The Concept of Polymorphism in Object Oriented Programing

With illustration in Python 2.7

Presented by

Michael S. James

Tuesday, November 14, 2017 @ 12:00 PM
Revision 2

DEFINITION:

“Generally, the ability to appear in many forms. In object-oriented programming, polymorphism refers to a programming language's ability to process objects differently depending on their data type or class.”

Source: Vangie Bael @ https://www.webopedia.com/TERM/P/polymorphism.html

DEFINITION (cont.):

“More specifically, it is the ability to redefine methods for derived classes.”

Source: Vangie Bael @ https://www.webopedia.com/TERM/P/polymorphism.html

DEFINITION (cont.):

“For example, given a base class *Animal*, polymorphism enables the programmer to define different *animal behavior* methods for any number of derived classes, such as *dogs* and *cats*.”

Source: Vangie Bael @ https://www.webopedia.com/TERM/P/polymorphism.html

DEFINITION (cont.):

No matter what *type an animal object is*, applying the *animal behavior* to it will return the correct results. Polymorphism is considered to be a requirement of any true object-oriented programming language (OOPL).

Source: Vangie Bael @ https://www.webopedia.com/TERM/P/polymorphism.html

CODE EXAMPLE: create base class *Animal*

```
# Create Animal() superclass object.
class Animal(object):

    # Class Object Attributes
    species = 'mammal'
    bloodtype = 'warm blooded'

    def __init__(self):
        pass

    def whoAmi(self):
        print "Animal"

    def eat(self):
        print "Eating"
```

Source: Michael S. James
@ https://github.com/msjames19702/oop-polymorphism

CODE EXAMPLE: create derived class *Dog*

```
# Create Dog() subclass object and functions. Make use of Inheritance.

class Dog(Animal):
    def __init__(self):
        Animal.__init__(self)
        pass

    def whoAmi(self):
        print "I'm a Dog"

    def sound(self):
        print "I go Woof!"

    def whatIdo(self):
        print "I play fetch the ball!"
```

Source: Michael S. James
@ https://github.com/msjames19702/oop-polymorphism

CODE EXAMPLE: create derived class *Cat*

```
# Create Cat() subclass object and functions. Make use of Inheritance.

class Cat(Animal):
    def __init__(self):
        Animal.__init__(self)
        pass

    def whoAmi(self):
        print "I'm a Cat"

    def sound(self):
        print "I go Meow!"

    def whatIdo(self):
        print "I play with yarn!"
```

Source: Michael S. James
@ https://github.com/msjames19702/oop-polymorphism

CODE EXAMPLE: create object instances and polymorphic functions

```
# Create polymorphic object instances of Dog() & Cat()

dogObject = Dog()
catObject = Cat()
wolfObject = Dog()
tigerOjbject = Cat()

# Functional polymorphisms

def whatAmi(animaltype):
    animaltype.whoAmi()

def soundImake(animalsound):
    animalsound.sound()

def iDo(animaldoes):
    animaldoes.whatIdo()
```

Source: Michael S. James
@ https://github.com/msjames19702/oop-polymorphism

CODE EXAMPLE: *dogObject* and results

```
# Calling the above functions with objects as arguments to illustrate polymorphism

print "Dog says ..."
whatAmi(dogObject)
soundImake(dogObject)
iDo(dogObject)
```

```
Python 2.7.13 |Anaconda 4.3.0 (64-bit)| (default, Dec 20 2016, 23:09:15)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux2
>>> print "Dog says ..."
Dog says ...
>>> whatAmi(dogObject)
I'm a Dog
>>> soundImake(dogObject)
I go Woof!
>>> iDo(dogObject)
I play fetch the ball!
```

Source: Michael S. James
@ https://github.com/msjames19702/oop-polymorphism

CODE EXAMPLE: *catObject* and results

```
# Calling the above functions with objects as arguments to illustrate polymorphism

print "Cat says ..."
whatAmi(catObject)
soundImake(catObject)
iDo(catObject)
```

```
Python 2.7.13 |Anaconda 4.3.0 (64-bit)| (default, Dec 20 2016, 23:09:15)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux2
>>> print "Cat says ..."
Cat says ...
>>> whatAmi(catObject)
I'm a Cat
>>> soundImake(catObject)
I go Meow!
>>> iDo(catObject)
I play with yarn!
```

Source: Michael S. James
@ https://github.com/msjames19702/oop-polymorphism

CODE EXAMPLE: *wolfObject* and results

```
# Calling the above functions with objects as arguments to illustrate polymorphism

print "Wolf says ..."
whatAmi(wolfObject)
soundImake(wolfObject)
iDo(wolfObject)
```

```
Python 2.7.13 |Anaconda 4.3.0 (64-bit)| (default, Dec 20 2016, 23:09:15)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux2
>>> print "Wolf says ..."
Wolf says ...
>>> whatAmi(wolfObject)
I'm a Dog
>>> soundImake(wolfObject)
I go Woof!
>>> iDo(wolfObject)
I play fetch the ball!
```

Source: Michael S. James
@ https://github.com/msjames19702/oop-polymorphism

CODE EXAMPLE: *tigerObject* and results

```
# Calling the above functions with objects as arguments to illustrate polymorphism

print "Tiger says ..."
whatAmi(tigerObject)
soundImake(tigerObject)
iDo(tigerObject)
```

```
Python 2.7.13 |Anaconda 4.3.0 (64-bit)| (default, Dec 20 2016, 23:09:15)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux2
>>> print "Tiger says ..."
Tiger says ...
>>> whatAmi(tigerObject)
I'm a Cat
>>> soundImake(tigerObject)
I go Meow!
>>> iDo(tigerObject)
I play with yarn!
```

Source: Michael S. James
@ https://github.com/msjames19702/oop-polymorphism

Thank you !!!

Presented by

Michael S. James

Tuesday, November 14, 2017 @ 12:00 PM