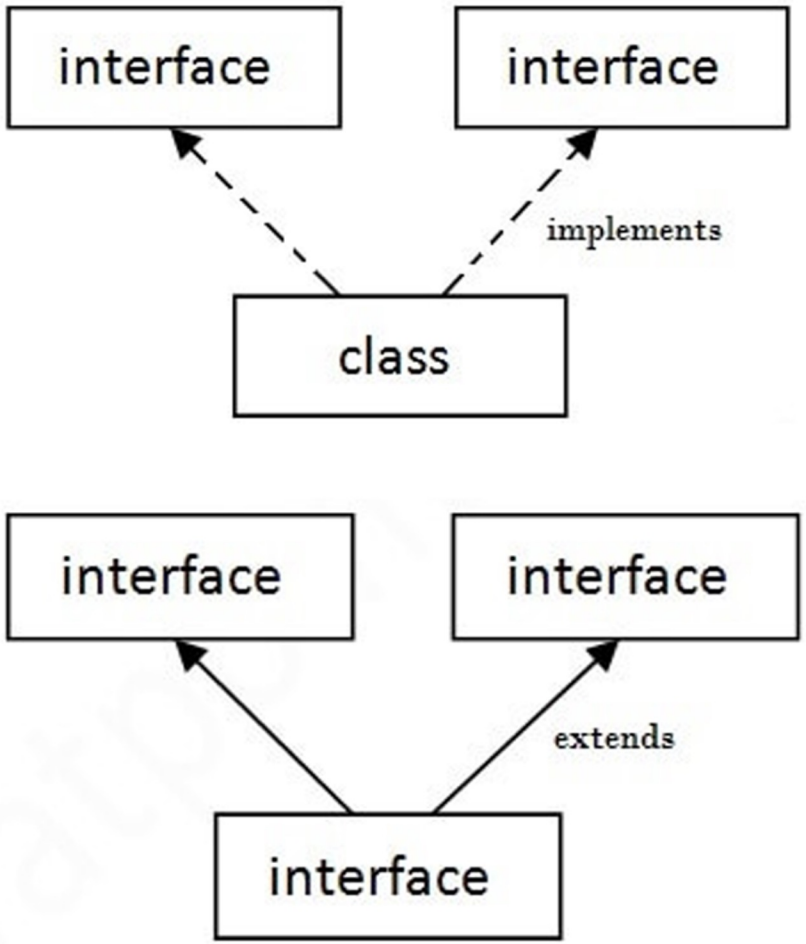
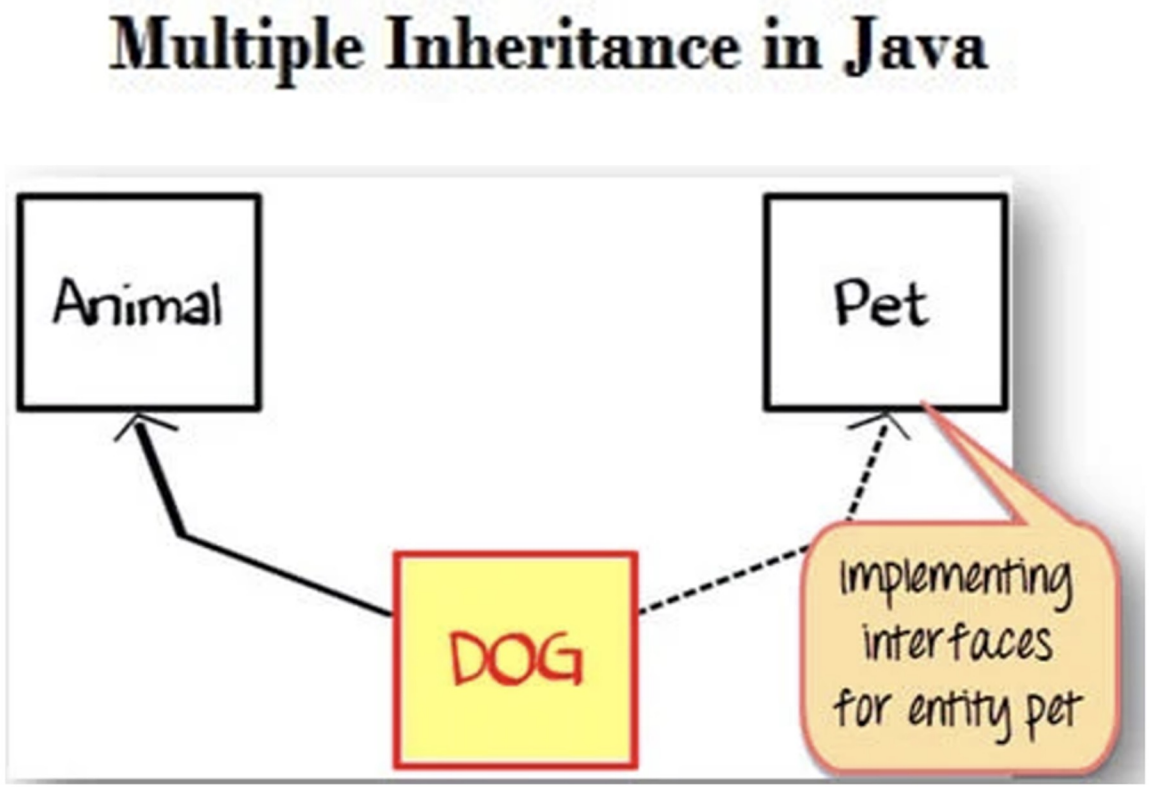
CYDEO

