

Polymorphism Notes

From the Greek *(loosely translated)*:

Poly - Many

Morphism - Having a state or form

}

Many forms

In object-oriented programming, polymorphism is the idea that something can be assigned a different meaning or usage based on its context. This specifically allows variables and objects to take on more than one form.

`object.method()` - Normally the **datatype/class of the object** is used to determine which method to run.

The compiler decides which method to run: **compile-time binding**

```
AmericanPlayingCard aUSACard      = new AmericanPlayingCard(1, "HEARTS");
ItalianPlayingCard  anItalianCard1 = new ItalianPlayingCard(13, "SWORDS");
SwissPlayingCard    aSwissCard1     = new SwissPlayingCard(13, "ROSES");

aUSACard.showCard();           // runs the AmericanPlayingCard showCard() method
anItalianCard1.showCard();     // runs the ItalianPlayingCard showCard() method
aSwissCard1.showCard();        // runs the SwissPlayingCard showCard() method
```

`object.method()` - With Polymorphism the **datatype/class of the object stored in the variable** is used to determine which method to run.

The method to run is determined at run-time and **overrides the compiler decision** made at compile time.
run-time binding

All classes, **especially the super-class** must have an implementation of the polymorphic method, even if it does nothing. The **virtual** keyword is specified on the base-class method to indicate it is Polymorphic.

Polymorphism requires inheritance (super classes and sub classes)

Inheritance DOES NOT require Polymorphism.

To implement Polymorphism:

1. **Define a** variable of the **super-class** and **store a sub-class** object in it.
2. **Use the super-class variable** to invoke the common/overriden methods

```
PlayingCard aCard;      // define a Super class object

// Define some sub-class objects

AmericanPlayingCard aUSACard      = new AmericanPlayingCard(1, "HEARTS");
ItalianPlayingCard  anItalianCard1 = new ItalianPlayingCard(13, "SWORDS");
SwissPlayingCard    aSwissCard1     = new SwissPlayingCard(13, "ROSES");

aCard = aUSACard;          // assign a sub-class object to a super-class variable
aCard.showCard();          // Use the super-class variable to invoke the method
                           //      the AmericanPlayingCard showCard() is run

aCard = anItalianCard1;    // assign a sub-class object to a super-class variable
aCard.showCard();          // Use the super-class variable to invoke the method
                           //      the ItalianPlayingCard showCard() is run

aCard = aSwissCard1        // assign a sub-class object to a super-class variable
aCard.showCard();          // Use the super-class variable to invoke the method
                           //      the SwissPlayingCard showCard() is run
```

Without polymorphism, `aCard.showCard()` would try to run the `PlayingCard` class `showCard()` method.

Polymorphism allows the user of an inheritance hierarchy to code without regard to which class in the inheritance hierarchy an object is a instance of.
i.e. **Easier for application programmer to code** and use the objects.

