ABSTRACTION

Abstraction is a concept of representing objects in a simplified way by focusing only on the essential features and ignoring the irrelevant details.
It provides a blueprint or a contract for derived classes to follow, ensuring consistency in your code.

Benefit

Makes the program loosely coupled (makes the classes less dependent on the other).

Implemented using

- Abstract classes [or]
- Interfaces

ABSTRACTION

Abstract Classes:

Abstract class is a parent class, for which, we can't create object; but we can create child classes.

- The main intention of abstract class is to provide common set of fields and methods to all of its child classes of a specific group.
- Abstract class can contain all types of members (fields, properties, methods, constructors etc.).
- We can't create object for abstract class; but we can access its members through child class's object.
- So 'creating child class of abstract class' is the only-way to utilize abstract classes.
- Use Abstract class concept, for the classes, for which, you feel creating object is not meaningful.
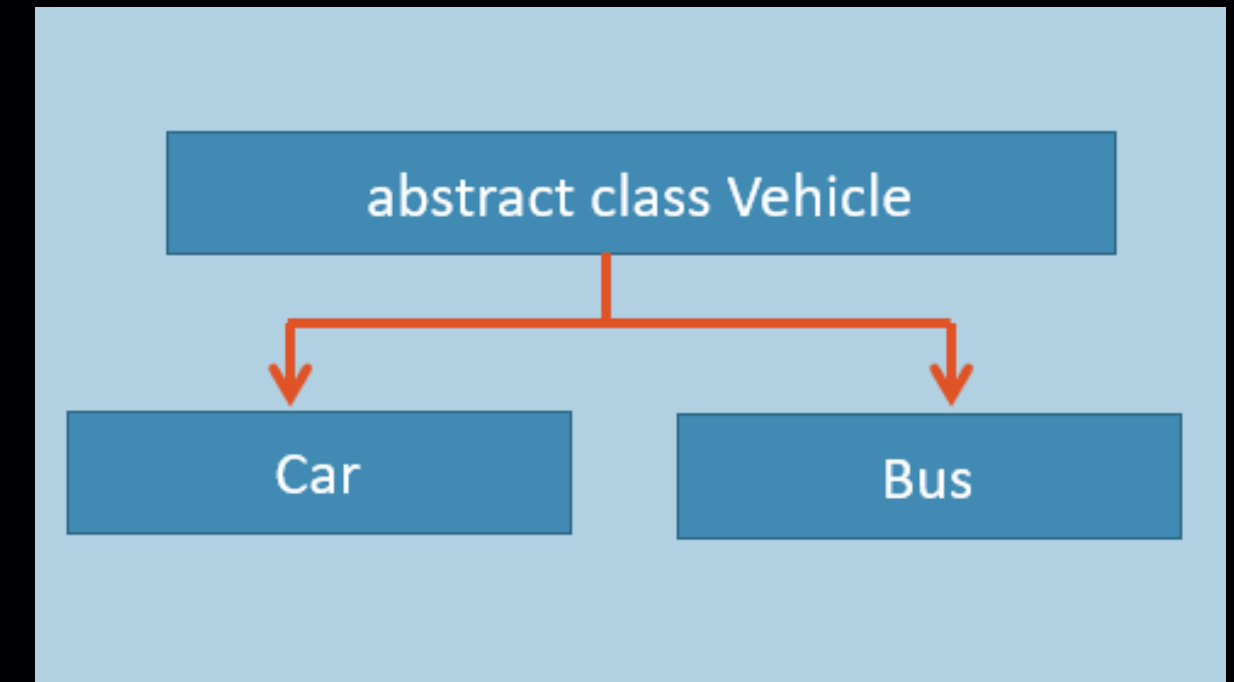
| Class Type	Can Inherit from Other Classes	Can Inherit from Other Interfaces	Can be Inherited	Can be Instantiated
Normal Class	Yes	Yes	Yes	Yes
Abstract Class	Yes	Yes	Yes	No

ABSTRACTION

Abstract Methods

Abstract methods are declared in parent class, with "abstract" keyword; implemented in child classes, with "override" keyword.
When the parent class don't want to provide the definition of a method;it wants to let child classes to implement the method.

