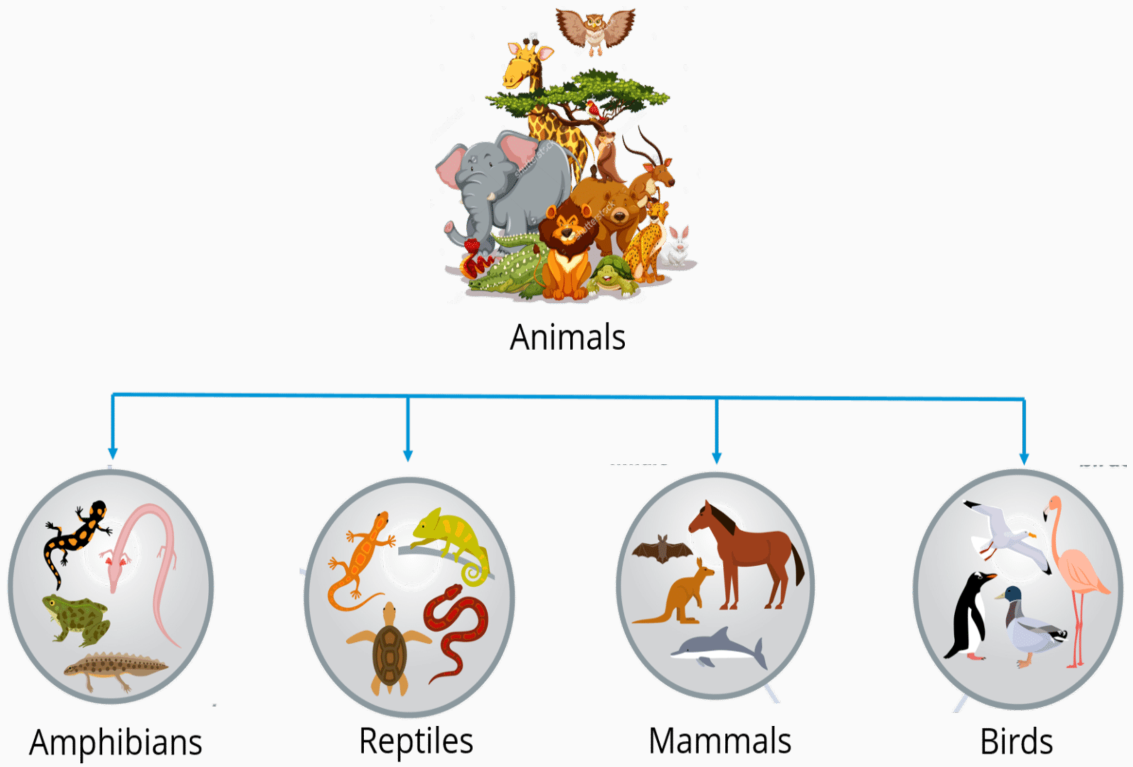
CYDEO

# Polymorphism

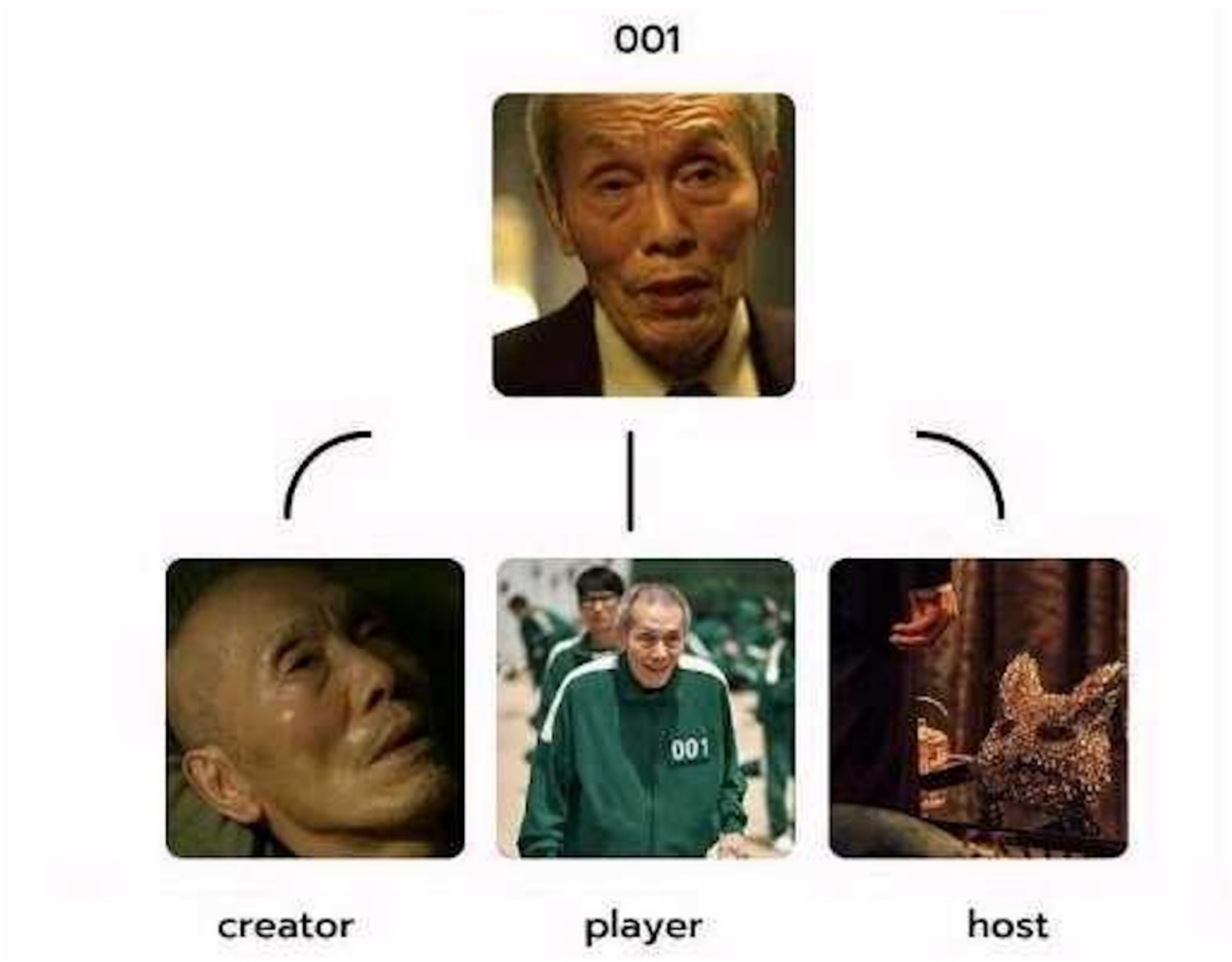
## OOP Principles

- There are 4 Object Oriented Programming (OOP) principles:
    - Encapsulation
    - Inheritance
    - Abstraction
    - Polymorphism

