Abstract

1. The methods in C# can be divided to two parts, which are declaration and content. As for abstract method, it is a special method that does not contain implementation.

2. The main purpose of abstract method is to declare method in advance, but it hands implementation over to the inherited class.

3. By using abstract method, we can enable derived classes to have common actions; however, the actual approach is customized by each class.

4. Abstract method must be placed in abstract class. Abstract class is a class that cannot be instantiated(does not invoke by new). The reason lies in the fact that abstract method does not have defined contents; therefore, it is not able to start any action if instantiated.

Base

1. Through the keyword of “this”, we can access properties and methods of objects; it is often used when the parameter name and the property name of method are repeated. The base keyword has the same principle, but what base access is bass class.

2. **The first** circumstance when base is used: When there are properties with the same name in derived class, you can use base to access if you’d like to access base class properties but not derived class properties.

3. The **second** circumstance when base is used: When derived class overrides a method of base class, you can use base if you’d like to access the original method(the original version in base class).

4. **The third** circumstance when base is used: You can add base on your constructor to call when you’d like to call the base class constructor.

Interface

1. Interface and class are similar, except that it can only contain methods that are not implemented.

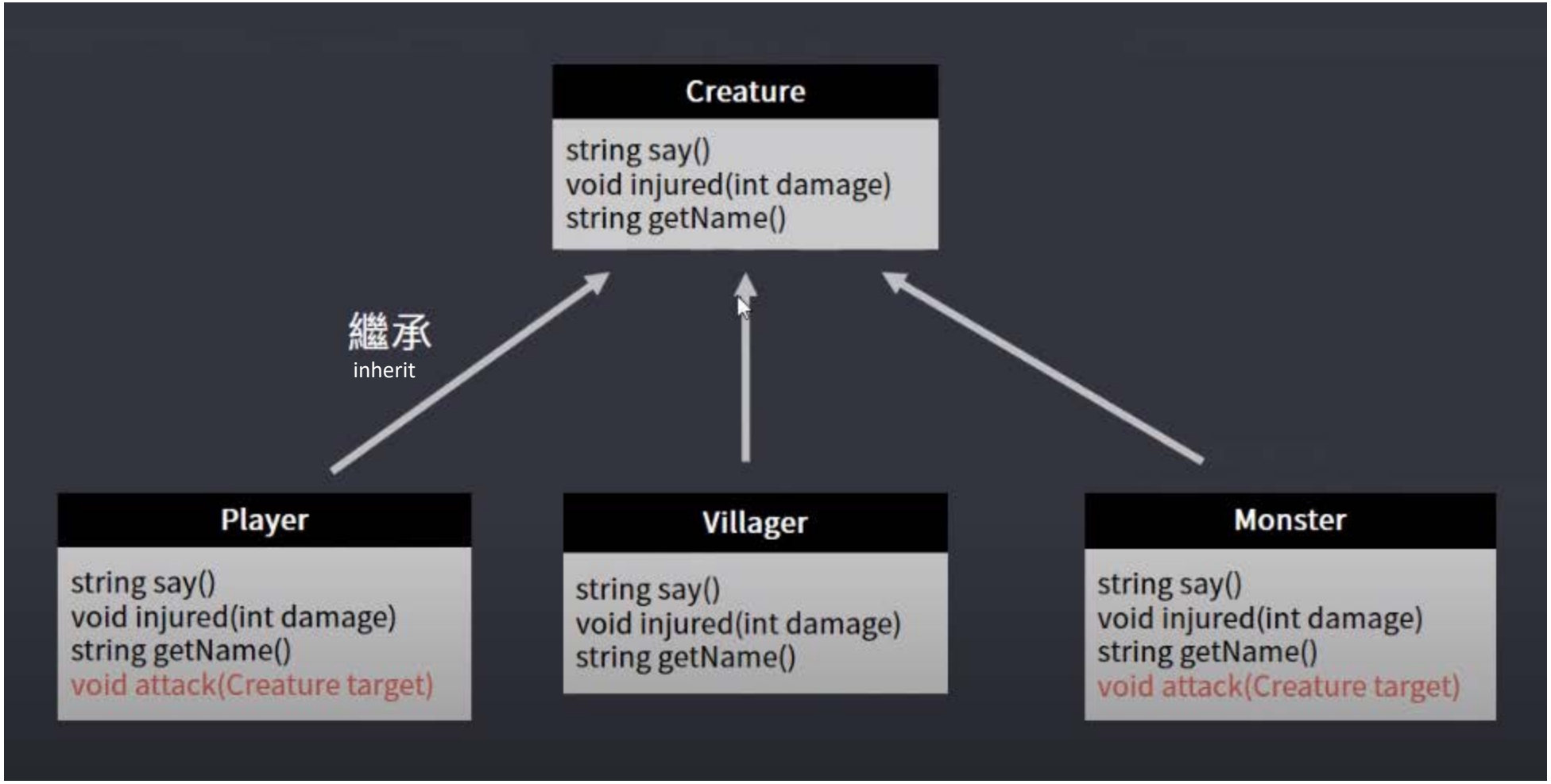
2. Interface has similar effects as contracts; it stipulates class to fulfill required methods.

