Programmering og Problemløsning

19 December 2016

Christina Lioma

c.lioma@di.ku.dk

Today's lecture

- Class inheritance
  - Overriding
  - Definition & Implementation
  - Abstract classes
  - Concrete classes

Inheritance

- *Derived* inherits all non-private members from *Base*

Inheritance

- *Derived* inherits all non-private members from *Base*
- In F#, *Derived* has only one direct *Base*

Inheritance

- *Derived* inherits all non-private members from *Base*
- In F#, *Derived* has only one direct *Base*
- If *Base* has additional constructors, the constructor(s) to be inherited must be specified

```
type Laser() =
    member x.ID = “Galaxy235”
    member x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
```

```
type Laser() =
    member x.ID = “Galaxy235”
    member x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()

let laser1 = Laser()
laser1.ShowID()                    Galaxy235
let laser2 = SpeedLaser()
laser2.ShowID()                    Galaxy235
```

```
type Laser() =
    member x.ID = “Galaxy235”
    member x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
    member x.Power = 70
    member x.ShowPower() = System.Console.Write(x.Power)
let laser1 = Laser()
laser1.ShowID()                                    Galaxy235
let laser2 = SpeedLaser()
laser2.ShowID()                                    Galaxy235
```

```
type Laser() =
    member x.ID = “Galaxy235”
    member x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
    member x.Power = 70
    member x.ShowPower() = System.Console.Write(x.Power)
let laser1 = Laser()
laser1.ShowID()                    Galaxy235
let laser2 = SpeedLaser()
laser2.ShowID()                    Galaxy235
laser2.ShowPower()                 70
```

```
type Laser() =
    member x.ID = “Galaxy235”
    member x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
    member x.Power = 70
    member x.ShowPower() = System.Console.Write(x.Power)
let laser1 = Laser()
laser1.ShowID()                    Galaxy235
let laser2 = SpeedLaser()
laser2.ShowID()                    Galaxy235
laser2.ShowPower()                 70
laser1.ShowPower()                 what does this output?
```

```
type Laser() =
    member x.ID = "Galaxy235"
    member x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
    member x.Power = 70
    member x.ShowPower() = System.Console.Write(x.Power)
let laser1 = Laser()
laser1.ShowID()                    Galaxy235
let laser2 = SpeedLaser()
laser2.ShowID()                    Galaxy235
laser2.ShowPower()                 70
laser1.ShowPower()                 "ShowPower is not defined"
```

Inheritance

- *Derived* inherits all non-private members from *Base*
- In F#, *Derived* has only one direct *Base*
- If *Base* has additional constructors, the constructor(s) to be inherited must be specified
- *Derived* can contain additional members not found in *Base.* These members are not accessed by *Base*

Inheritance

- *Derived* inherits all non-private members from *Base*
- In F#, *Derived* has only one direct *Base*
- If *Base* has additional constructors, the constructor(s) to be inherited must be specified
- *Derived* can contain additional members not found in *Base.* These members are not accessed by *Base*
- We can customise Inherited members in *Derived* (***overriding***)

Three steps to override an inherited member:

Three steps to override an inherited member:

- State in the base class that the member can be overridden

Three steps to override an inherited member:

- State in the base class that the member can be overridden
- State in the base class how the member works if it is not overridden

Three steps to override an inherited member:

- State in the base class that the member can be overridden
- State in the base class how the member works if it is not overridden
- State in the derived class how the member is overridden

Three steps to override an inherited member:

- State in the base class that the member can be overridden

  use keyword *abstract*
- State in the base class how the member works if it is not overridden

- State in the derived class how the member is overridden

Three steps to override an inherited member:

- State in the base class that the member can be overridden

  use keyword *abstract*
- State in the base class how the member works if it is not overridden

  use keyword *default*
- State in the derived class how the member is overridden

Three steps to override an inherited member:

