Fundamentals Of OOP

Samyak Jain
Jul 23 · 6 min read

OOP is considered the crown jewel of computer science. Essentially for building complex applications, it is considered as the ultimate solution for code organization. So **what is this OOP?** thing trending,

> *Object-oriented programming (or OOP) is a paradigm or pattern of programming where the solution to a programming problem is modelled as a collection of collaborating* objects. *It is most suitable for managing large, complex problems.*

There are 4 major principles that make a language Object Oriented. These are Encapsulation, Data Abstraction, Polymorphism, and Inheritance. Before diving into the pillars let's quickly understand what are classes and objects.

. . .

CLASS

> Class is the blueprint or detailed description of how the object will be.It is like a Template for an object.

- Classes have the following characteristics-

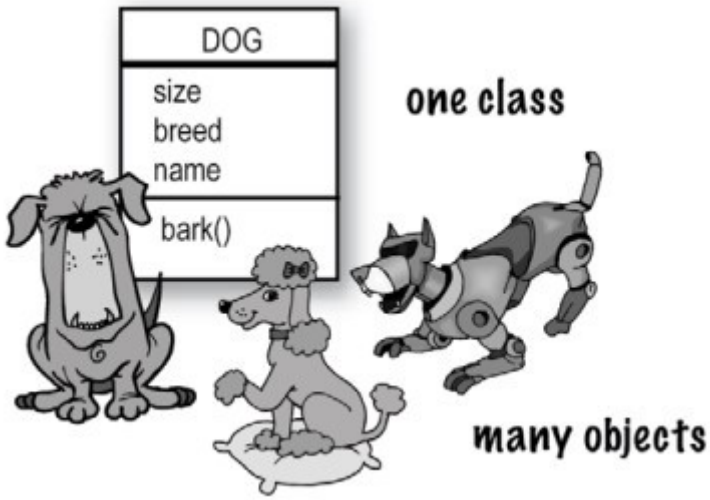
Name(What is it), **Attributes**(How the objects are described), **Operations**(What can the objects do).

OBJECT

> *Object is a real world entity in object oriented environment that may have physical or conceptual existence.*

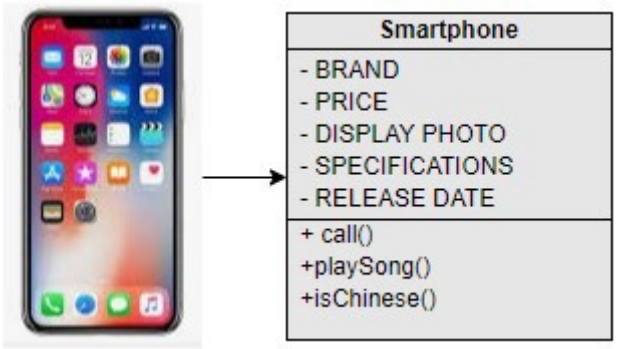
- An object is an instance of a class i.e. specific occurrence of a class.

- Class is a logical Construct while an object has physical reality i.e. an object occupies space in memory.



Beautiful Illustration from HEAD FIRST JAVA

Encapsulation



Depiction of Encapsulation(P.S-Sorry, I am not a good designer)

> *Idea of encapsulation is to surround the attributes and behaviour of object , keeping related functionality together and to protect them from outside world.*

- Encapsulation leads to data hiding.
- We use access specifiers to hide the data(properties) and methods. Use of `private` and `public` comes under this.
- Why it is important to hide variables if we are anyway exposing them through methods? This is a very common question and here's the explanation

What if your client while integrating your code somewhere uses:

mobileObject.price = -1000;

That would be a disaster. Your phone will be sold out in seconds but not for good. So how will implementing method help here? We can use validations and proper authentication in methods that alter our mobile object.