3. Although class can only inherit one base class, it can implement multiple interfaces.

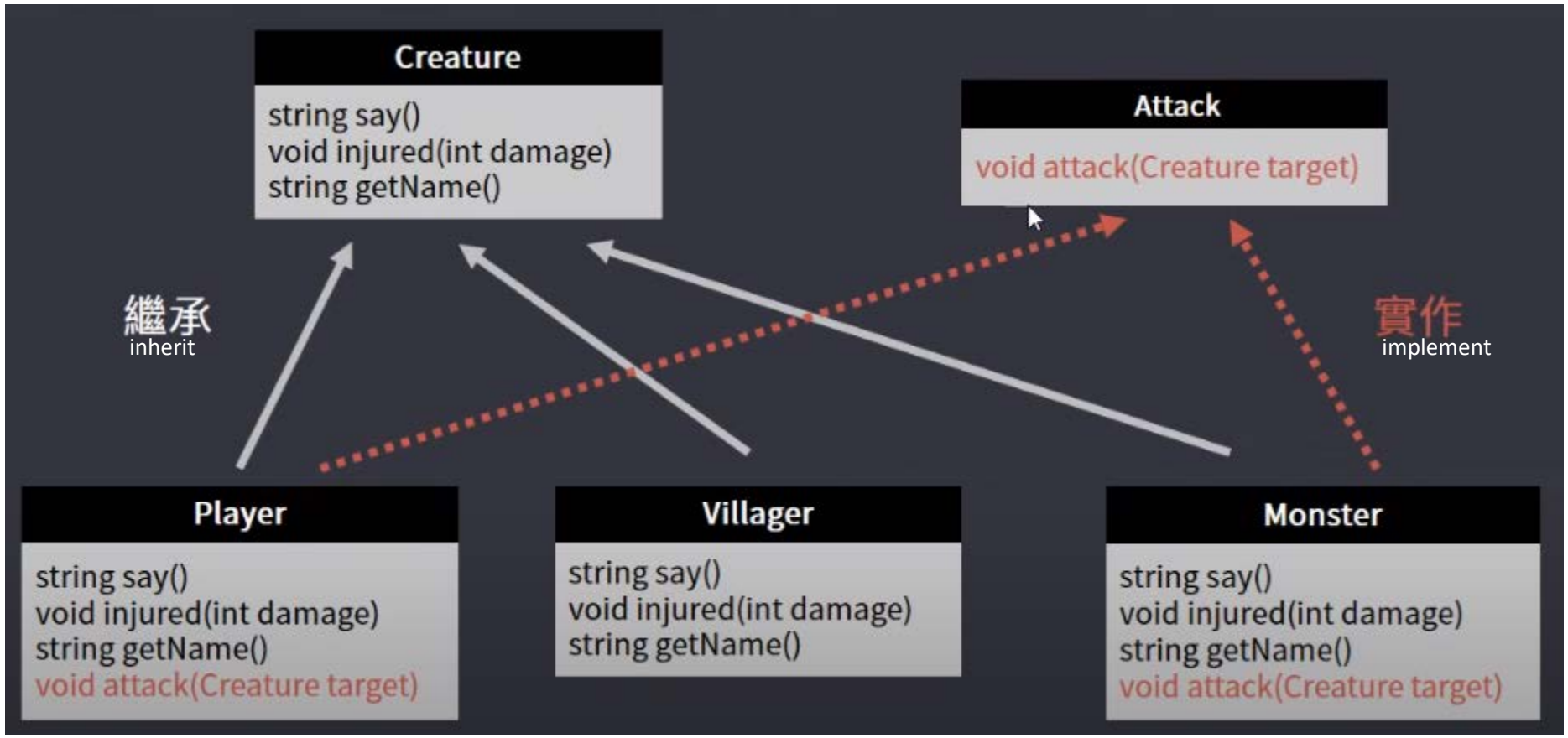
4. When implementing methods of interfaces, they had to be public but not static.