Abstraction

Interface

What Is An Interface?





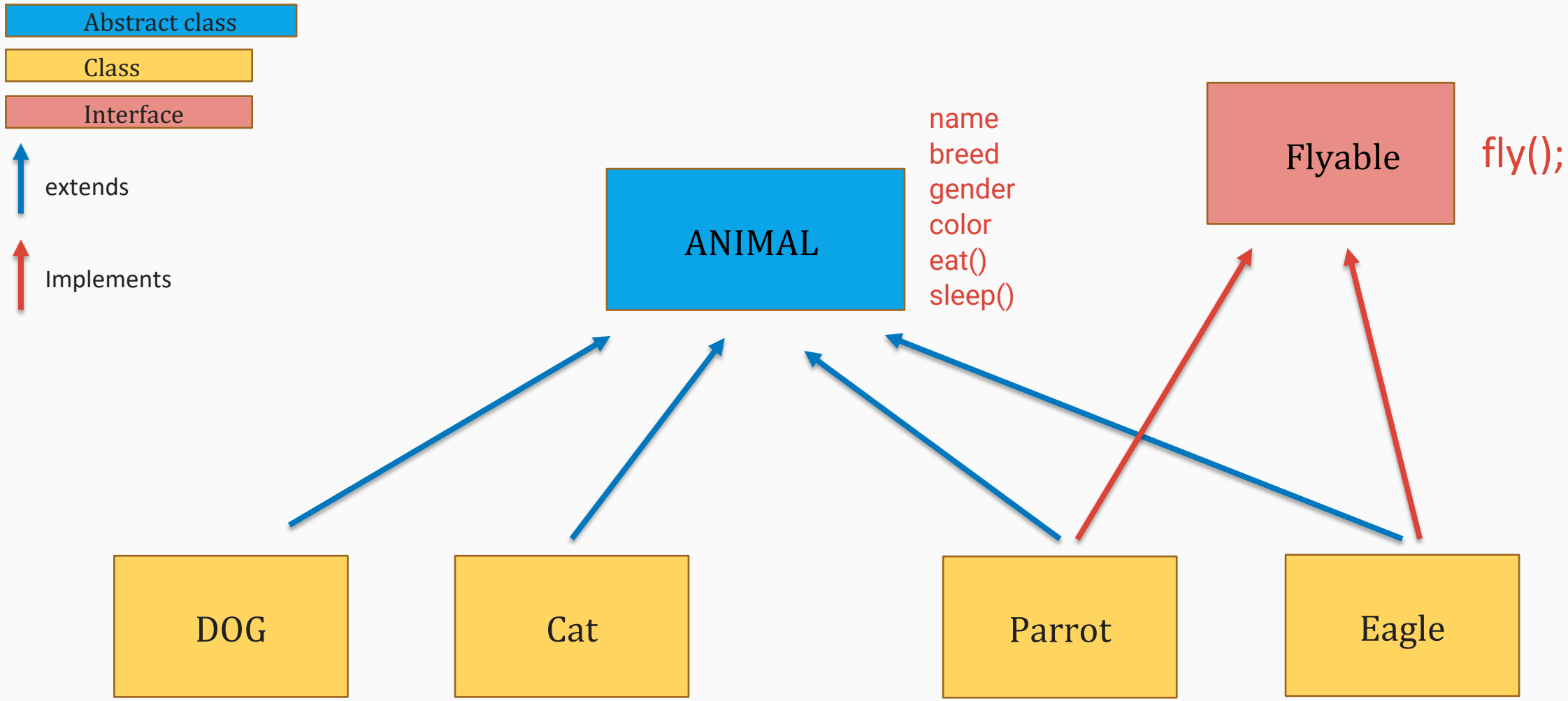
Interface

- It’s a template, a blueprint of a class
- Interface keyword is used to create interface
- Specifies the behavior(s) that a class should implement
- Provides additional methods that class(es) need.
- We can achieve 100% abstraction using interfaces