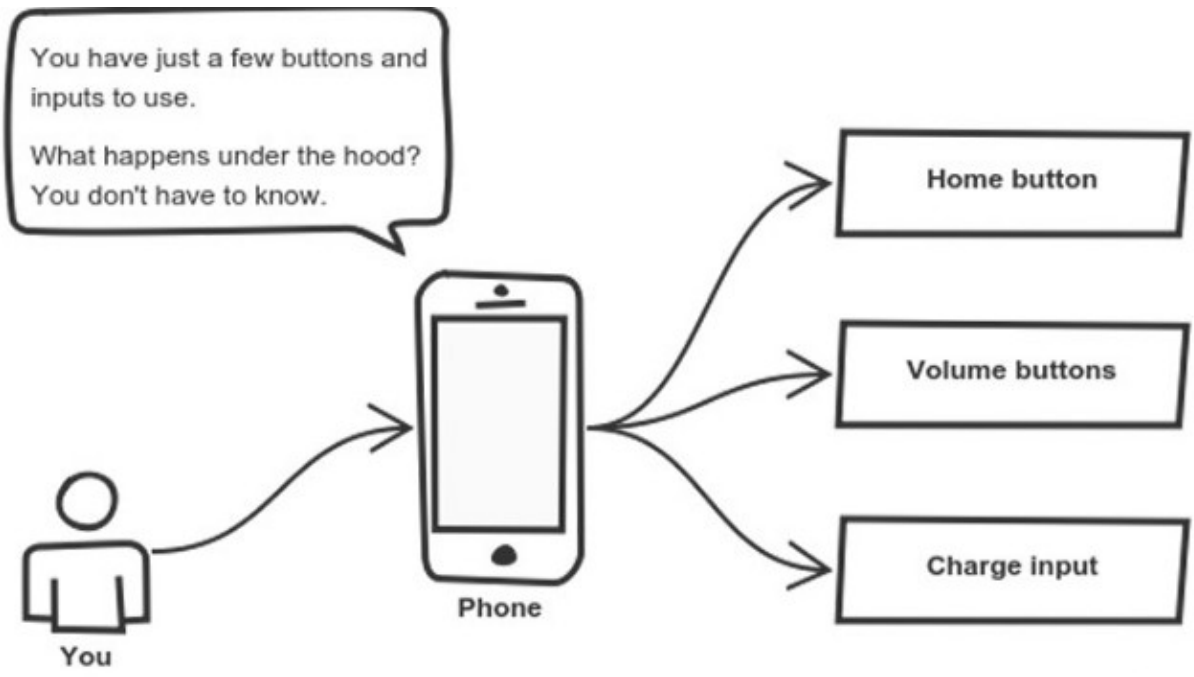
public void setPrice(int price){

if(price<0 OR InvalidUser()) return;

}

- **Advantages-**Data hiding, Maintainability, Data Consistency

Abstraction



Abstraction Illustrated(from freecodecamp.org)

> *Abstraction is a concept that aims to expose only high-level details of functionalities for the users and hiding all the background/implementation details.*

- **Examples** -Software libraries (Collections/STL), API Providers, JVM are good examples of abstraction. They tell you the methods you can use without telling you

how they are implemented. The TCP/IP stack built into your operating system abstracts away the details of transmitting bits over a network.