- State in the base class that the member can be overridden

  use keyword *abstract*
- State in the base class how the member works if it is not overridden

  use keyword *default*
- State in the derived class how the member is overridden

  use keyword *override*

```
type Laser() =
    member x.ID = “Galaxy235”
    member x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
```

```
type Laser() =
    member x.ID = “Galaxy235”
    member x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
```

```
type Laser() =
    member x.ID = “Galaxy235”
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
```

```
type Laser() =
    member x.ID = “Galaxy235”
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
```

CAN BE OVERRIDEN

```
type Laser() =
    member x.ID = “Galaxy235”
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
```

CAN BE OVERRIDEN

IF NOT OVERRIDEN, USE THIS

```
type Laser() =
    member x.ID = “Galaxy235”
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
```

DEFINITION

IMPLEMENTATION

```
type Laser() =
    member x.ID = “Galaxy235”
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
```

DEFINITION

DEFINITION SYNTAX:

abstract member MemberName **:** data type

```
type Laser() =
    member x.ID = "Galaxy235"
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
```

IMPLEMENTATION

DEFINITION SYNTAX:

abstract member MemberName **:** data type

IMPLEMENTATION SYNTAX:

default selfIdentifier**.**MemberName **=** …

```
type Laser() =
    member x.ID = "Galaxy235"
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
```

DEFINITION

IMPLEMENTATION

DEFINITION SYNTAX:

abstract member MemberName **:** data type

IMPLEMENTATION SYNTAX:

default selfIdentifier**.**MemberName **=** …

unit data type: indicates the absence of a value (placeholder when no value exists / is needed)
https://msdn.microsoft.com/en-us/library/dd483472.aspx

```
type Laser() =
    member x.ID = “Galaxy235”
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
```

DEFINITION

IMPLEMENTATION

```
type Laser() =
    member x.ID = “Galaxy235”
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
    override x.ShowID() = System.Console.Write(base.ID+”.v2”)
```

DEFINITION

IMPLEMENTATION

```
type Laser() =
    member x.ID = “Galaxy235”
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
    override x.ShowID() = System.Console.Write(base.ID+”.v2”)
```

DEFINITION

IMPLEMENTATION

NEW IMPLEMENTATION

```
type Laser() =
    member x.ID = “Galaxy235”
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
    override x.ShowID() = System.Console.Write(base.ID+”.v2”)
```

“**override**” keyword: re-implements the method of the base class

“**base**” keyword: accesses directly members of the base class

```
type Laser() =
    member x.ID = “Galaxy235”
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
    override x.ShowID() = System.Console.Write(base.ID+”.v2”)

let laser1 = Laser()
laser1.ShowID()
let laser2 = SpeedLaser()
laser2.ShowID()
```

```
type Laser() =
    member x.ID = “Galaxy235”
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
    override x.ShowID() = System.Console.Write(base.ID+”.v2”)

let laser1 = Laser()
laser1.ShowID()                          Galaxy235
let laser2 = SpeedLaser()
laser2.ShowID()                          Galaxy235.v2
```

```
type Laser() =
    member x.ID = “Galaxy235”
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
    override x.ShowID() = System.Console.Write(base.ID+”.v2”)

let laser1 = Laser()
laser1.ShowID()                                  Galaxy235
let laser2 = SpeedLaser()
laser2.ShowID()                                  Galaxy235.v2
```

CAN OVERRIDE ATTRIBUTES TOO

```
type Laser() =
    member x.ID = “Galaxy235”
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
    abstract member                    :
    default x.                         =
```

```
type Laser() =
    member x.ID = “Galaxy235”
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
    abstract member                    :                    DEF
    default x.                         =                    IMPL
```

```
type Laser() =
    member x.ID = “Galaxy235”
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
    abstract member CompatibleWith : string list
    default x.CompatibleWith = []
```

