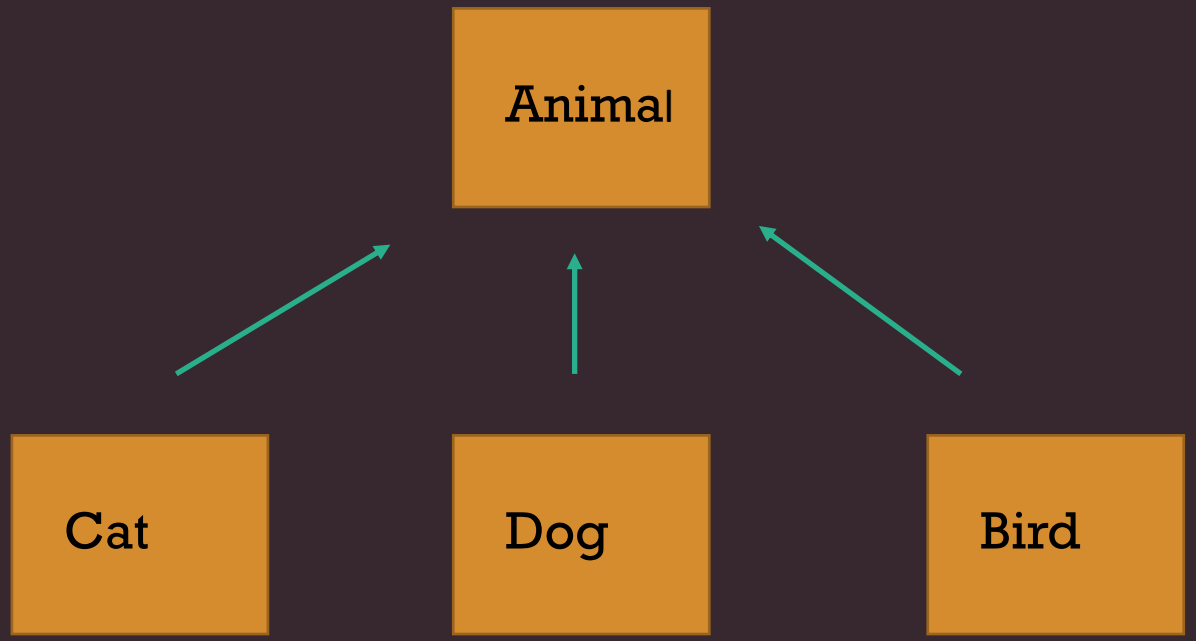
OOP PRINCIPLES

- Encapsulation
- Inheritance
- Abstraction
- **Polymorphism**

Poly + Morphism (Many Forms)

- Polymorphism is the ability of the object to take on many forms

- Polymorphism is when reference type is a parent class/interface and object type is child.



Reference Type		Object Type
Animal a	=	new Animal();
Animal a	=	new Dog();
Animal a	=	new Bird();

```java
public interface MyInterface{}

public class ClassA implements MyInterface{}

public class ClassB implements MyInterface{}

ClassA a = new ClassA();
ClassB b = new ClassB();

MyInterface a = new ClassA();
MyInterface b = new ClassB();
```

Calling method in polymorphism

- When we call a method, it will call overridden version from a child class, if we have overridden the method.

- If method is not overridden, it will call parent/super class version.

- getClass() method helps us to access the object information
- getClass().getName() : returns package.className of the object
- getClass().getSimpleName() : returns just the class name of the object

instanceof

- instanceof operator can be used to check if the object is certain class.

```
Animal a = new Dog();

if(a instanceof Dog){

    System.out.println("it is Dog")
}else{
    System.out.println("it is not Dog")
}
```

Rules

- Reference type can be parent or interface , object can be any extending or implementing child class

- Reference type decides what is accessible

Casting (down-casting)

- Instructs the compiler to change the existing type of an object reference to another type

Casting 

```
public class A{
    public void mA(){}
}

public class B extends A{
    public void mB(){}
}

A obj = new B();
obj.mA(); //is only accessible. Reference is A


B obj2 = (B) obj;

obj2.mA();
obj2.mB();
```