Abstract Methods contain "method declaration" only; but not "method body".
Child class must provide method body for abstract methods.
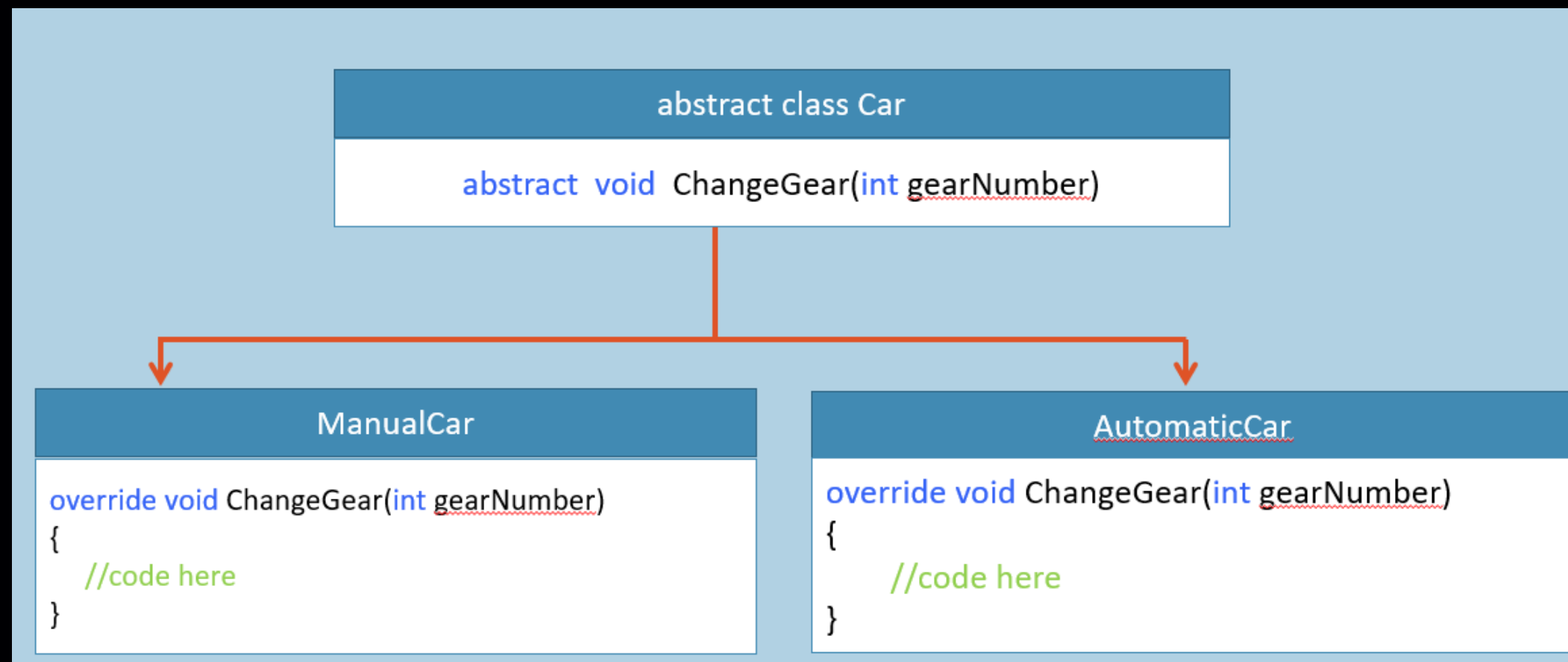
ABSTRACTION

Interfaces

Interface is a set of abstract methods, that must be implemented by the child classes.