DEF

IMPL

```
type Laser() =
    member x.ID = “Galaxy235”
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
    abstract member CompatibleWith : string list        DEF
    default x.CompatibleWith = []                       IMPL
type SpeedLaser() =
    inherit Laser()
    override x.ShowID() = System.Console.Write(base.ID+”.v2”)
```

```
type Laser() =
    member x.ID = “Galaxy235”
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
    abstract member CompatibleWith : string list        DEF
    default x.CompatibleWith = []                       IMPL
type SpeedLaser() =
    inherit Laser()
    override x.ShowID() = System.Console.Write(base.ID+”.v2”)
    override x.CompatibleWith = [“Space0”; “Space9”]    IMPL
```

```
type Laser() =
    member x.ID = “Galaxy235”
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
    abstract member CompatibleWith : string list          DEF
    default x.CompatibleWith = []                         IMPL
type SpeedLaser() =
    inherit Laser()
    override x.ShowID() = System.Console.Write(base.ID+”.v2”)
    override x.CompatibleWith = [“Space0”; “Space9”]      IMPL
let laser1 = Laser()
let laser2 = SpeedLaser()
System.Console.Write(laser1.CompatibleWith)
System.Console.Write(laser2.CompatibleWith)
```

```
type Laser() =
    member x.ID = "Galaxy235"
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
    abstract member CompatibleWith : string list        DEF
    default x.CompatibleWith = []                       IMPL
type SpeedLaser() =
    inherit Laser()
    override x.ShowID() = System.Console.Write(base.ID+".v2")
    override x.CompatibleWith = ["Space0"; "Space9"]    IMPL
let laser1 = Laser()
let laser2 = SpeedLaser()
System.Console.Write(laser1.CompatibleWith)  []
System.Console.Write(laser2.CompatibleWith)  [Space0; Space9]
```

```
type Laser() =
    member x.ID = “Galaxy235”
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
    override x.ShowID() = System.Console.Write(base.ID+”.v2”)

let laser1 = Laser()
let laser2 = SpeedLaser()
laser2.ShowID()
```

```
type Laser() =
    member x.ID = “Galaxy235”
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
    override x.ShowID() = System.Console.Write(base.ID+”.v2”)
    override x.ShowID() = System.Console.Write(base.ID+”.v3”)
let laser1 = Laser()
let laser2 = SpeedLaser()
laser2.ShowID()
```

What does this output?

```
type Laser() =
    member x.ID = “Galaxy235”
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
    override x.ShowID() = System.Console.Write(base.ID+”.v2”)
    override x.ShowID() = System.Console.Write(base.ID+”.v3”)
let laser1 = Laser()
let laser2 = SpeedLaser()
laser2.ShowID()
```

Line 5*: More than one override implements 'ShowID : unit -> unit'*

Overriding in inheritance

- A *Base* class member can be overridden **only once** in **the same** *Derived* class

```
type Laser() =
    member x.ID = “Galaxy235”
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
    override x.ShowID() = System.Console.Write(base.ID+”.v2”)
type OtherLaser() =
    inherit SpeedLaser()
    override x.ShowID() = System.Console.Write(base.ID+”.v3”)
let laser2 = SpeedLaser()
laser2.ShowID()
let laser3 = OtherLaser()
laser3.ShowID()
```

What does this output?

```
type Laser() =
    member x.ID = “Galaxy235”
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
    override x.ShowID() = System.Console.Write(base.ID+”.v2”)
type OtherLaser() =
    inherit SpeedLaser()
    override x.ShowID() = System.Console.Write(base.ID+”.v3”)
let laser2 = SpeedLaser()
laser2.ShowID()
let laser3 = OtherLaser()
laser3.ShowID()
```

LASER

|

SPEEDLASER

|

OTHERLASER

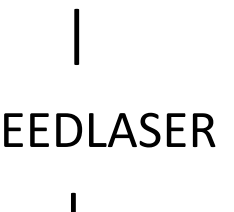
What does this output?

```
type Laser() =
    member x.ID = “Galaxy235”
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
    override x.ShowID() = System.Console.Write(base.ID+”.v2”)
type OtherLaser() =
    inherit SpeedLaser()
    override x.ShowID() = System.Console.Write(base.ID+”.v3”)
let laser2 = SpeedLaser()
laser2.ShowID()
let laser3 = OtherLaser()
laser3.ShowID()
```

Galaxy235.v2

Galaxy235.v3

```
type Laser() =
    member x.ID = “Galaxy235”
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
    override x.ShowID() = System.Console.Write(base.ID+”.v2”)
type OtherLaser() =
    inherit SpeedLaser()
    override x.ShowID() = System.Console.Write(base.ID+”.v3”)
let laser2 = SpeedLaser()
laser2.ShowID()
let laser3 = OtherLaser()
laser3.ShowID()
```



Galaxy235.v2

Why not Galaxy235.v2.v3?

