Polymorphism

In programming languages and type theory, is the provision of a single interface to entities of different types. A polymorphic type is a type whose operations can also be applied to values of some other type, or types. There are several fundamentally different kinds of polymorphism:

Ad hoc polymorphism

If a function denotes different and potentially heterogeneous implementations depending on a limited range of individually specified types and combinations, it is called ad hoc polymorphism. Ad hoc polymorphism is supported in many languages using function overloading.

Parametric polymorphism

If the code is written without mention of any specific type and thus can be used transparently with any number of new types, it is called parametric polymorphism. In the object-oriented programming community, this is often known as generics or generic programming. In the functional programming community, this is often simply called polymorphism.

Subtyping polymorphism

Subtyping (or inclusion polymorphism) is a concept wherein a name may denote instances of many different classes as long as they are related by some common superclass. In object-oriented programming, this is often referred to simply as polymorphism.

Pillar of object-oriented programming

Polymorphism is often referred to as the third pillar of object-oriented programming, after encapsulation and inheritance. Polymorphism is a Greek word that means "many-shaped" and it has two distinct aspects:

At run time, objects of a derived class may be treated as objects of a base class in places such as method parameters and collections or arrays. When this occurs, the object's declared type is no longer identical to its run-time type.

Base classes may define and implement virtual methods, and derived classes can override them, which means they provide their own definition and implementation. At run-time, when client code calls the method, the CLR looks up the run-time type of the object, and invokes that override of the virtual method. Thus in your source code you can call a method on a base class, and cause a derived class's version of the method to be executed.

Examples

- A variable with a given name may be allowed to have different forms and the program can determine which form of the variable to use at the time of execution. For example, a variable named USERID may be capable of being either an integer (whole number) or a string of characters (perhaps because the programmer wants to allow a user to enter a user ID as either an employee number - an integer - or with a name - a string of characters). By giving the program a way to distinguish which form is being handled in each case, either kind can be recognized and handled.

- A named function can also vary depending on the parameters it is given. For example, if given a variable that is an integer, the function chosen would be to seek a match against a list of employee numbers; if the variable were a string, it would seek a match against a list of names. In either case, both functions would be known in the program by the same name. This type of polymorphism is sometimes known as overloading.

- In C#, for example, the operator known as the plus sign (+) - which is effectively a simple named function - can be assigned to operate on two objects such that it adds them together (perhaps the most common form of the + operation). In another context, the + sign could mean an operation to concatenate the two objects or strings of letters on either side of the + sign.

- A given operator can also be given yet another meaning when combined with another operator. For example, in the C# language, a "++" following a variable can mean "increment this value by 1". The meaning of a particular operator is defined as part of a class definition. Since the programmer can create classes, the programmer can also define how operators work for this class of objects; in effect, the programmer can redefine the computing language.

- Polymorphism can mean, a data type of "any," such that when specified for a list, a list containing any data types can be processed by a function. (For example, if a function simply determines the length of a list, it doesn't matter what data types are in the list.