Object Oriented Methodology

Week – *5*, Lecture – *1*

Polymorphism

SAURABH SRIVASTAVA

VISITING FACULTY

IIIT LUCKNOW

Definition of Polymorphism

Polymorphism (biology)

From Wikipedia, the free encyclopedia

For other uses, see Polymorphism.

In biology, **polymorphism**[1] is the occurrence of two or more clearly different **morphs** or **forms**, also referred to as alternative *phenotypes*, in the population of a species. To be classified as such, morphs must occupy the same habitat at the same time and belong to a panmictic population (one with random mating).[2]

Put simply, polymorphism is when there are two or more possibilities of a trait on a gene. For example, there is more than one possible trait in terms of a jaguar's skin colouring; they can be light morph or dark morph. Due to having more than one possible variation for this gene, it is termed 'polymorphism'. However, if the jaguar has only one possible trait for that gene, it would be termed "monomorphic". For example, if there was only one possible skin colour that a jaguar could have, it would be termed monomorphic.

The term polyphenism can be used to clarify that the different forms arise from the same genotype. Genetic polymorphism is a term used somewhat differently by geneticists and molecular biologists to describe certain mutations in the genotype, such as single nucleotide polymorphisms that may not always correspond to a phenotype, but always corresponds to a branch in the genetic tree. See below.

Polymorphism is common in nature; it is related to biodiversity, genetic variation, and adaptation. Polymorphism usually functions to retain variety of form in a population living in a varied environment.[3]:126 The most common example is sexual dimorphism, which occurs in many organisms. Other examples are mimetic forms of butterflies (see mimicry), and human hemoglobin and blood types.

According to the theory of evolution, polymorphism results from evolutionary processes, as does any aspect of a species. It is heritable and is modified by natural selection. In polyphenism, an individual's genetic makeup allows for different morphs, and the switch mechanism that determines which morph is shown is environmental. In genetic polymorphism, the genetic makeup determines the morph.

The term polymorphism also refers to the occurrence of structurally and functionally more than two different types of individuals, called zooids, within the same organism. It is a characteristic feature of cnidarians.[2] For example, *Obelia* has feeding individuals, the gastrozooids; the individuals capable of asexual reproduction only, the gonozooids, blastostyles; and free-living or sexually reproducing individuals, the medusaearyana.

Light-morph jaguar

Dark-morph or melanistic jaguar (about 6% of the South American population)

Part of a series on

Evolutionary biology

Definition of Polymorphism

Polymorphism (biology)

From Wikipedia, the free encyclopedia

For other uses, see Polymorphism.

In biology, **polymorphism**[1] is the occurrence of two or more clearly different **morphs** or **forms**, also referred to as alternative *phenotypes*, in the population of a species. To be classified as such, morphs must occupy the same habitat at the same time and belong to a panmictic population (one with random mating).[2]

Put simply, polymorphism is when there are two or more possibilities of a trait on a gene. For example, there is more than one possible trait in terms of a jaguar's skin colouring; they can be light morph or dark morph. Due to having more than one possible variation for this gene, it is termed 'polymorphism'. However, if the jaguar has only one possible trait for that gene, it would be termed "monomorphic". For example, if there was only one possible skin colour that a jaguar could have, it would be termed monomorphic.

The term polyphenism can be used to clarify that the different forms arise from the same genotype. Genetic polymorphism is a term used somewhat differently by geneticists and molecular biologists to describe certain mutations in the genotype, such as single nucleotide polymorphisms that may not always correspond to a phenotype, but always corresponds to a branch in the genetic tree. See below.

Polymorphism is common in nature; it is related to biodiversity, genetic variation, and adaptation. Polymorphism usually functions to retain variety of form in a population living in a varied environment.[3]:126 The most common example is sexual dimorphism, which occurs in many organisms. Other examples are mimetic forms of butterflies (see mimicry), and human hemoglobin and blood types.

According to the theory of evolution, polymorphism results from evolutionary processes, as does any aspect of a species. It is heritable and is modified by natural selection. In polyphenism, an individual's genetic makeup allows for different morphs, and the switch mechanism that determines which morph is shown is environmental. In genetic polymorphism, the genetic makeup determines the morph.

The term polymorphism also refers to the occurrence of structurally and functionally more than two different types of individuals, called zooids, within the same organism. It is a characteristic feature of cnidarians.[2] For example, *Obelia* has feeding individuals, the gastrozooids; the individuals capable of asexual reproduction only, the gonozooids, blastostyles; and free-living or sexually reproducing individuals, the medusaearyana.

Light-morph jaguar

Dark-morph or melanistic jaguar (about 6% of the South American population)

Part of a series on

Evolutionary biology

Definition of Polymorphism

Polymorphism (biology)

From Wikipedia, the free encyclopedia

For other uses, see Polymorphism.

