*** Is-a = Inheritance**

Your class is a subclass of a more generalized class, e.g. HouseCat inherits from Feline because a house cat "is a" feline.

*** Has-a = Composition**

A LittleGirl has a cat, so obviously she should not be a subclass to HouseCat (she is not a cat). It is best that she "has a" HouseCat member field.

```
public class LittleGirl {
    private int age;
    private String name;
    private HouseCat pet;
}
```

*** Uses-a = Interface**

Interfaces should be used when a class or group of classes all have similar functionality, but when there is no obvious line of inheritance. Think of them as a certificate that says "this object performs this functionality."

For example, a HouseCat might inherit from Feline, but implement the ICanHasChesseburgers (or ICanHazChzbrgrsPlz) interface. That way you have a BurgerJoint class with a method public CheeseBurger Serve(ICanHasCHeeseburgers patron) and be able to pass either Humans or HouseCats to the Serve method in order to feed them a Cheeseburger.

This is useful because HouseCat does not inherit from Person nor vice versa. However, they both perform acts involving CheeseBurgers.