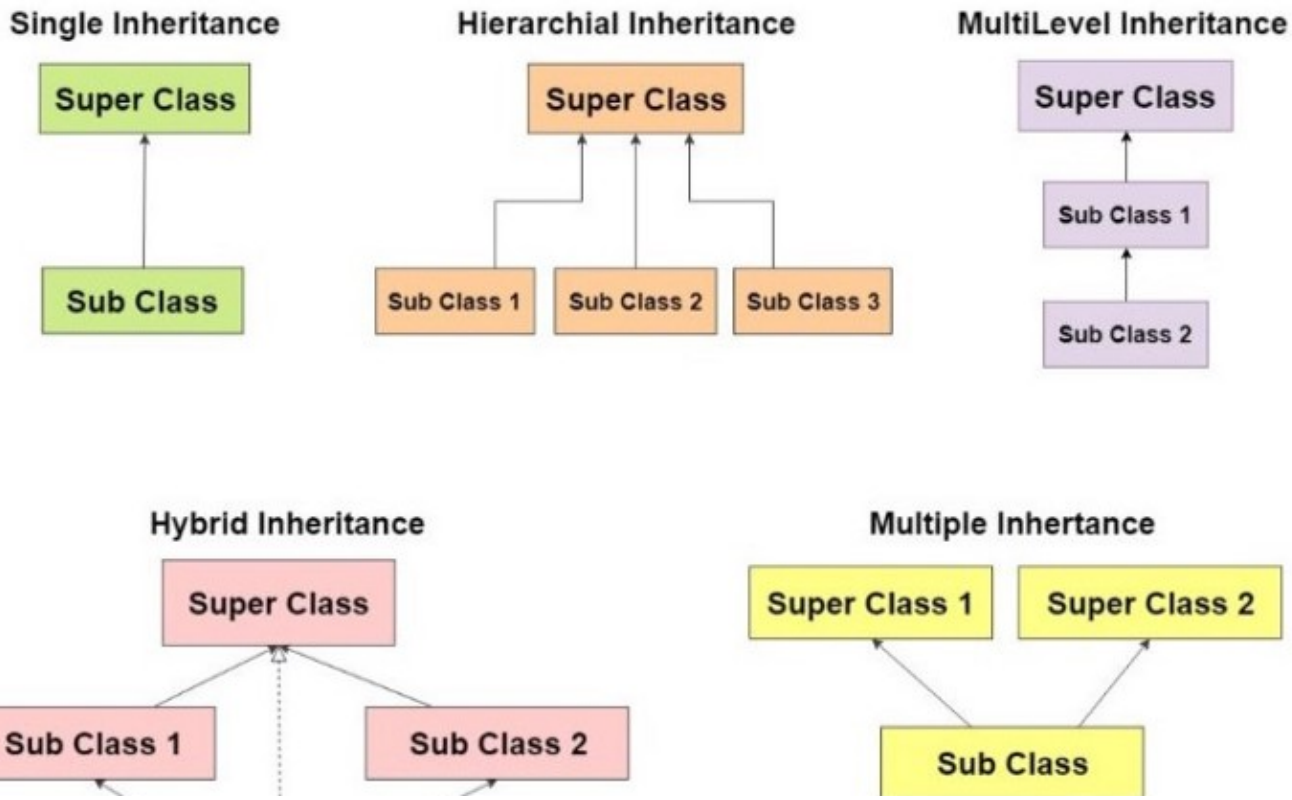
- An abstract class or interface is a good way to implement Abstraction. It separates the method signature from its implementation.
- **Advantages**- Flexibility, Code Reusability, Simplifies Code.

Inheritance

> *Inheritance describes **IS A** Relationship.*

- Example-A Car(derived) is a Vehicle(Base).Corona(derived) is a Virus(Base). For Bollywood fans-Remember how **KRRISH** inherited his powers From his father Rohit(**Koi mil Gaya**) or if you can relate how Mahendra **Bahubali** inherited power from Amarendra **Bahubali(After katappa killed him…)**
- **When to use**-If you have a class and you want another class, similar to the first but expanded with new attributes or functionalities, or you may want one or more methods to work differently then inheritance comes in.
- **Advantages-**Code Reusability, Easy Maintainability