In biology, **polymorphism**[1] is the occurrence of two or more clearly different **morphs** or **forms**, also referred to as alternative *phenotypes*, in the population of a species. To be classified as such, morphs must occupy the same habitat at the same time and belong to a panmictic population (one with random mating).[2]

Put simply, polymorphism is when there are two or more possibilities of a trait on a gene. For example, there is more than one possible trait in terms of a jaguar's skin colouring; they can be light morph or dark morph. Due to having more than one possible variation for this gene, it is termed 'polymorphism'. However, if the jaguar has only one possible trait for that gene, it would be termed "monomorphic". For example, if there was only one possible skin colour that a jaguar could have, it would be termed monomorphic.

The term polyphenism can be used to clarify that the different forms arise from the same genotype. Genetic polymorphism is a term used somewhat differently by geneticists and molecular biologists to describe certain mutations in the genotype, such as single nucleotide polymorphisms that may not always correspond to a phenotype, but always corresponds to a branch in the genetic tree. See below.

Polymorphism is common in nature; it is related to biodiversity, genetic variation, and adaptation. Polymorphism usually functions to retain variety of form in a population living in a varied environment.[3]:126 The most common example is sexual dimorphism, which occurs in many organisms. Other examples are mimetic forms of butterflies (see mimicry), and human hemoglobin and blood types.

According to the theory of evolution, polymorphism results from evolutionary processes, as does any aspect of a species. It is heritable and is modified by natural selection. In polyphenism, an individual's genetic makeup allows for different morphs, and the switch mechanism that determines which morph is shown is environmental. In genetic polymorphism, the genetic makeup determines the morph.

The term polymorphism also refers to the occurrence of structurally and functionally more than two different types of individuals, called zooids, within the same organism. It is a characteristic feature of cnidarians.[2] For example, *Obelia* has feeding individuals, the gastrozooids; the individuals capable of asexual reproduction only, the gonozooids, blastostyles; and free-living or sexually reproducing individuals, the medusaearyana.

Light-morph jaguar

Dark-morph or melanistic jaguar (about 6% of the South American population)

Part of a series on

Evolutionary biology

Pretty cool.. Isn't it?

Polymorphism in OOP

When discussed in the context of Inheritance, Polymorphism usually means the following

- Two pointers (object references) of the same type, behave differently when used in the same fashion
- The same method behaving differently, when used with different parameter list
- The same method behaving differently, when used with same parameter list

We will discuss all these variations, with the help of some C++ code

There's no specific term for the first variation, people usually just call it "polymorphism"

- Maybe I don't know the term, if you guys find it out, do tell me as well !!

The second type has a term – *method overloading*

- You can find this in almost all languages that support OOP

The third type also has a term – *method overriding*

- Most languages that support OOP have it, although you may have to follow some bureaucratic nonsense :-P

Revisiting the Pointer to Member operator

To take advantage of Polymorphism, you'll need to go back to the world of Pointers

In particular, you will use the Pointer to Member operator heavily

- Previously, we saw the pointer to member operator (`->`) with respect to structure variables
- They work in exactly the same fashion with objects too
- With objects, the operator can also be used to invoke methods on the object being pointed to

In fact, there is a *special pointer* often used in member functions, called the `this` pointer

- Using the `this` pointer, you can make your constructor parameter names more meaningful
- Check your homework !!

One aspect of pointers with inheritance, is that they can point to different types of objects

- This means that a pointer can point to any "compatible" object, even if the types are not an exact match
- Let us see some examples now of this "compatibility"

Polymorphic Pointers in C++

In C++, an object of a super type point to an object of a derived type

```
SuperType st; // An object of super type
DerivedType dt; // An object of derived type

SuperType* ptr1 = &st; // Works !!
DerivedType* ptr2 = &dt; // Works !!
SuperType* ptr3 = &dt; // Works as well…
```

It is not possible the other way around !!

```
DerivedType* ptr4 = &st; // Do this to get a free scolding from the compiler !!
```

Polymorphic Pointers in C++

In C++, an object of a super type point to an object of a derived type

```
SuperType st; // An object of super type
DerivedType dt; // An object of derived type

SuperType* ptr1 = &st; // Works !!
DerivedType* ptr2 = &dt; // Works !!
SuperType* ptr3 = &dt; // Works as well…
```

It is not possible the other way around !!

```
DerivedType* ptr4 = &st; // Do this to get a free scolding from the compiler !!
```

However, when you do so, only the members inherited from the super type are accessible

```
ptr3->methodInheritedFromSuperType(); // Works !!
ptr3->methodAddedInDerivedType(); // Here comes the scolding again …
```

Polymorphic Pointers in C++

In C++, an object of a super type point to an object of a derived type

```
SuperType st; // An object of super type
DerivedType dt; // An object of derived type

SuperType* ptr1 = &st; // Works !!
DerivedType* ptr2 = &dt; // Works !!
SuperType* ptr3 = &dt; // Works as well…
```

