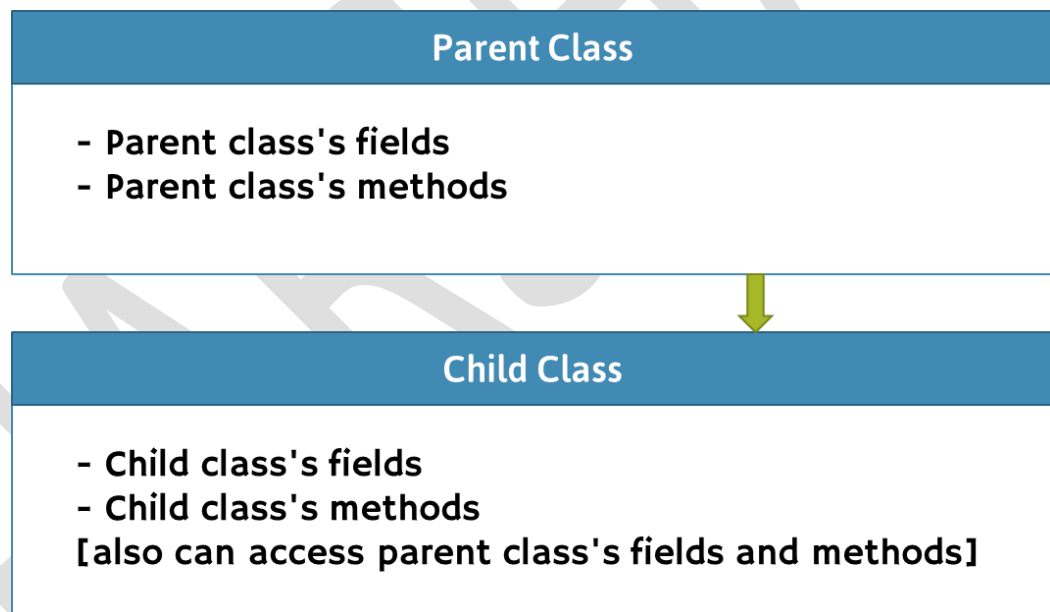


14

Inheritance

Introducing Inheritance

- › This allows classes to be arranged in a hierarchy that represents "is-a-type-of" relationships.
- › The "parent class" acts as a "base type" of "one or more child classes".
- › Child classes are derived from parent class.



- › Concept of extending the parent class, by creating child class.
- › "Child class" extends "parent class".
- › The child class's object stores members of both child class and parent class.

Creating Parent Class

```
class ParentClassName
{
    Parent Class Members here
}
```



Creating Child Class

```
class ChildClassName : ParentClassName
{
    Child Class Members here
}
```

Object of Child Class

Parent Class's
Fields

Child Class's
Fields

Types of Inheritance – Single Inheritance

I. Single Inheritance

```
class ParentClassName
{
}

class ChildClassName: ParentClassName
{
}
```

One Parent Class, One Child Class.

Types of Inheritance – Multi-Level Inheritance

2. Multi-Level Inheritance

```
class ParentClassName
{
}

class ChildClass1: ParentClassName
{
}

class ChildClass2: ChildClass1
{
}
```

One Parent Class, One Child Class;
and the Child class has another Child class.

Types of Inheritance – Hierarchical Inheritance

3. Hierarchical Inheritance

```
class ParentClassName
{
}

class ChildClass1 : ParentClassName
{
}

class ChildClass2 : ParentClassName
{
}
```

One Parent Class, Multiple Child Classes.

Types of Inheritance – Multiple Inheritance

4. Multiple Inheritance

```
class ParentClass1
{
}

class ParentClass2
{
}

class ChildClass : ParentClass1, ParentClass2
{
}
```

Multiple Parent Classes, One Child class.

Types of Inheritance – Hybrid Inheritance

5. Hybrid Inheritance

```
class ParentClassName
{
}

class ChildClass1 : ParentClassName
{
}

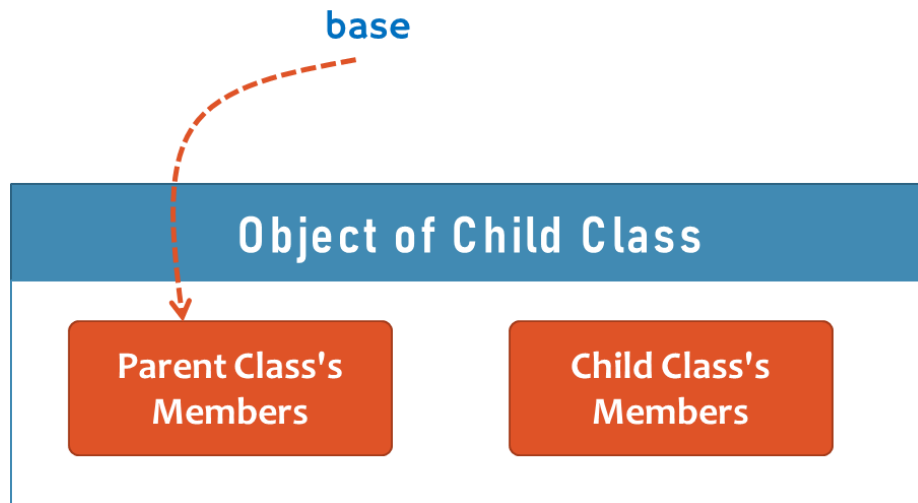
class ChildClass2 : ChildClass1
{
}

class ChildClass3 : ChildClass1
{
}
```