- The child class that implements the interface, MUST implement ALL METHODS of the interface.
- Interface methods are by default "public" and "abstract".
- The child class must implement all interface methods, with same signature.
- You can't create object for interface.
- You can create reference variable for the interface.
- The reference variable of interface type can only store the address of objects of any one of the corresponding child classes.
- You can implement multiple interfaces in the same child class [Multiple Inheritance].
- An interface can be child of another interface.
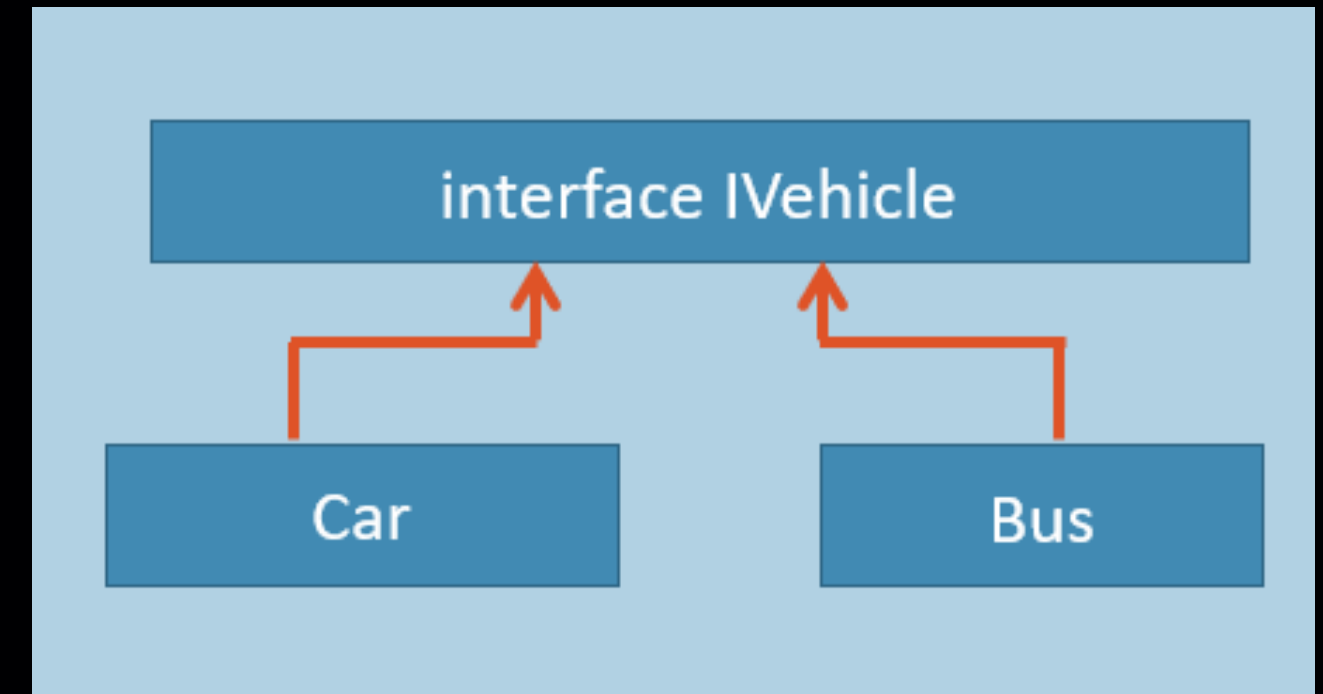
ABSTRACTION

Interfaces

Comparison Table: Abstract Class (vs) Interface

Class Type	Can Inherit from Other Classes	Can Inherit from Other Interfaces	Can be Inherited	Can be Instantiated
Normal Class	Yes	Yes	Yes	Yes
Abstract Class	Yes	Yes	Yes	No
Interface	No	Yes	Yes	No

Based on members:

Type	1. Non-Static Fields	2. Non-Static Methods	3. Non-Static Constructors	4. Non-Static Properties	5. Non-Static Events	6. Non-Static Destructors	7. Constants
Normal Class	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Abstract Class	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Interface	No	No	No	No	Yes	No	No

Type	8. Static Fields	9. Static Methods	10. Static Constructors	11. Static Properties	12. Static Events	13. Virtual Methods	14. Abstract Methods	15. Non-Static Auto-Impl Properties	16. Non-Static Indexers
Normal Class	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes
Abstract Class	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Interface	No	No	No	No	No	No	Yes	Yes	No

ABSTRACTION

Polymorphism

Polymorphism provides the ability to the developer, to define different implements for the same method in the same class or different classes.

Compile-time polymorphism:

- Eg: Method Overloading
- Decission will be taken at compilation time.
- Also known as "Early binding" / "Static polymorphism".

