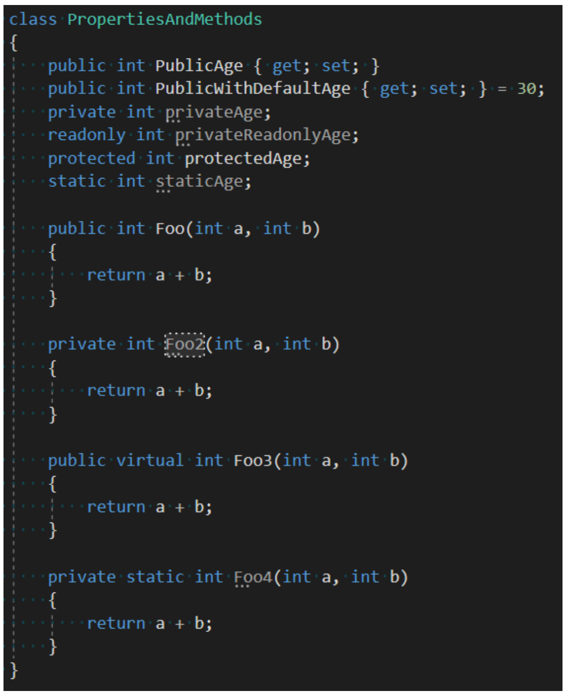
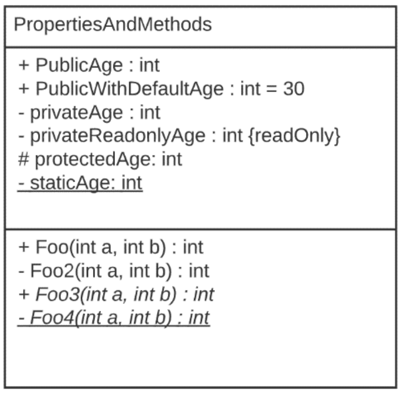
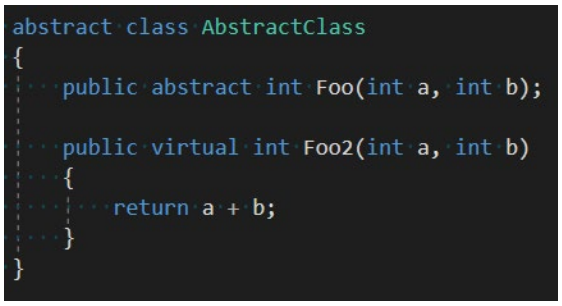
UML DIAGRAM SUMMARY

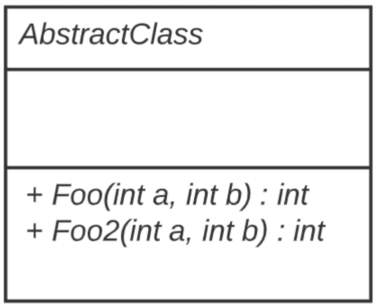


```
class PropertiesAndMethods
{
    public int PublicAge { get; set; }
    public int PublicWithDefaultAge { get; set; } = 30;
    private int privateAge;
    readonly int privateReadonlyAge;
    protected int protectedAge;
    static int staticAge;

    public int Foo(int a, int b)
    {
        return a + b;
    }

    private int Foo2(int a, int b)
    {
        return a + b;
    }

    public virtual int Foo3(int a, int b)
    {
        return a + b;
    }

    private static int Foo4(int a, int b)
    {
        return a + b;
    }
}
```


PropertiesAndMethods
+ PublicAge : int
+ PublicWithDefaultAge : int = 30
- privateAge : int
- privateReadonlyAge : int {readOnly}
# protectedAge: int
- staticAge: int
+ Foo(int a, int b) : int
- Foo2(int a, int b) : int
+ *Foo3(int a, int b) : int*
- Foo4(int a, int b) : int



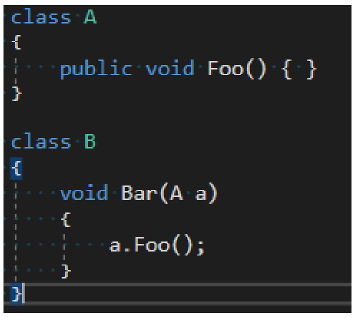
```
abstract class AbstractClass
{
    public abstract int Foo(int a, int b);

    public virtual int Foo2(int a, int b)
    {
        return a + b;
    }
}
```