Overriding in inheritance

- A *Base* class member can be overridden **only once** in **the same** *Derived* class
- A *Base* class member can be overridden **repeatedly** in **different** *Derived* classes in **different** ways

```
type Laser() =
    member x.ID = “Galaxy235”
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
    override x.ShowID() = System.Console.Write(base.ID+”.v2”)
type OtherLaser() =
    inherit Laser()
    override x.ShowID() = System.Console.Write(base.ID+”XX”)
let laser2 = SpeedLaser()
laser2.ShowID()
let laser3 = OtherLaser()
laser3.ShowID()
```

```
type Laser() =
    member x.ID = “Galaxy235”
    abstract member ShowID : unit -> unit
    default x.ShowID() = System.Console.Write(x.ID)
type SpeedLaser() =
    inherit Laser()
    override x.ShowID() = System.Console.Write(base.ID+”.v2”)
type OtherLaser() =
    inherit Laser()
    override x.ShowID() = System.Console.Write(base.ID+”XX”)
let laser2 = SpeedLaser()
laser2.ShowID()
let laser3 = OtherLaser()
laser3.ShowID()
```

Galaxy235.v2

Galaxy235XX

Overriding in inheritance

- A *Base* class member can be overridden **only once** in **the same** *Derived* class
- A *Base* class member can be overridden **repeatedly** in **different** *Derived* classes in **different** ways
- When a *Base* class member is overridden in a *Derived* class, **only** the overridden version can be used in the *Derived* class

Overriding in inheritance

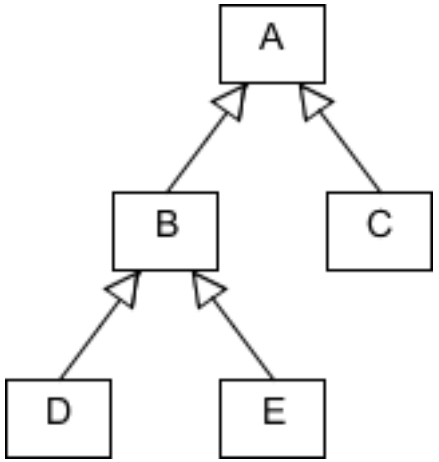
- A *Base* class member can be overridden **only once** in **the same** *Derived* class
- A *Base* class member can be overridden **repeatedly** in **different** *Derived* classes in **different** ways
- When a *Base* class member is overridden in a *Derived* class, **only** the overridden version can be used in the *Derived* class
- Even if a *Base* class member has been overridden, **only** the *Base* version can be used in the *Base* class

Overriding in inheritance

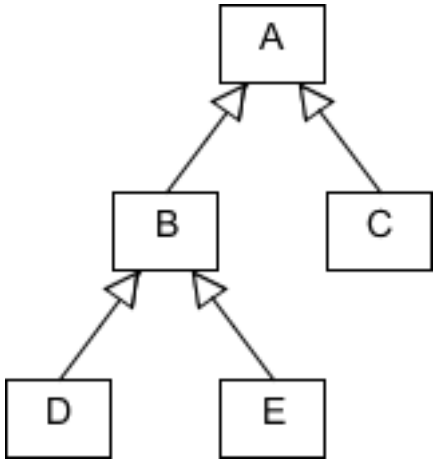
- A *Base* class member can be overridden **only once** in **the same** *Derived* class
- A *Base* class member can be overridden **repeatedly** in **different** *Derived* classes in **different** ways
- When a *Base* class member is overridden in a *Derived* class, **only** the overridden version can be used in the *Derived* class
- Even if a *Base* class member has been overridden, **only** the *Base* version can be used in the *Base* class
- Cannot override constructors