# What Is Polymorphism?

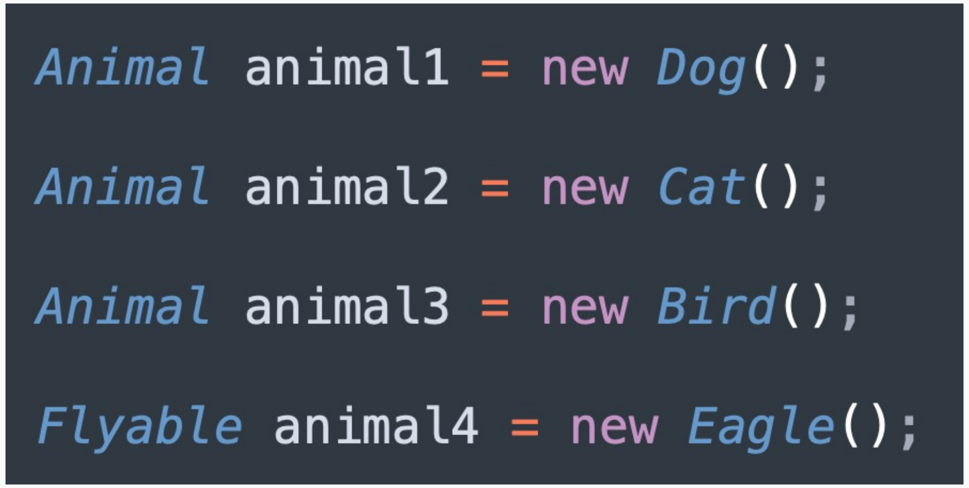


## What Is Polymorphism?



# Poly + Morphism (Many Forms)

- Ability of the objects to take on many forms
- Occurs when reference type is parent class/interface and object type is child



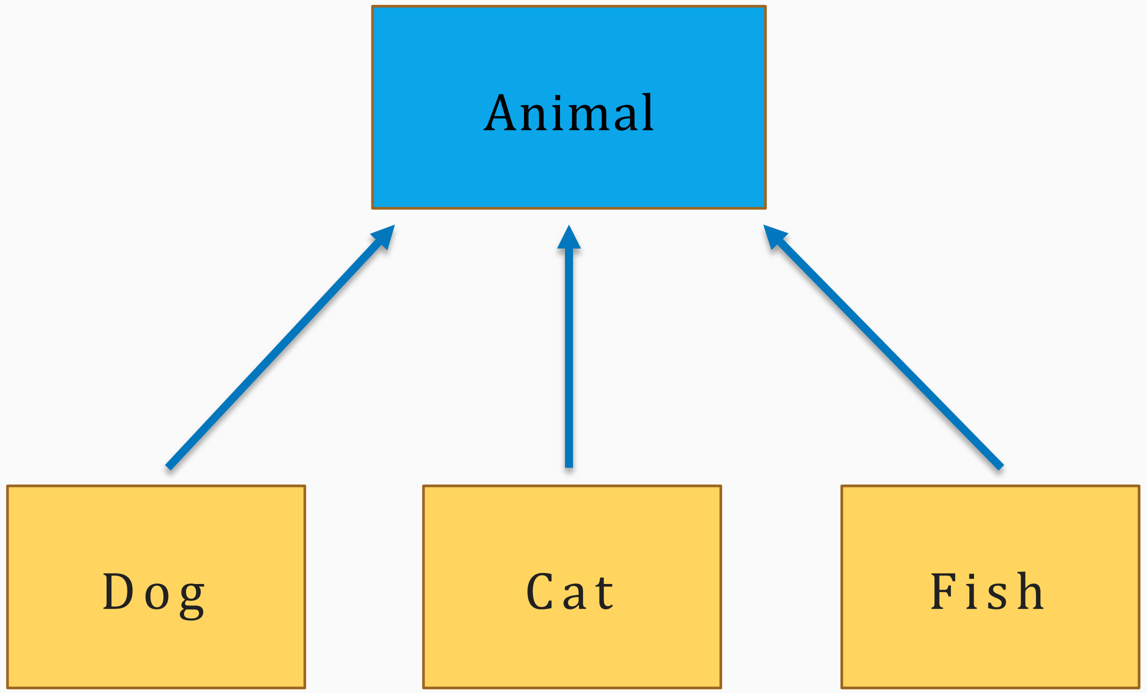
```java
Animal animal1 = new Dog();

Animal animal2 = new Cat();

Animal animal3 = new Bird();

Flyable animal4 = new Eagle();
```