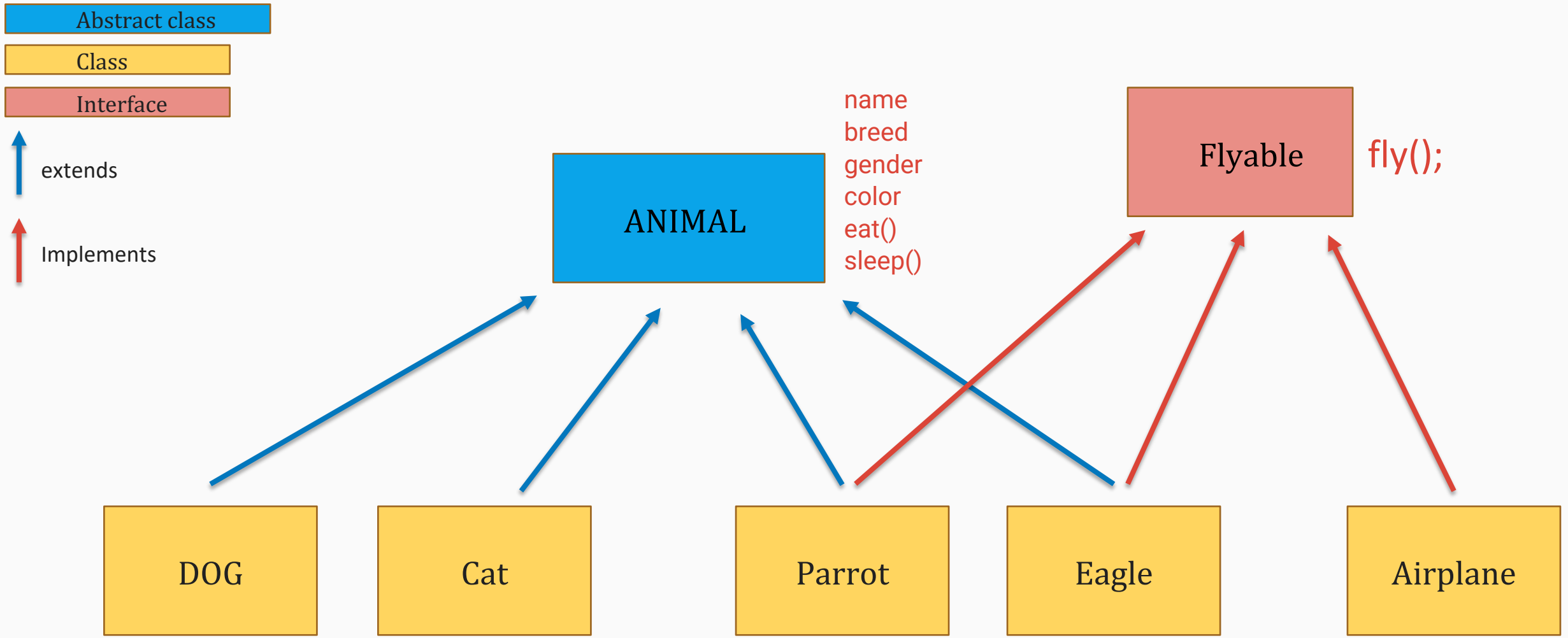
```
public interface Flyable {

    public abstract void fly();

}
```