Inheritance creates class hierarchies

Every .NET class (incl. primitive data types) participates in inheritance



Inheritance creates class hierarchies



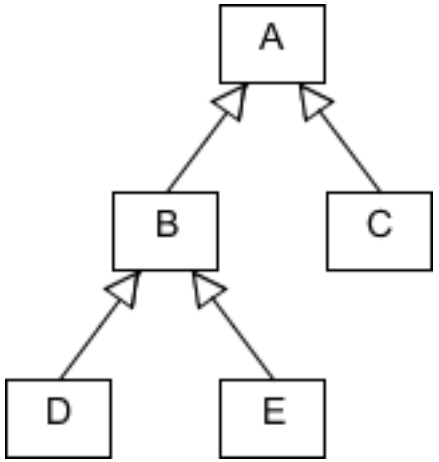
Every .NET class (incl. primitive data types) participates in inheritance

Classes close to the top tend to be general

Classes close to the bottom tend to be specialised

The further up, the more general the classes

Inheritance creates class hierarchies



Every .NET class (incl. primitive data types) participates in inheritance

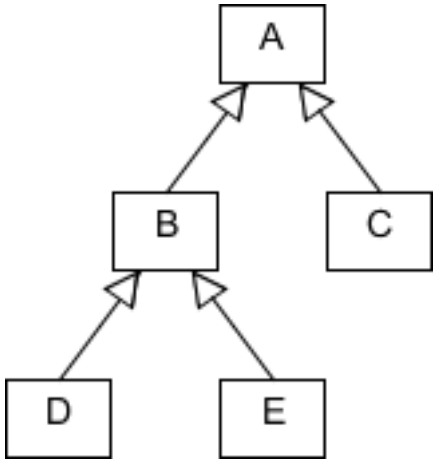
Classes close to the top tend to be general

Classes close to the bottom tend to be specialised

The further up, the more general the classes

Abstract classes (typically top of hierarchy)

Inheritance creates class hierarchies



Every .NET class (incl. primitive data types) participates in inheritance

Classes close to the top tend to be general

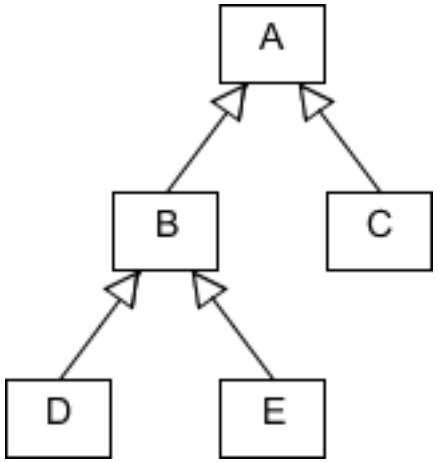
Classes close to the bottom tend to be specialised

The further up, the more general the classes

Abstract classes (typically top of hierarchy):

- *Cannot be instantiated*

Inheritance creates class hierarchies



Every .NET class (incl. primitive data types) participates in inheritance

Classes close to the top tend to be general

Classes close to the bottom tend to be specialised

The further up, the more general the classes

Abstract classes (typically top of hierarchy):