## Polymorphism

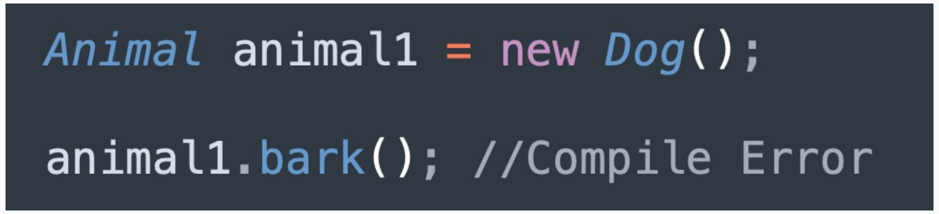


Reference Type      Object Type

```
Animal  animal1  =  new Dog();
Animal  animal2  =  new Cat();
Animal  animal3  =  new Fish();
```

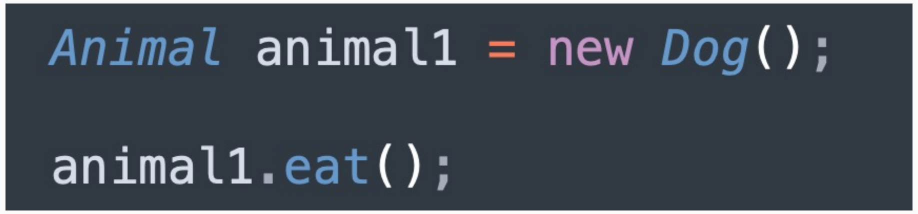
## Calling method in polymorphism

- Only the methods/variables in reference type can be called
- When we call a method, it will call overridden version from a child class
- If method is not overridden, it will call parent/super class version



```java
Animal animal1 = new Dog();

animal1.bark(); //Compile Error
```



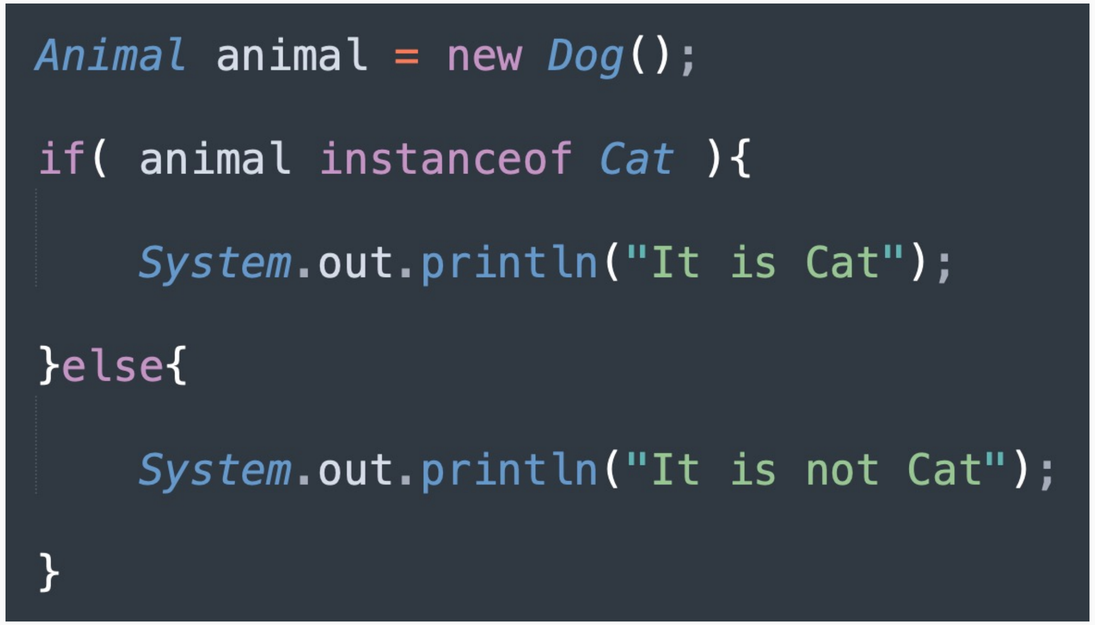
```java
Animal animal1 = new Dog();

animal1.eat();
```