5. Name interface by starting from the capital letter “I” to be separated from inheritance.

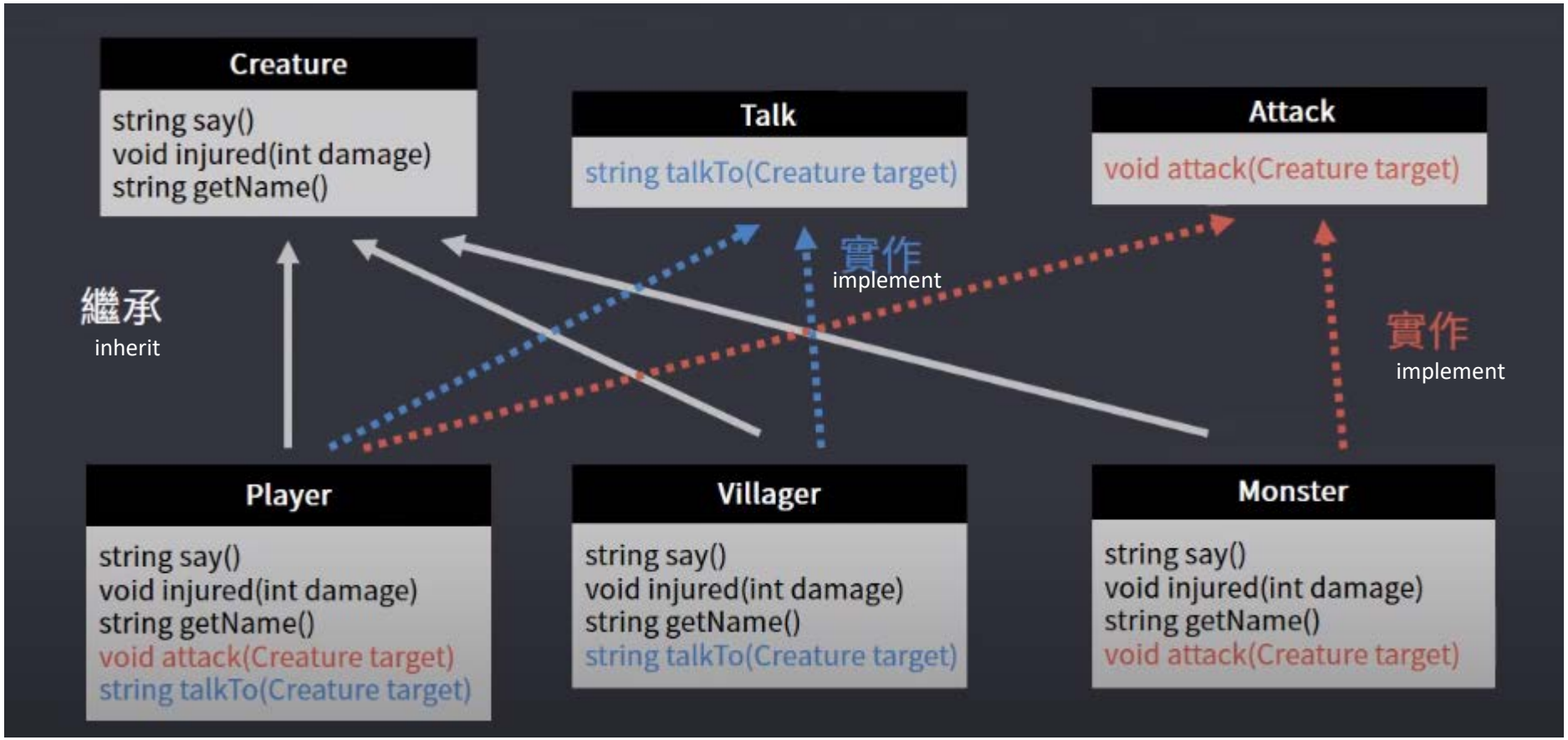
Interface_1


Creature
string say()
void injured(int damage)
string getName()
繼承
inherit
Player
string say()
void injured(int damage)
string getName()
void attack(Creature target)
Villager
string say()
void injured(int damage)
string getName()
Monster
string say()
void injured(int damage)
string getName()
void attack(Creature target)

Interface_2


Creature
string say()
void injured(int damage)
string getName()
Attack
void attack(Creature target)
繼承
inherit
實作
implement
Player
string say()
void injured(int damage)
string getName()
void attack(Creature target)
Villager
string say()
void injured(int damage)
string getName()
Monster
string say()
void injured(int damage)
string getName()
void attack(Creature target)

Interface_3


Creature
string say()
void injured(int damage)
string getName()
Talk
string talkTo(Creature target)
Attack
void attack(Creature target)
繼承
inherit
實作
implement
實作
implement
Player
string say()
void injured(int damage)
string getName()
void attack(Creature target)
string talkTo(Creature target)
Villager
string say()
void injured(int damage)
string getName()
string talkTo(Creature target)
Monster
string say()
void injured(int damage)
string getName()
void attack(Creature target)