- *Cannot be instantiated directly*
- *Accessible only through derived classes*
- *Contain members without an implementation*

```
[<AbstractClass>]
type Laser() =
    abstract member ID : string
    abstract member ShowID : unit -> unit
```

```
[<AbstractClass>]                                   → ABSTRACT CLASS
type Laser() =
    abstract member ID : string
    abstract member ShowID : unit -> unit
```

```
[<AbstractClass>]                                  → ABSTRACT CLASS
type Laser() =
    abstract member ID : string                    → DEFINITION
    abstract member ShowID : unit -> unit          → DEFINITION
```

```
[<AbstractClass>]                          → ABSTRACT CLASS
type Laser() =
    abstract member ID : string            → DEFINITION
    abstract member ShowID : unit -> unit  → DEFINITION
type SpeedLaser() =
    inherit Laser()
    override x.ID = “Galaxy”
    override x.ShowID() = System.Console.Write(x.ID)
```

```
[<AbstractClass>]                                  → ABSTRACT CLASS
type Laser() =
    abstract member ID : string                    → DEFINITION
    abstract member ShowID : unit -> unit          → DEFINITION
type SpeedLaser() =
    inherit Laser()
    override x.ID = “Galaxy”                                  → IMPL
    override x.ShowID() = System.Console.Write(x.ID)          → IMPL
```

```
[<AbstractClass>]                                  → ABSTRACT CLASS
type Laser() =
    abstract member ID : string                    → DEFINITION
    abstract member ShowID : unit -> unit          → DEFINITION
type SpeedLaser() =
    inherit Laser()
    override x.ID = “Galaxy”                                  → IMPL
    override x.ShowID() = System.Console.Write(x.ID)          → IMPL
```

```
let laser2 = new SpeedLaser()
laser2.ShowID()
```

```
[<AbstractClass>]                                  → ABSTRACT CLASS
type Laser() =
    abstract member ID : string                    → DEFINITION
    abstract member ShowID : unit -> unit          → DEFINITION
type SpeedLaser() =
    inherit Laser()
    override x.ID = “Galaxy”                                  → IMPL
    override x.ShowID() = System.Console.Write(x.ID)          → IMPL
```

```
let laser2 = new SpeedLaser()
laser2.ShowID()                                    Galaxy
```

```
[<AbstractClass>]                                  → ABSTRACT CLASS
type Laser() =
    abstract member ID : string                    → DEFINITION
    abstract member ShowID : unit -> unit          → DEFINITION
type SpeedLaser() =
    inherit Laser()
    override x.ID = “Galaxy”                                  → IMPL
    override x.ShowID() = System.Console.Write(x.ID)          → IMPL

 let laser1 = new Laser()
laser1.ShowID()                                    output?
let laser2 = new SpeedLaser()
laser2.ShowID()                                    Galaxy
```

```
[<AbstractClass>]                                  → ABSTRACT CLASS
type Laser() =
    abstract member ID : string                    → DEFINITION
    abstract member ShowID : unit -> unit          → DEFINITION
type SpeedLaser() =
    inherit Laser()
    override x.ID = "Galaxy"                                  → IMPL
    override x.ShowID() = System.Console.Write(x.ID)          → IMPL

 let laser1 = new Laser()
laser1.ShowID()                                    Does not run
let laser2 = new SpeedLaser()
laser2.ShowID()                                    Galaxy
```

"Instances of this type cannot be created since it has been marked abstract"

```
[<AbstractClass>]                                  → ABSTRACT CLASS
type Laser() =
    abstract member ID : string                    → DEFINITION
    abstract member ShowID : unit -> unit          → DEFINITION
type SpeedLaser() =
    inherit Laser()
    override x.ID = “Galaxy”                                  → IMPL
    override x.ShowID() = System.Console.Write(x.ID)          → IMPL
```

Abstract class:

- **Cannot be instantiated**
- **Accessed only from Derived**
- **Contains unimplemented members**

```
[<AbstractClass>]                                  → ABSTRACT CLASS
type Laser() =
    abstract member ID : string                    → DEFINITION
    abstract member ShowID : unit -> unit          → DEFINITION
type SpeedLaser() =
    inherit Laser()
    override x.ID = "Galaxy"                                  → IMPL
    override x.ShowID() = System.Console.Write(x.ID)          → IMPL
                                                       ↑
```

Why not base.ID?

Abstract class:

