What we learnt so far

- A real-life scenario can be modelled using objects.
- All **objects** that share similar characteristics or behaviours, that are of the same kind, belong to the same **class.**
- Object Oriented languages provide the features to implement an object-oriented model.

Object Oriented Principles we saw so far

- **Encapsulation**

It keeps the data and the code safe from external interference. It is a mechanism for restricting direct access to some of the object's component. Binding the data with the code that manipulates it.

- **Inheritance**

Inheritance allows a class to use the properties and methods of another class. In other words, the derived class inherits the states and behaviors from the base class.

- **Polymorphism**.

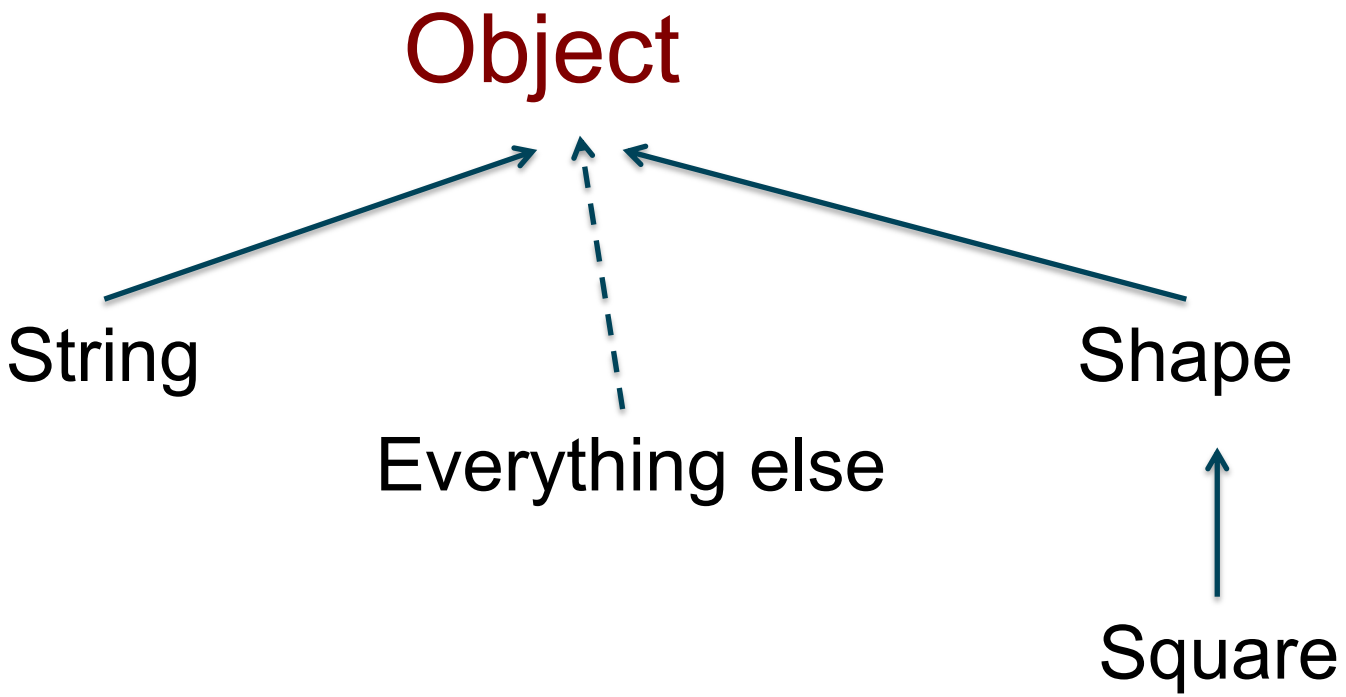
Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object.

Summary

- Object Class in Java
- Final classes and methods
- Abstract classes
- Interfaces
- Introduction to design patterns

Java Inheritance Tree

- All Java classes you ever use or write yourself are in the *inheritance tree* with class Object at the top:



Everything is an Object

```
Object obj = new Square() ;
```

- OK
- But can only call methods declared by class Object.
- Of course, they may be overridden by subclasses.

The Object Class

- Every java class has Object as its superclass and thus inherits the Object methods.
- Object is a non-abstract class (You will see in a few slides the meaning of Abstract class)
- Many Object methods, however, have implementations that aren't particularly useful in general
- In most cases, it is a good idea to override these methods with more useful versions.
- In other cases, it is **required** if you want your objects to correctly work with other class libraries.