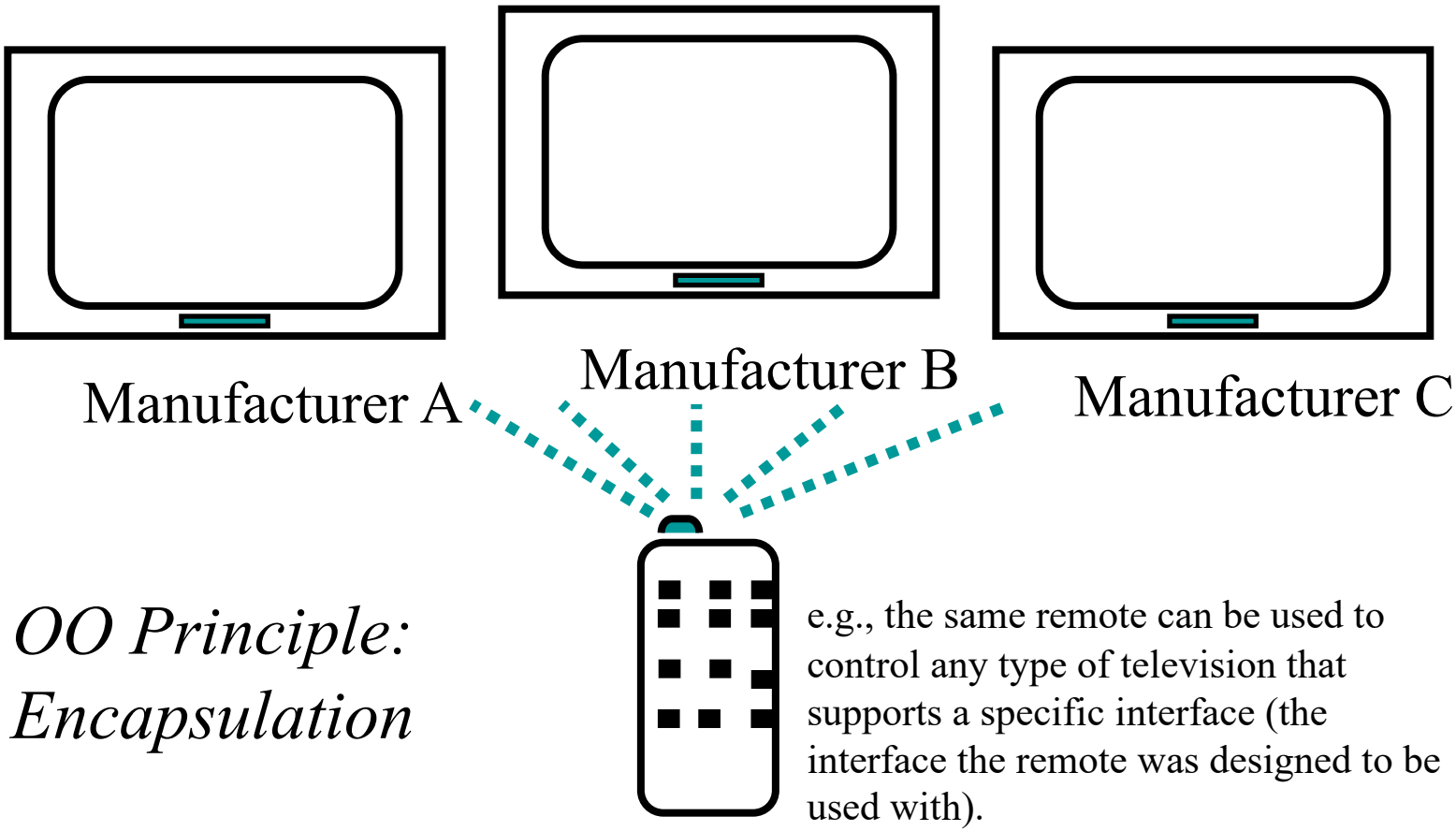
What is Polymorphism?
(A Closer Look)

Polymorphism

- In Biology, the term polymorphism describes the characteristic of an element that may take on different forms.
- In other words, the occurrence of more than one *form* or *morphs*.
- Then, what does it mean in object-oriented programming?

What Is Polymorphism?