CYDEO

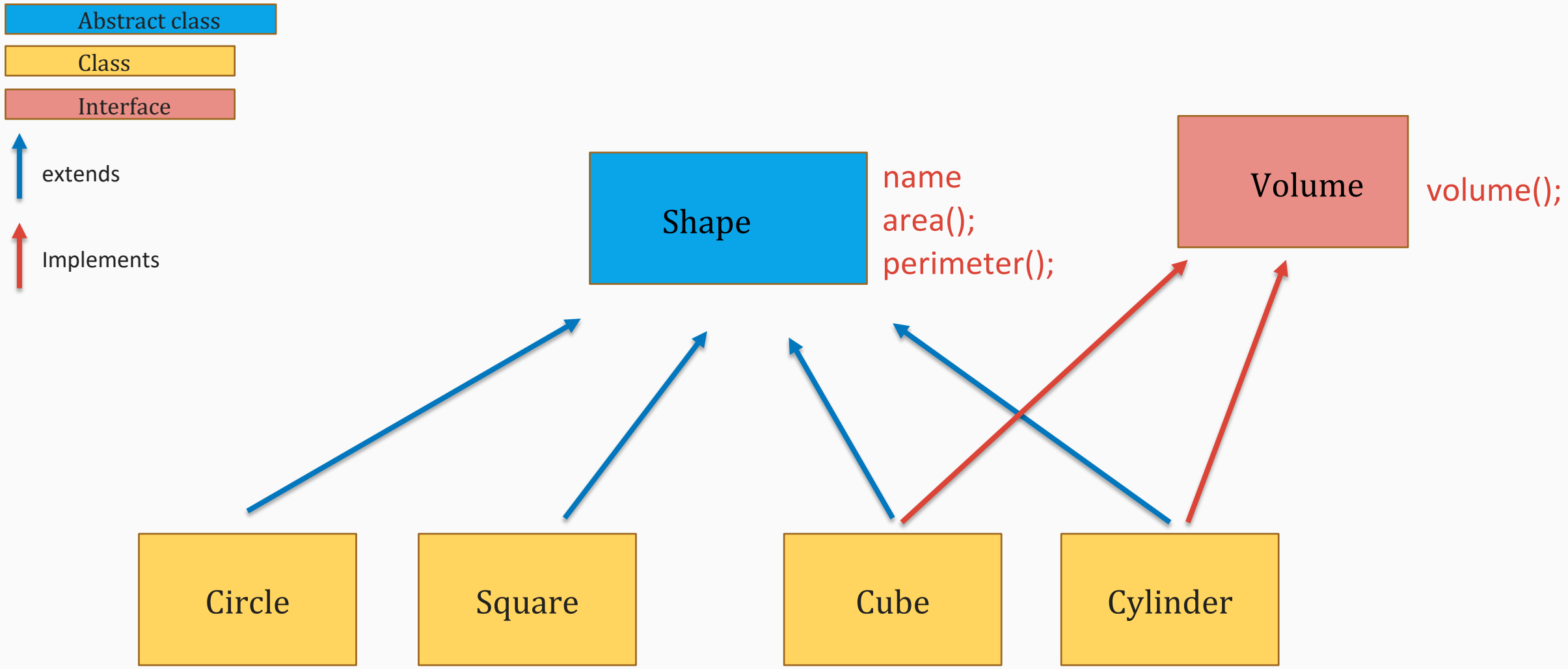
Interface Example



Interface Example



Interface Example



Properties of Interface

- Variables are static & final by default
- Interface can have static methods, abstract methods and default methods only
- Public is the only access modifier that can be used in interface (except for static methods) and given by default
- Abstract keyword is given by default to the abstract methods of interface

```
public interface Interface1{

    int a = 100; // static & final by default

    void method1(); //abstract method

}
```

CYDEO

What interface can have?

```java
public interface Interface1{

    int a = 100; // static & final

    void method1(); //abstract method

    static void method2(){ // static method
        System.out.println("Static Method");
    }

    default void method3(){ // default method
        System.out.println("Static Method");
    }


}
```

Variables: static & final variable only

Methods: static, abstract and default methods only

What Interface can not have?