Run-time polymorphism:

- Eg: Method Overriding
- Decission will be taken at run time.
- Also known as "Late binding" / "Dynamic polymorphism".
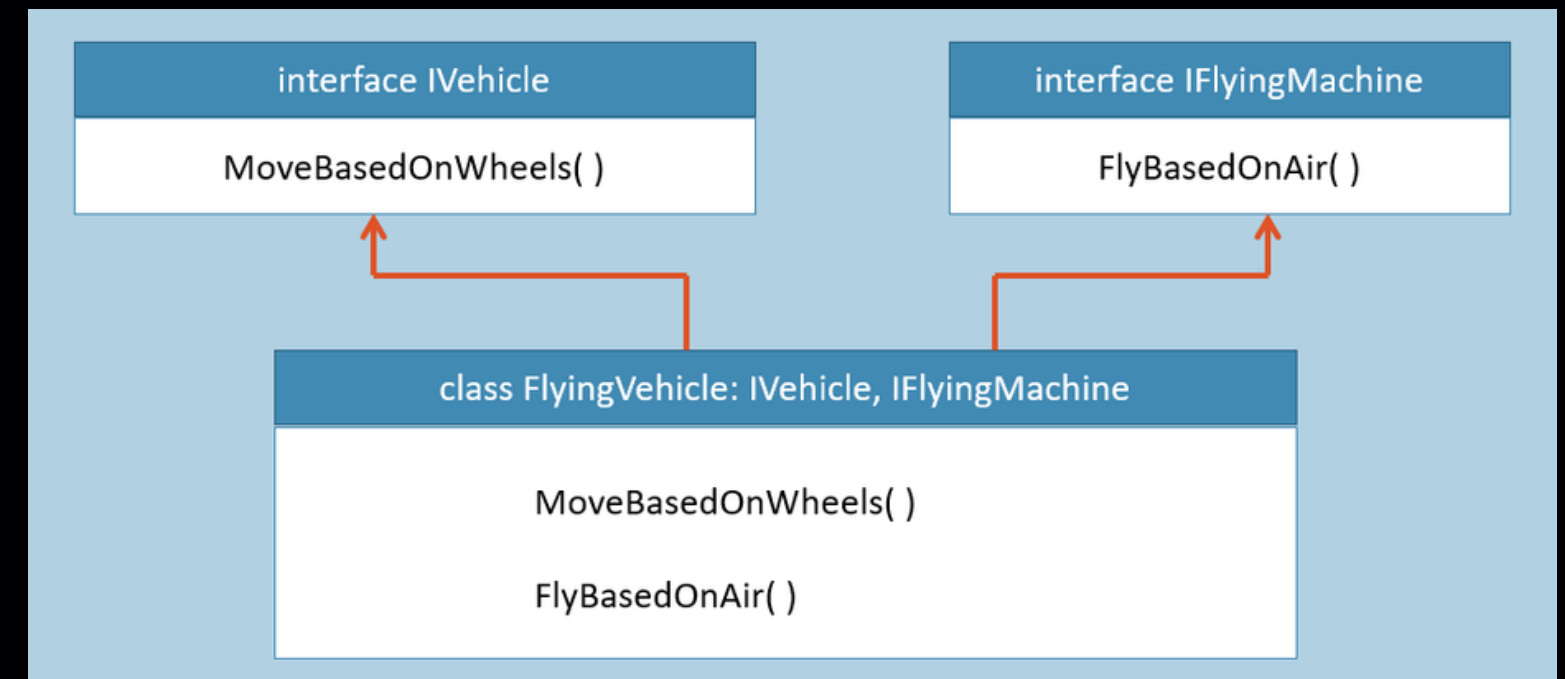
ABSTRACTION

Multiple Inheritance

In C#, "multiple inheritance" IS POSSIBLE with INTERFACES; that means a child class can have multiple parent interfaces.

"One Child - Multiple Parent Classes / Parent Interfaces" is called as "Multiple Inheritance".

In C#.NET, "multiple inheritance" IS NOT POSSIBLE with CLASSES; that means you can't specify multiple parent classes.