- The ability to hide many different implementations behind a single interface



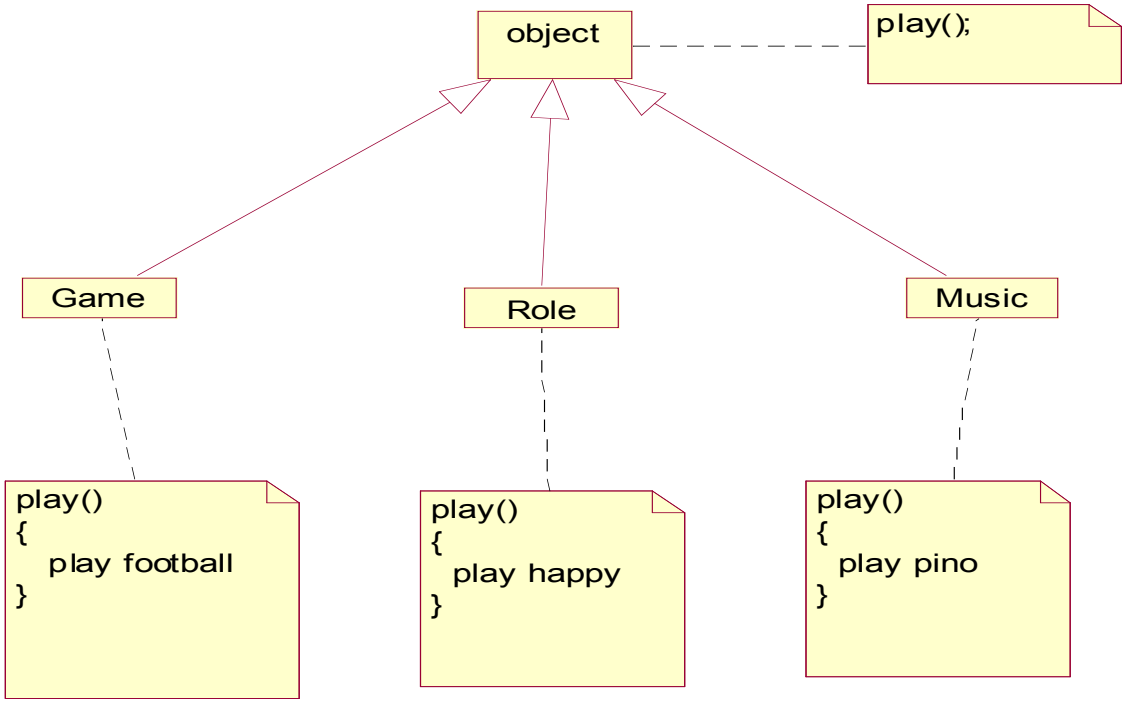
OO Principle: Encapsulation

e.g., the same remote can be used to control any type of television that supports a specific interface (the interface the remote was designed to be used with).

Figure From: IBM Rational Rose

Polymorphism in Object-Oriented Methodology

- In O.O. languages, polymorphism is the property that different objects may react to the same message, differently.
- Let's consider this hypothetical situation:



- The following statements will receive the same message 'play' but implement different operations:
  - game.play()
  - music.play()
  - role.play

Implementation of Polymorphism in Java (Quick Review)

Example of Polymorphism in Java

```
class Shape {
  public Shape(double x, double y) {
        origin = new Point (x, y);
  }
  public abstract double area();
  private Point origin;
}
class Rectangle extend Shape{
  public Rectangle(…) {    }
  public double area() { return width * height;}
 private double width, height;
}

class Circle extend Shape{
  public Circle(…) {    }
  public double area() { return radius*radius*PI; }
 private double radius;
}
```

```
class User {
   public double area(Shape x) {
       return x.area();
    }

   public static void main(String []s) {
       User user = new User();
       Rectangle r = new Rectangle(6, 7, 8, 9);
       Circle c = new Circle(2, 3, 4);
       System.out.println(user.area(r));
       System.out.println(user.area(c));
    }

};
```

Implementation of Polymorphism in C++ (Quick Review)

Implementation of Polymorphism in C++

- There is lots similarities in principles of polymorphism concept implemented in Java and C++.
- However, the differences stems from the fact that in C++ there is no abstract identifier to make a member function declared as abstract-member-function.
- Instead, C++ allows declaring a member function that its implementation is not necessary, to be declared as **pure virtual function**:
  - **Java:**

    public abstract double area();
  - **C++:**

    virtual double area() = 0;
- Note: The C++ version of the pervious implementation of polymorphism in Java will be discussed during the lectures.