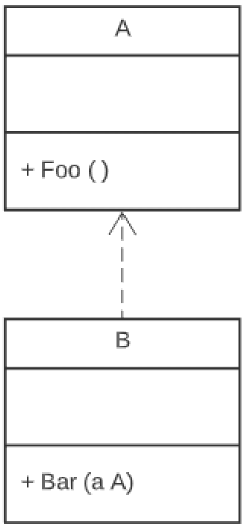
DEPENDENCY

'is dependent'

(dashed line with open arrow)



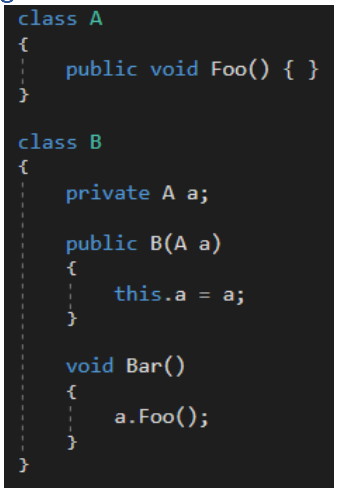
```
class A
{
    public void Foo() { }
}

class B
{
    void Bar(A a)
    {
        a.Foo();
    }
}
```



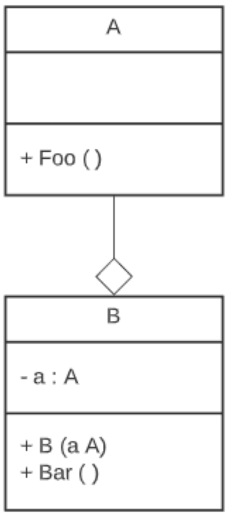
AGGREGATION

'has a'

(single line with hollow diamond)



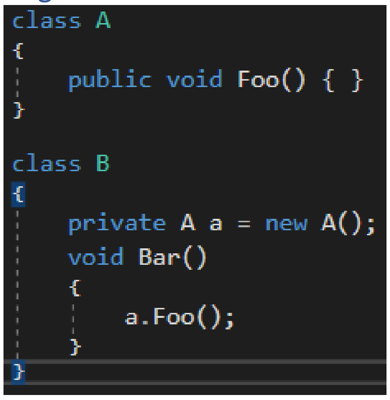
```
class A
{
    public void Foo() { }
}

class B
{
    private A a;

    public B(A a)
    {
        this.a = a;
    }

    void Bar()
    {
        a.Foo();
    }
}
```



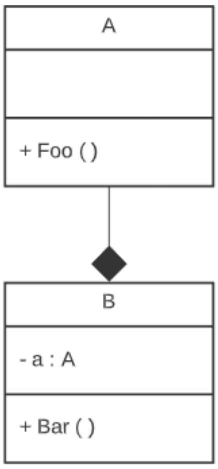
COMPOSITION

'owns a'

(single line with filled diamond)



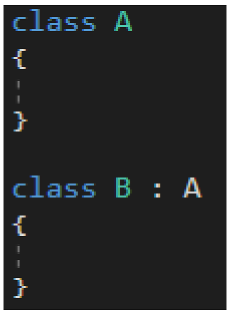
```
class A
{
    public void Foo() { }
}

class B
{
    private A a = new A();
    void Bar()
    {
        a.Foo();
    }
}
```



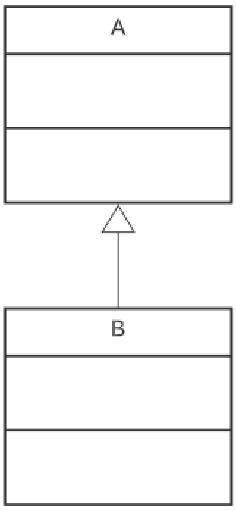
INHERITANCE

'is a'

(single line with hollow triangle)



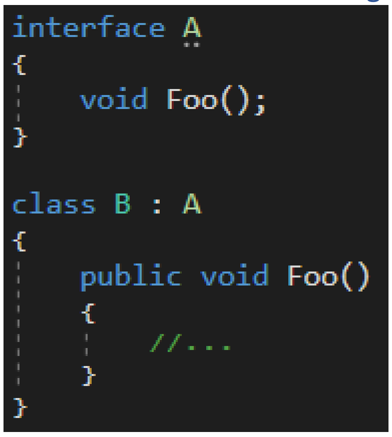
```
class A
{
}

class B : A
{
}
```



REALIZATION

'implements'

(dashed line with hollow triangle)



```
interface A
{
    void Foo();
}

class B : A
{
    public void Foo()
    {
        //...
    }
}
```