area` (get-only).
    2. A method `Draw()`.

    - Create a class `Rectangle` implementing `IShape` with its own version of `Draw()` and `Area`.
    - Test the implementation.
- **Question**: Why can't you create an instance of an interface directly?

- **Problem**:
    - Modify the `IShape` interface to include a default implementation of a method `PrintDetails()`.
    - Create a class `Circle` that implements `IShape`.
    - Call `PrintDetails()` on an instance of `Circle`.
- **Question**: What are the benefits of default implementations in interfaces introduced in C# 8.0?

- **Problem**:
    - Define an interface `IMovable` with a method `Move()`.
    - Create a class `Car` implementing `IMovable`.
    - Use an `IMovable` reference to access the `Car` object and call `Move()`.
- **Question**: Why is it useful to use an interface reference to access implementing class methods?

- **Problem**:
    - Create two interfaces, `IReadable` and `IWritable`, each with a method (`Read()` and `Write()`).
    - Create a class `File` that implements both interfaces.
    - Demonstrate using the `File` class.
- **Question**: How does C# overcome the limitation of single inheritance with interfaces?

- **Problem**:
    - Create a base class `Shape` with:
        1. A virtual method `Draw()` that prints "Drawing Shape".
        2. An abstract method `CalculateArea()` for area calculation.
    - Create a derived class `Rectangle` overriding `Draw()` and implementing `CalculateArea()`.
    - Demonstrate the usage with objects of `Rectangle`.
- **Question**: What is the difference between a virtual method and an abstract method in C#?

Part02

What is the difference between class and struct in C#?

If inheritance is relation between classes clarify other relations between classes

Part03 Bonus

1- self study report

2- what is static and dynamic binding