CYDEO

## instanceof keyword

- Instanceof keyword can be used to check if the object is certain class. (Returns boolean)



```java
Animal animal = new Dog();

if( animal instanceof Cat ){

    System.out.println("It is Cat");

}else{

    System.out.println("It is not Cat");

}
```

CYDEO

## Polymorphism Rules

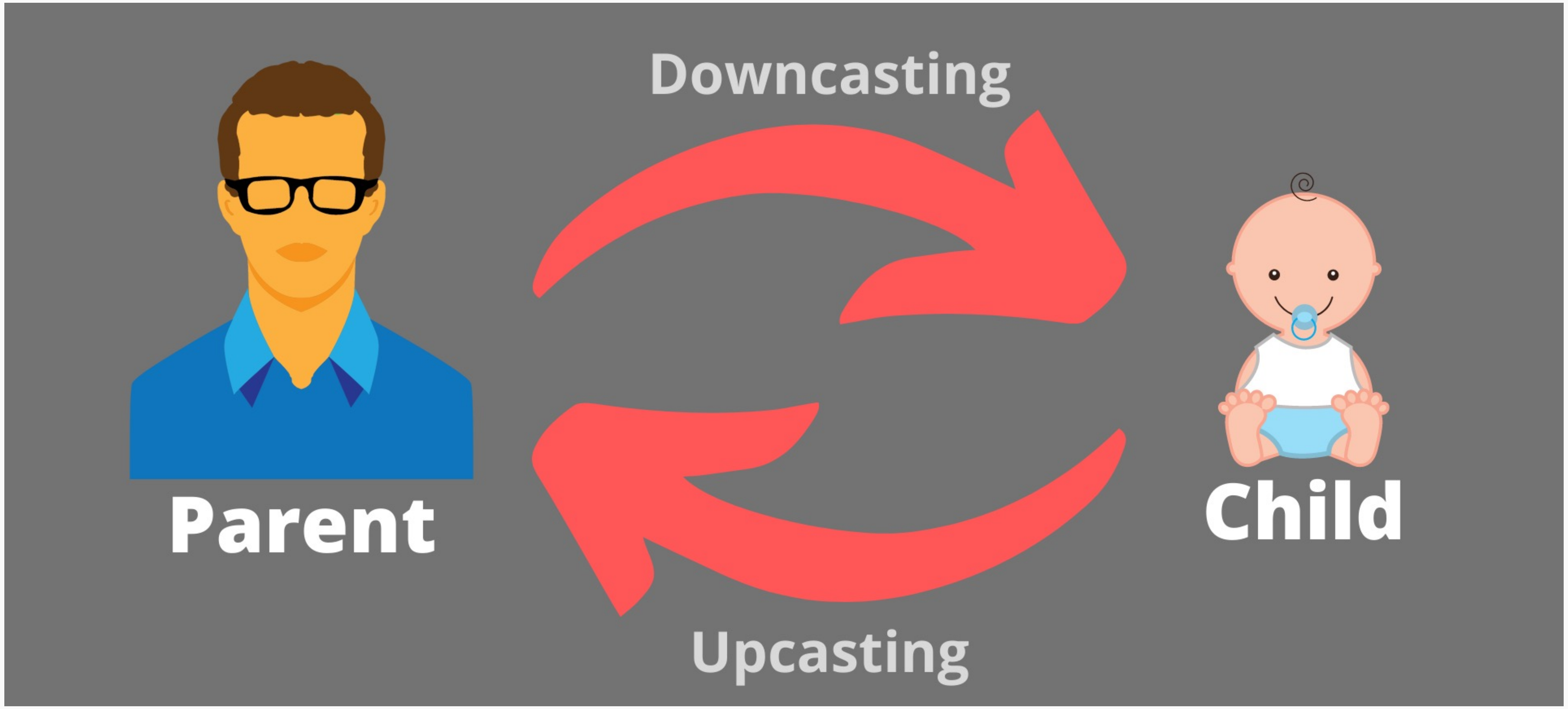
- Reference type can be parent class or interface
- Object type can be any extending or implementing child class
- Reference type decides what is accessible
- Object type decides which implementation of the method to be executed when the method is called

