Programming language paradigm

Procedural or structured programming
C, PASCAL, Fortran
In procedural language large program is divided into small procedure (Function).
For example
Calculator Software

```
add()
{
        //steps for addition
}

sub()
{
        //steps for subtraction
}

mul()
{
        //steps for multiplication
}
div()
{
        //steps for division
                        }
```

Functional programming
Scala, Julia
Lambda function,

Introduction to oops

Before the development of c each language is used for specific purpose. For eg COBOL for commercial applications and FORTRAN for Engineering and scientific applications. At this stage people started thinking that instead of learning and using so many languages, each for a different purpose why not use only one language, which can perform all possible applications. So they developed C. C is a structured language. The feature of a structure language is compartmentalization of code and data.
During the 1970s and 1980s, c becomes the most widely used programming language and it is still widely used today. Since c is a successful and useful language why c++ is invented. The answer is complexity.

BPCL->B-> C->C++

To manage complexity of the programs oops was introduced. The oops is based on three important concepts namely Encapsulation, Polymorphism, Inheritance.

All computers program consist of two elements, functions (methods, code) and variables (data).
A program can be conceptually organized around its function or around its data. That is some programs are written around "What is happening" and others are written around "Who is being affected". The first way is called the process-oriented model.
In processes oriented model the program is written as series of instructions i.e. function. In process oriented model function acting on variables. C uses process-oriented model.
To handle complexity object oriented programming was invented. OOP organizes a program around its variables and set of well-defined interfaces to those variables. An Object Oriented program can be characterized by variable controlling access to function.

OOPS:

Object Oriented programming structure is a programming methodology that helps organize complex programs through the use of inheritance, encapsulation and polymorphism.

Object oriented programming languages

C++, Java, Python, Kotlin, JavaScript, PHP,

What Is an Object:

Anything which we can see or touch is called object. Ex Dog, Fan, Bicycle.

The principal building blocks of object-oriented programs. Each object is a programming unit consisting of data (variables) and functionality (methods).

You can look around you now and see many examples of real-world objects: your dog, your desk, your television set, your bicycle.

These real-world objects share two characteristics: they all have *state* and they all have *behavior*. For example, dogs have state (name, color, hungry) and dogs have behavior (barking, fetching). Bicycles have state (current gear, current pedal cadence, two wheels, number of gears) and behavior (braking, accelerating, slowing down, changing gears)

Software objects are modeled after real world objects in that they, too, have state and behavior. A software object maintains its state in variables and implements its behavior with methods

Definition: An object is a software group of variables and related methods (functions)

What Is a Class:

In the real world, you often have many objects of the same kind. For example, your bicycle is just one of many bicycles in the world. Using object oriented terminology, we say that your bicycle object is a instance of the class of objects known as bicycles. Bicycles have some state (current gear, current cadence, two wheels) and behavior (change gears, brake) in common. However, each bicycle's state is independent of and can be different from other bicycles.

When building bicycles, manufacturers take advantage of the fact that bicycles share characteristics by building many bicycles from the same blueprint--it would be very inefficient to produce a new blueprint for every individual bicycle they manufactured

In object-oriented software, it's also possible to have many objects of the same kind that share characteristics: rectangles, employee records, video clips and so on. Like the bicycle manufacturers, you can take advantage of the fact that objects of the same kind are similar and you can create a blueprint for those objects. Software "blueprints" for objects is called classes.

The values for instance variables are provided by each instance of the class. So, after you've created the bicycle class, you must *instantiate* it (create an instance of it) before you can use it. When you create an instance of a class, you create an object of that type and the system allocates memory for the instance variables declared by the class. Then you can invoke the object's instance methods to make it do something. Instances of the same class share the same instance method implementations (method implementations are not duplicated on a per object basis), which reside in the class itself.

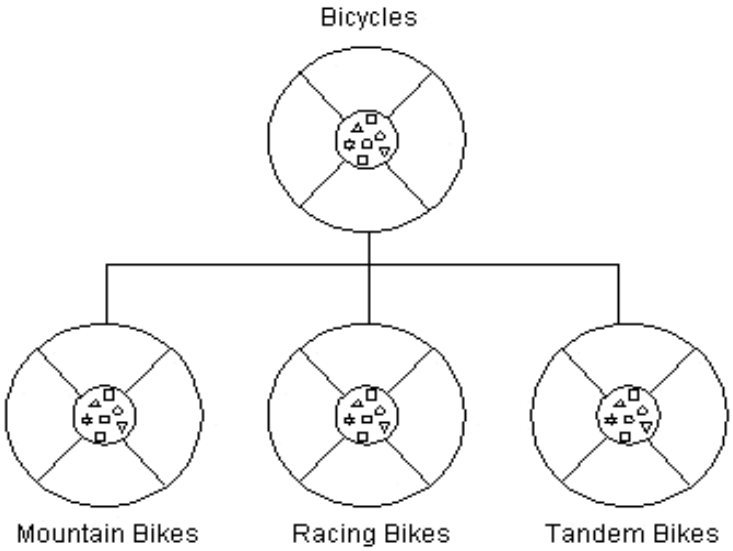
The Benefit of Classes

Objects provide the benefit of modularity and information hiding. Classes provide the benefit of reusability. Bicycle manufacturers reuse the same blueprint over and over again to build lots of bicycles. Software programmers use the same class, and thus the same code, over and over again to create many objects.

What Is Inheritance?