- **Cannot be instantiated**
- **Accessed only from Derived**
- **Contains unimplemented members**

```
[<AbstractClass>]                                  → ABSTRACT CLASS
type Laser() =
    abstract member ID : string                    → DEFINITION
    abstract member ShowID : unit -> unit          → DEFINITION
type SpeedLaser() =
    inherit Laser()
    override x.ID = “Galaxy”                                  → IMPL
    override x.ShowID() = System.Console.Write(x.ID)          → IMPL
type OtherLaser() =
    inherit Laser()
    default x.ID = “Galaxy”                                   → IMPL
    default x.ShowID() = System.Console.Write(x.ID)           → IMPL
```

```
[<AbstractClass>]                                  → ABSTRACT CLASS
type Laser() =
    abstract member ID : string                    → DEFINITION
    abstract member ShowID : unit -> unit          → DEFINITION
type SpeedLaser() =
    inherit Laser()
    override x.ID = “Galaxy”                                  → IMPL
    override x.ShowID() = System.Console.Write(x.ID)          → IMPL
type OtherLaser() =
    inherit Laser()
    default x.ID = “Galaxy”                                   → IMPL
    default x.ShowID() = System.Console.Write(x.ID)           → IMPL
```

```
[<AbstractClass>]                                  → ABSTRACT CLASS
type Laser() =
    abstract member ID : string                    → DEFINITION
    abstract member ShowID : unit -> unit          → DEFINITION
type SpeedLaser() =
    inherit Laser()
    override x.ID = "Galaxy"                                  → IMPL
    override x.ShowID() = System.Console.Write(x.ID)          → IMPL
type OtherLaser() =
    inherit Laser()
    default x.ID = "Galaxy"                                   → IMPL
    default x.ShowID() = System.Console.Write(x.ID)           → IMPL
```

BOTH ARE VALID: When inheriting from *abstract* base class, *override* and *default* can be used interchangeably

```
[<AbstractClass>]
type Laser() =
    abstract member ID : string
    abstract member ShowID : unit -> unit
type SpeedLaser() =
    inherit Laser()
    override x.ID = "Galaxy"
    override x.ShowID() = System.Console.Write(x.ID)
type OtherLaser() =
    inherit Laser()
    default x.ID = "Galaxy"
    default x.ShowID() = System.Console.Write(x.ID)
```

Convention:

- Use *override* in derived class
- Use *default* in base class

BOTH ARE VALID: When inheriting from *abstract* base class, *override* and *default* can be used interchangeably

AN ABSTRACT CLASS:

```
[<AbstractClass>]
type Laser() =
    abstract member ID : string
    abstract member ShowID : unit -> unit
```

AN ABSTRACT CLASS:

```
[<AbstractClass>]
type Laser() =
    abstract member ID : string                    -> DEF
    abstract member ShowID : unit -> unit          -> DEF
```

AN ABSTRACT CLASS:

```
[<AbstractClass>]
type Laser() =
    abstract member ID : string                    -> DEF
    abstract member ShowID : unit -> unit          -> DEF
```

```
type Laser() =
    member x.ID = “Galaxy”                         -> DEF & IMPL
    member x.ShowID() = System.Console.Write(x.ID) -> DEF & IMPL
```

AN ABSTRACT CLASS:

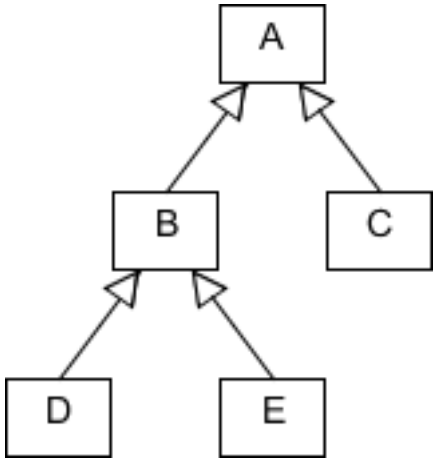
```
[<AbstractClass>]
type Laser() =
    abstract member ID : string                    -> DEF
    abstract member ShowID : unit -> unit          -> DEF
```

A CONCRETE CLASS:

```
type Laser() =
    member x.ID = “Galaxy”                         -> DEF & IMPL
    member x.ShowID() = System.Console.Write(x.ID) -> DEF & IMPL
```

AN ABSTRACT CLASS:

```
[<AbstractClass>]
type Laser() =
    abstract member ID : string                    -> DEF
    abstract member ShowID : unit -> unit          -> DEF
```



A CONCRETE CLASS:

```
type Laser() =
    member x.ID = “Galaxy”                         -> DEF & IMPL
    member x.ShowID() = System.Console.Write(x.ID) -> DEF & IMPL
```