The child class MUST IMPLEMENT all methods of all the interfaces, that are inherited from.
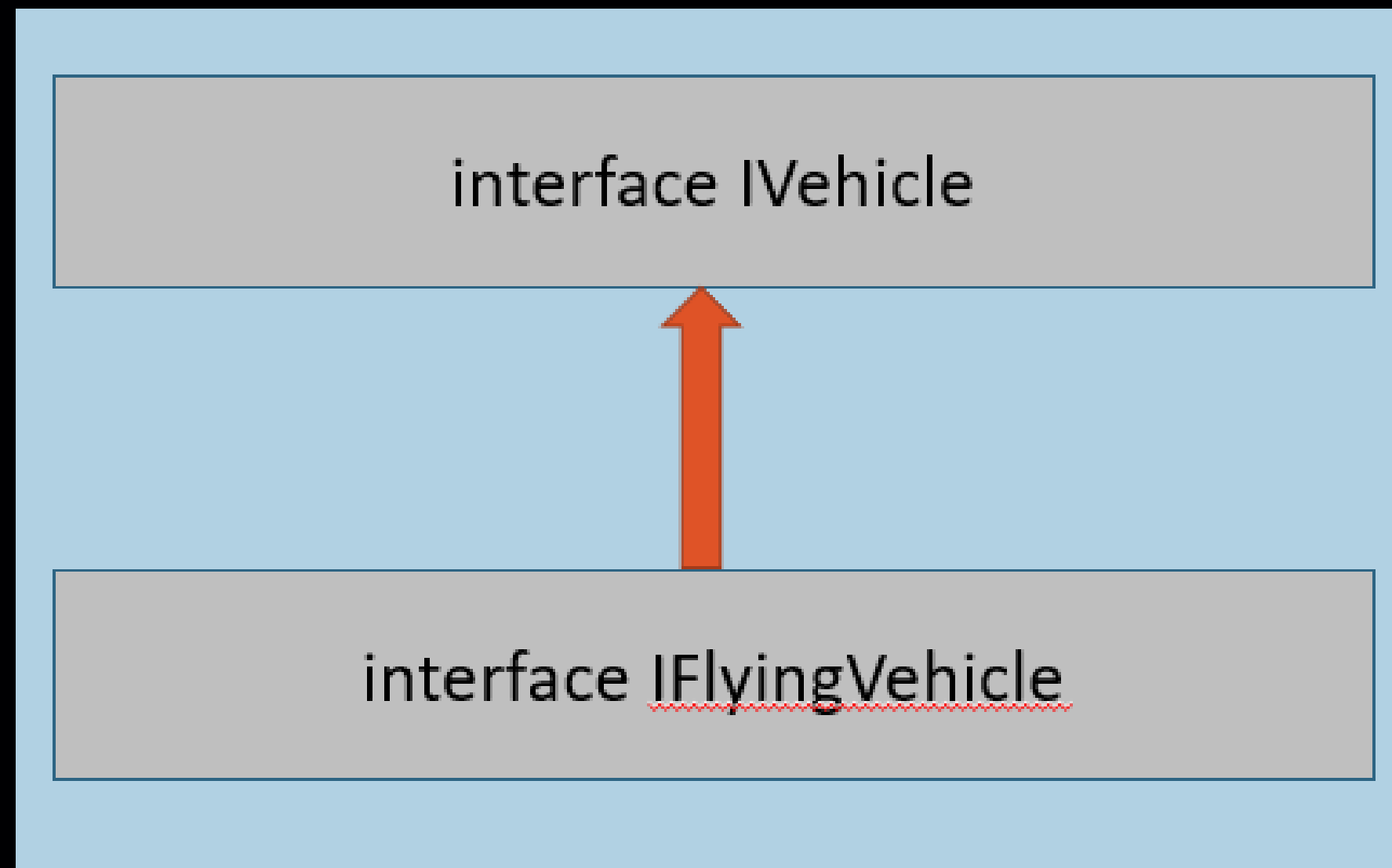
ABSTRACTION

Interface Inheritance

If an interface inherits from another interface, we call it as "Interface Inheritance".

The child class that implements the child interface must implement all the members of both parent interface and child interface too.

ABSTRACTION

Explicit Interface Implementation

"Explicit Interface Implementation" is used to implement an interface method privately; that means the interface method becomes as "private member" to the child class.

If a child class inherits from two or more interfaces, and there is a duplicate method (having same name and parameters) among those interfaces; then use "Explicit Interface Implementation", to provide different implementations for different interface methods respectively.

You can use "Explicit Interface Implementation" to create private implementation of interface method; so that you can create abstraction for those methods.