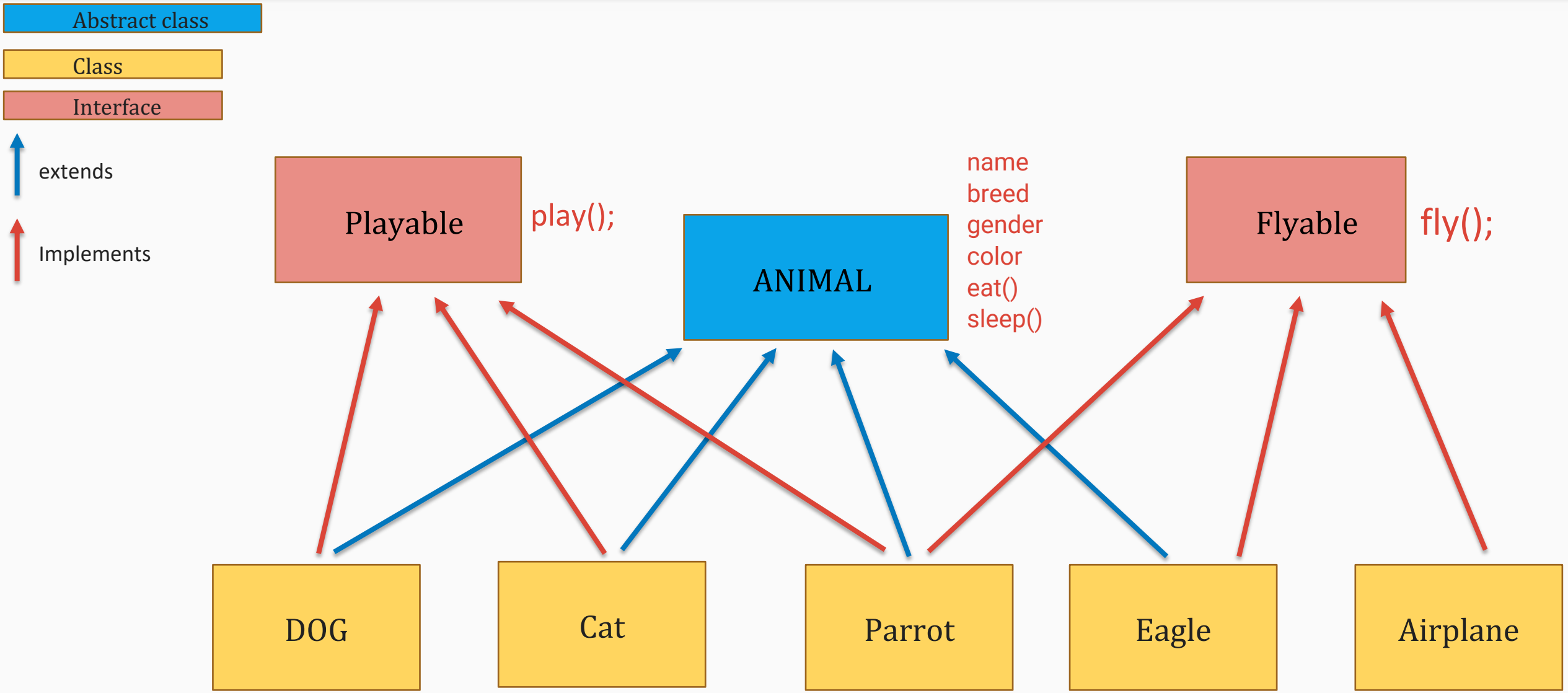
- An interface can not have instance variables
- An interface can not have instance methods
- An interface can not have constructors
- An interface can not have initializer blocks
- We can not create objects from interface

Implementing the Interface

- Class can extend only one superclass, but java allows a class to implement multiple interfaces
- Implements keyword is used to inherit from interface(s)
- When a class implements multiple interfaces, it must implement (override) all the abstract methods

```
public class MyClass implements Interface1, Interface2, Interface3{}
```

Interface Example



CYDEO

Implementing the interface

```java
public abstract class Animal{

    public String name;
    public String breed;
    public char gender;
    public String color;

    public abstract void eat();

}
```

```java
public interface Flyable{

    boolean canFly = true;

    void fly();

}
```

```java
public class Eagle extends Animal implements Flyable{

    @Override
    public void eat(){
        System.out.println("Eagle eats snake");
    }

    @Override
    public void fly(){
        System.out.println("Eagle can fly 50 km/h");
    }

}
```

Implementing the interface

```
public abstract class Shape{

    public abstract double area();

    public abstract double perimeter();

}
```

```
public interface Volume{

    boolean hasVolume = true;

    double volume(); // abstract method

}
```

```
public class Cube extends Shape implements Volume{

    public double side;

    @Override
    public double area(){
        return 6 * side * side;
    }

    @Override
    public double perimeter(){
        return side * 12;
    }

    @Override
    public double volume(){
        return side * side * side;
    }

}
```

Abstract Class vs Interface

Abstract class	Interface
Can not be instantiated	Can not be instantiated
Multiple inheritance is not allowed	Multiple inheritance is allowed
Can have constructor	Can not have constructor
Can have instance, static and abstract methods	Can have static, abstract and default methods
Can have instance and static variables	Can only have static variable (final by default)
Can not be final	Can not be final
Can use other access modifiers than public	Can not use the access modifiers other than public & private (for static methods)