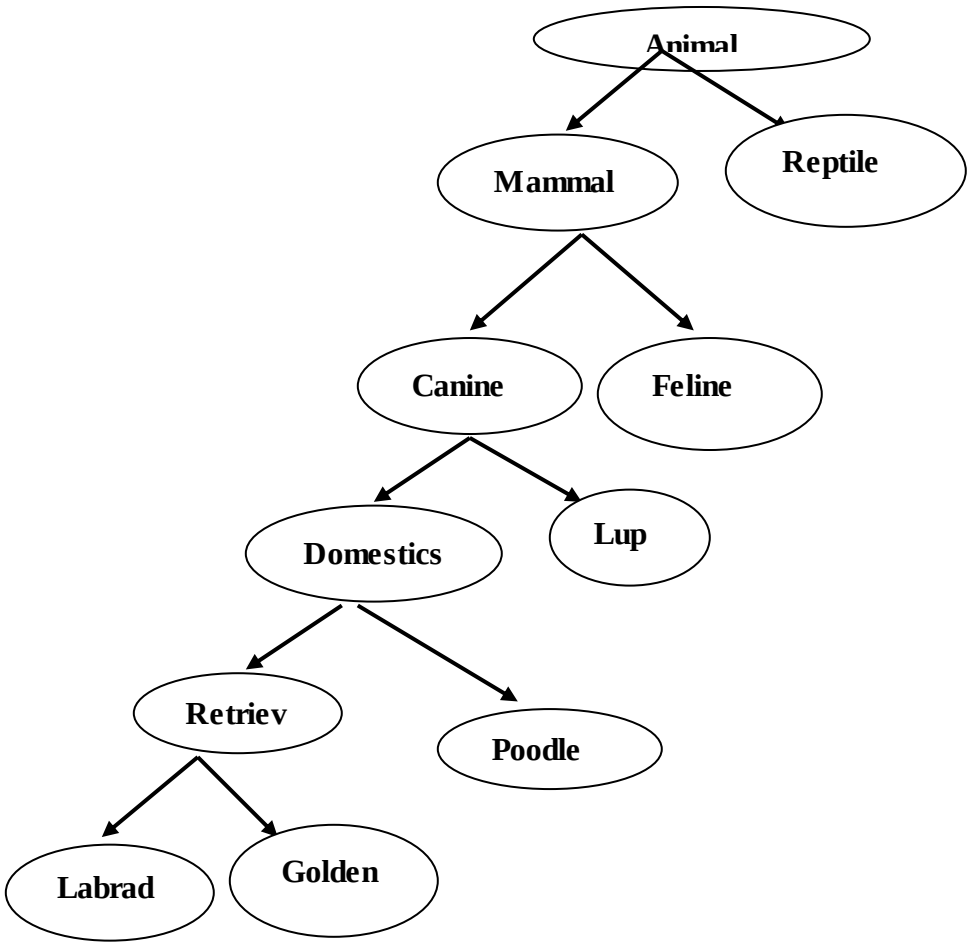
Inheritance is a process by which one object acquires the properties of another object. Inheritance supports the concepts of hierarchical classification.

> Object-oriented systems allow classes to be defined in terms of other classes. For example, mountain bikes, racing and tandem bikes are all different kinds of bicycles. In object-oriented terminology, mountain bikes, racing bikes, and tandem are all derived class of the bicycle class. Similarly, the bicycle class is the base class of mountain bikes, racing bikes and tandem bikes.



Each derived class inherits state (in the form of variable declarations) from the base class. Mountain bikes, racing bikes, and tandems share some states: cadence, speed, and the like. Also, each subclass inherits methods from the base class. Mountain bikes, racing bikes, and tandems share some behaviors: braking and changing pedaling speed, for example.

However, derived classes are not limited to the state and behaviors provided to them by their base class. Derived classes can add variables and methods to the ones they inherit from the base class. Tandem bicycles have two seats and two sets of handle bars; some mountain bikes have an extra set of gears with a lower gear ratio.



The Benefits of Inheritance

- The use of inheritance, programmers can reuse the code in the base class many times.

Encapsulation: is the mechanism that binds together variables and functions and keep safe from outside misuse. Encapsulation is used to hide unimportant implementation details from other

objects. When you want to change gears on your bicycle, you don't need to know how the gear mechanism works, you just need to know which lever to move. Similarly in software programs, you don't need to know how a class is implemented; you just need to know which methods to invoke.

The Benefits of Encapsulation

Modularity:- The source code for an object can be written and maintained independently of the source code for other objects. Also, an object can be easily passed around in the system. You can give your bicycle to someone else and it will still work.

Information hiding:- If a class has private members than no other members outside that class use that member. If a class has public member than any members outside that class can use that members.

Polymorphism: (One name many forms): (The occurrence of something in several different forms. Polymorphism is a feature that allows one interface to be used for a general class of action.

Dog's sense of smell is polymorphism. If the dog smells a cat, it will bark and run after it. If the dog smells its food, it will salivate and run to its bowl. The same sense of smell is at work in both situations. The difference is what is being smelled, that is, the type of data being operated upon by dog's nose.

CLASS: - A class is a collection of variables (data) and methods (code, functions) that are common to certain objects. A class defines the shape and behavior of an object and is a template for multiple objects with similar feature. Class is a basic element of Object – Oriented Programming.

A class is a collection of similar objects.

A class is a blueprint or prototype that defines the variables and methods common to all objects of a certain type.

A class is a user defined data type.