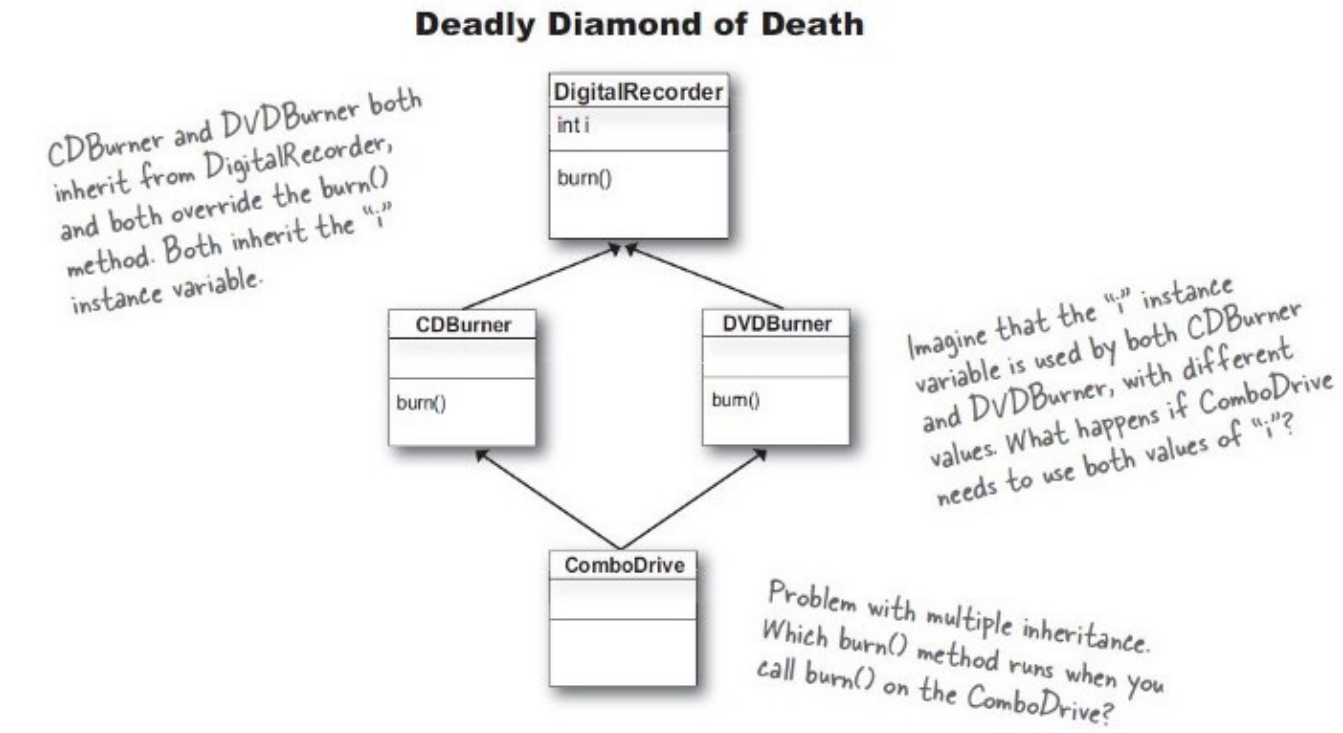
Types Of Inheritance:



1. Single Inheritance-1 parent, 1 child
2. Multiple Inheritance-2 parents, 1 child
3. Multi-Level Inheritance-1 grandparent,1 parent, 1 child,….so on
4. Hierarchical Inheritance-1 parent, multiple children
5. Hybrid Inheritance-Combination of more than one types

Why Multiple Inheritance is not supported in JAVA?

Due to “Deadly Diamond of Death”. What is it and what problems are there? Have a look. Explore more on how C++ resolves this issue.



Deadly Diamond of death-taken from HEAD FIRST JAVA

POLYMORPHISM

I know many of you are here for this topic only. So let me give you an idea and then provide you with a more formal definition.

For a basic idea what polymorphism is:

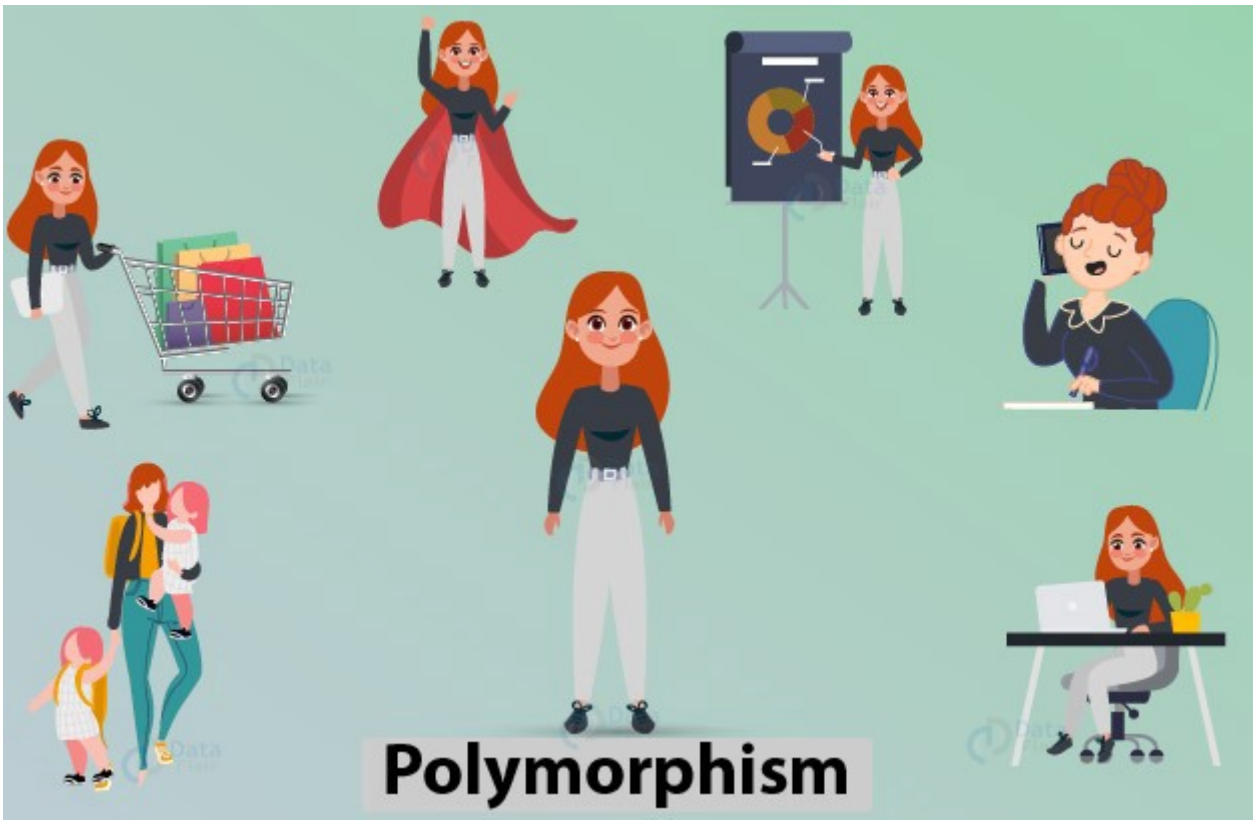
If anybody says the word *CLASS* to these people

1. Kid
2. Software Developer
3. Actor

What will happen?

- The Kid would think of a class full of students and a teacher teaching.
- The Software developer would think of a complex project he did using OOPS.
- The Actor would think of a **Social class** (i.e. to a group of people with similar levels of wealth, influence, and status).

Did you notice how the word *CLASS* changed its meaning depending on whom you are speaking too? Same functionality we can apply on methods i.e. **about being able to change the functionality offered by a method, depending on the situation.**



MOM is the best example of Polymorphism, especially in this lockdown. SALUTE!

> *Polymorphism is the ability to take more than one form by methods and variables.*

- **When to use** — when child classes need to implement their logic for methods provided by the parent class. Or when a class needs related methods to have the same method name with only a different set of parameters.

- Polymorphism is achieved in two ways

1. Method Overriding(Dynamic polymorphism) — Child class implements a method that is already present in the parent class, resulting in the child method being invoked.

2. Method Overloading (Static polymorphism) — Writing multiple methods with the same name but with different sets of parameters (or order of parameters) so that the appropriate method gets invoked based on the parameters passed.

> ***QUICK TIP*** *— In Java following methods cannot be overrided*
>
> *FINAL,PRIVATE,STATIC(Can be re-declared),CONSTRUCTOR*

- **Advantages**-Maintainability, Code Reusability, Clean Code

You might think about Encapsulation vs. Abstraction

- The difference between concepts of encapsulation and abstraction is that encapsulation is about the packaging of the class (like how data should be accessed (setters/getters) and what data should be accessed (access specifiers)), whereas abstraction more about what the class does for you at a conceptual level.

- Abstraction is the process of refining away all the unneeded/unimportant attributes of an object and keeps only the characteristics best suitable for your domain.E.g. for a person: you decide to keep first and last name and SSN. Age, height, weight, etc are ignored as irrelevant.

