**What are objects and why should we use this particular programming paradigm?**

**Object-oriented programming (OOP)** is a way of writing computer programs that use the idea of “objects” to represent data and methods.

It may sound a bit scary, but we’re going to break this down into tiny lego blocks.

First and foremost, we will look at the major keys words associated with Object-Oriented programming, “**Classes**” and “**Objects**”

**Classes**

**Classes** are like the sand baking sets that you can use to mold sand into various shapes. Classes are basically these plastics that have a specific shape. 

**Objects**

**Objects** are the molded sand we get from these plastics. Why are they objects? – because they inherit every bit of curve, size, and corner of the plastic.



Now that we have some understanding of these concepts, we can dive a bit more into the basic principles of OOP.

**Principles of OOP**

The four basic principles of object-oriented programming are abstraction, encapsulation, inheritance, and polymorphism.

These may sound a bit off, but the point of this article is to break OOP concepts into small bits that could be used as a reference while programming.

**Abstraction**

There are so many concepts that could be used to explain abstraction but think of it as a simple espresso machine.

We don’t really know what goes on in there when we press the on button, but we are very certain we will get coffee in 5-10 mins.



That’s what abstraction is, we don’t need to know the complete logic of how something works, our job is just to use it. Easy, right?

We can also think of abstraction when we use our cell phones. By just pressing a few buttons, we get work done without actually knowing how it all happened.

**Encapsulation**

**Encapsulation** in OOP is like a portfolio holding all the properties and actions of a specific item.

We can think of encapsulation as a video game. We can see everything that a character possesses and everything they’re capable of doing for each character, respectively.

**Polymorphism**

**Polymorphism**, in clear terms, means many forms.

A command could give birth to many forms or styles of output. For instance, four different animals are on a show and are asked to speak.



From here we see that they hear the same command but give different sounds according to their type.

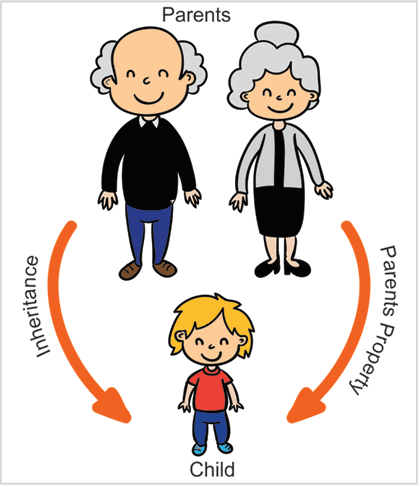
Another instance could be a circle and a rectangle having different calculations of the same concept (e.g., area).

 Woooh, that was so tiring. But we’re almost done!

**Inheritance**

This should be one of the simplest concepts for anyone to remember. Why? – because we’re using FAMILY to explain this.

**Inheritance** is basically acquiring traits or features from your parents.



Tommy has blonde hair and a slightly pointed nose, we could say that Tommy **inherited** these properties from his father or mother, but it doesn’t end there.

Inheritance in OOP gives objects (Tommy) the ability to possess their own properties outside the ones inherited.

That explains why Tommy knows how to play football but neither of his parents can.

That rounds up this very interesting read!