It is not possible the other way around !!

```
DerivedType* ptr4 = &st; // Do this to get a free scolding from the compiler !!
```

However, when you do so, only the members inherited from the super type are accessible

```
ptr3->methodInheritedFromSuperType(); // Works !!
ptr3->methodAddedInDerivedType(); // Here comes the scolding again …
```

The super type can be any class in the inheritance hierarchy of the derived class …

- … not just the immediate superclass

Usefulness of Polymorphism – Example

```
void put_toy_on_sale(Toy2* toy)
{
        cout<<"Putting "<<toy->get_name()<<" for sale(Id: "<<toy->get_id()<<"), ";
        cout<<"for the price of Rs "<<toy->get_price()<<endl;
}
```

Assume that we want to put a set of toys “on sale” at a shop

The only pieces of information required at this stage are the name, a unique id for the toy and its price

It doesn’t matter what features the toy has – we only need these details

This is where polymorphism can be really useful …

Usefulness of Polymorphism – Example

```
Toy2 *toy_ptr;
Toy2 blocks("Blocks", 399);
toy_ptr = &blocks;
put_toy_on_sale(toy_ptr);

BatteryOperatedToy2 piano("Kids' Piano", 399, "AA", 3);
toy_ptr = &piano;
put_toy_on_sale(toy_ptr);

FlyingToy frisby("Frisby", 299, 0);
toy_ptr = &frisby;
put_toy_on_sale(toy_ptr);

PlaneToy plane("Aeroplane", 999, "AA", 3, "Airbus A380");
toy_ptr = &plane;
put_toy_on_sale(toy_ptr);
```

This is how we use it to our advantage …

Usefulness of Polymorphism – Example

```
Toy2 *toy_ptr;
Toy2 blocks("Blocks", 399);
toy_ptr = &blocks;
put_toy_on_sale(toy_ptr);

BatteryOperatedToy2 piano("Kids' Piano", 399, "AA", 3);
toy_ptr = &piano;
put_toy_on_sale(toy_ptr);

FlyingToy frisby("Frisby", 299, 0);
toy_ptr = &frisby;
put_toy_on_sale(toy_ptr);

PlaneToy plane("Aeroplane", 999, "AA", 3, "Airbus A380");
toy_ptr = &plane;
put_toy_on_sale(toy_ptr);
```

This seems straightforward …

A pointer of type `Toy2`, pointing to an object of type `Toy2`

… which, is then passed to the `put_toy_on_sale()` function

Usefulness of Polymorphism – Example

```
Toy2 *toy_ptr;
Toy2 blocks("Blocks", 399);
toy_ptr = &blocks;
put_toy_on_sale(toy_ptr);

BatteryOperatedToy2 piano("Kids' Piano", 399, "AA", 3);
toy_ptr = &piano;
put_toy_on_sale(toy_ptr);

FlyingToy frisby("Frisby", 299, 0);
toy_ptr = &frisby;
put_toy_on_sale(toy_ptr);

PlaneToy plane("Aeroplane", 999, "AA", 3, "Airbus A380");
toy_ptr = &plane;
put_toy_on_sale(toy_ptr);
```

But how about this for polymorphism?

A pointer of type `Toy2`, pointing to an object of type `BatteryOperatedToy2`

… which, is then passed to the `put_toy_on_sale()` function in the same fashion as the object of type `Toy2`

Usefulness of Polymorphism – Example

```
Toy2 *toy_ptr;
Toy2 blocks("Blocks", 399);
toy_ptr = &blocks;
put_toy_on_sale(toy_ptr);

BatteryOperatedToy2 piano("Kids' Piano", 399, "AA", 3);
toy_ptr = &piano;
put_toy_on_sale(toy_ptr);

FlyingToy frisby("Frisby", 299, 0);
toy_ptr = &frisby;
put_toy_on_sale(toy_ptr);

PlaneToy plane("Aeroplane", 999, "AA", 3, "Airbus A380");
toy_ptr = &plane;
put_toy_on_sale(toy_ptr);
```

Similar stories here …

So basically, you don’t need to create separate functions for all these classes

If the properties or behaviour you need are available at a base class level, use a pointer to the base class, and use derived class objects seamlessly with it !!

Homework !!

A problem with constructors is that there is no way to differentiate between say a field called *x* and a parameter called *x*, e.g.

```
class C
{
    int x;
    public:
        C(int x)
        {
            // which x is actually "x" here?
        }
};
```

The `this` pointer can be helpful in this scenario

- Read and understand the example usage here: https://www.geeksforgeeks.org/this-pointer-in-c/
- It makes the constructor easier to read – since your parameter names could match their corresponding fields