- **Abstraction is a natural extension of encapsulation.**

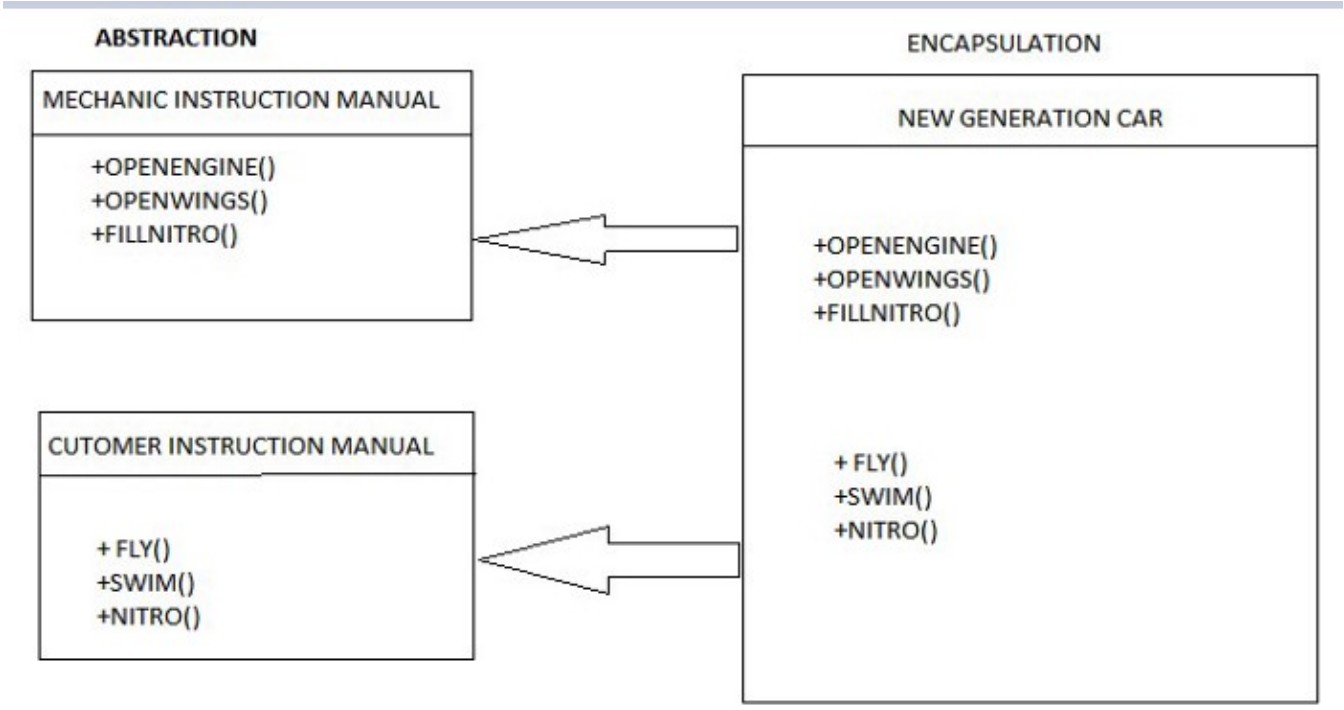
- Still Confused? Don't worry I have a crystal clear analogy for you.

Let's say we are a car manufacturing company and we designed a futuristic car(Like cyberTruck) with ample of features.

We Encapsulate all the features in a car class.

Now we need to provide specific instructions to our customers on how to use our exciting features. So we define an interface with all customer-relevant features.

Also, we need to specify all the internal details of the car to help the Mechanic to find and fix the issues. For this, we define another interface.



Encapsulation-All the functions available in the car class

Abstraction-Subset of functions present in class

. . .

Now, GO AHEAD and surprise the interviewer with your super clear concepts and use these pillars to model real-world applications. KEEP LEARNING!!

For any queries feel free to reach out to me via mail (samyak915jain@gmail.com)or in comments!!. Thanks

Oop Polymorphism Inheritance Abstraction Computer Science