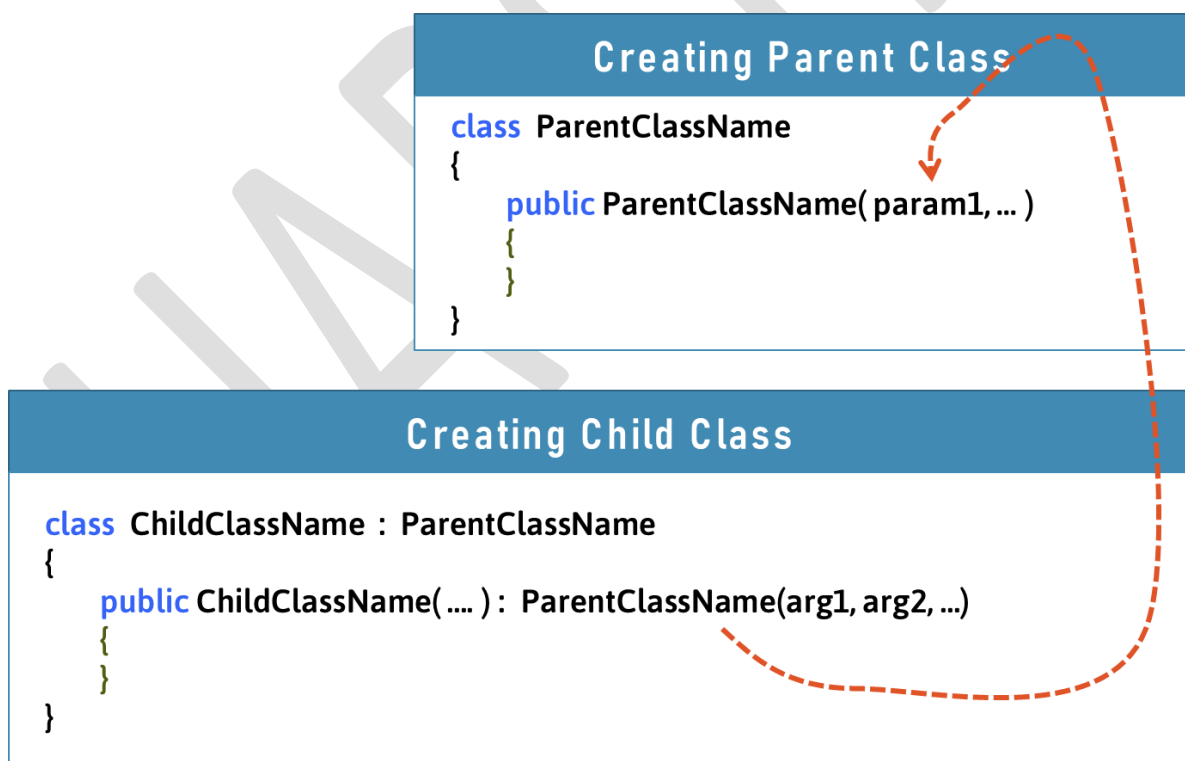
Hierarchical Inheritance + Multi Level Inheritance

'base' keyword

- › The "base" keyword represents parent class's members in the child class.
- › It is optional to use, by default.
- › It is must to use, when there is "name ambiguity" between parent class's member and child class's member.



Parent Class's Constructor




- › It is OPTIONAL to call "Parent Class's Parameter-less Constructor" from "Child Class".
- › It is MUST to call "Parent Class's Parameterized Constructor" from "Child Class" and pass necessary arguments.

Method Hiding

- › It is a concept, which is used to hide (overwrite) the parent class's method, by creating another method in the child class with same name and same parameters.

Creating Parent Class

```
class ParentClassName
{
    public void MethodName( param1, ... )
    {
    }
}
```



Creating Child Class

```
class ChildClassName : ParentClassName
{
    public new void MethodName( param1, ... )
    {
    }
}
```


- › When method hiding is done, if the method is called using child class's object; the child class's method only executes; parent class's method will not be executed.
- › Method hiding is done automatically; but is recommended to use "new" keyword (but not must).

Method Overriding

- › It is a concept, which is used to extend the parent class's method, by creating another method in the child class with same name and same parameters.


Creating Parent Class

```
class ParentClassName
{
    public virtual void MethodName( param1, ... )
    {
    }
}
```



Creating Child Class

```
class ChildClassName : ParentClassName
{
    public override void MethodName( param1, ... )
    {
        base.MethodName();
    }
}
```



- › When method overriding is done, if the method is called using child class's object; the parent class's method first and child's method executed next.
- › Method Overriding is done with "virtual" keyword at parent class; and "override" keyword at child class's method.
- › The parent class's method invoked using "base" keyword.
- › Without 'virtual' keyword are parent class's method; the child class's method can't be 'override'.

Points to Remember

- › The "parent class" acts as a "base type" of "one or more child classes".
- › The 'base' keyword inside the child class, can access the members of parent class.

- › The child class's object stores parent class's fields also, automatically; however they are accessible in child class if they are not 'private'.
- › Method hiding is a concept of 'overwriting' the parent class's method, by creating another method in child class, with same signature.
- › Method overriding is a concept of 'extending' the parent class's method, by creating another method in child class, with same signature.
- › C# doesn't support multiple inheritance (with multiple parent classes). That means, a child class can have ONLY ONE parent class; however, a child class can have MULTIPLE parent interfaces; so in this way, C# supports multiple inheritance.