CYDEO

## Reference Type Castings

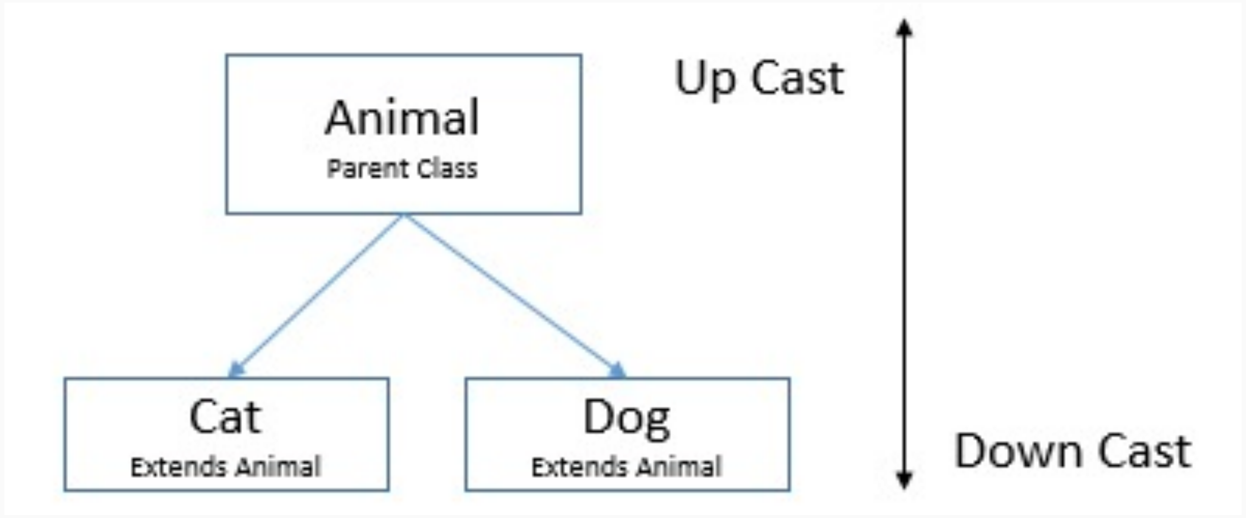
## What Are Reference Type Castings?

- Casting one reference type to another



## Reference Type Castings

- There must be IS A (inheritance) relation between the object type and reference type we are casting it to, otherwise ClassCastException will be thrown
- There are two types of reference type castings: upcasting and downcasting



CYDEO

## Upcasting

- Casting the smaller reference type (sub type) to larger reference type (super type)
- Upcasting is done implicitly and cast operator is not required to be given explicitly
- Allows us to achieve polymorphism without any explicit action

```java
Animal animal1 = new Cat(); //upcasting


Dog dog = new Dog();
Animal animal2 = dog; //upcasting
```

```java
Phone phone1 = new IPhone(); //upcasting


Samsung samsung = new Samsung();
Phone phone2 = samsung; //upcasting
```

# Downcasting

- Casting the larger reference type (super type) to smaller reference type (sub type)
- Downcasting is done explicitly and cast operator is required to be given explicitly
- Allows us to access the features of the objects type that are not in reference type

```
Animal animal = new Dog();

Dog dog = (Dog)animal; //downcasting
dog.bark();
```

OR

```
Animal animal = new Dog();

( (dog)animal ).bark(); //downcasting
```

CYDEO