# Inheritance

*Children and grandchildren*

Design Patterns for Web Programming
Web Design & Development Bachelor of Science Degree

## Inheritance

*From General to Specific*

- Inheritance is to base an object off of another object
- The class that is the base, which other classes will inherit from is called the **super class**
- The class that is inheriting is the **sub class**

FULL SAIL
UNIVERSITY

## Inheritance

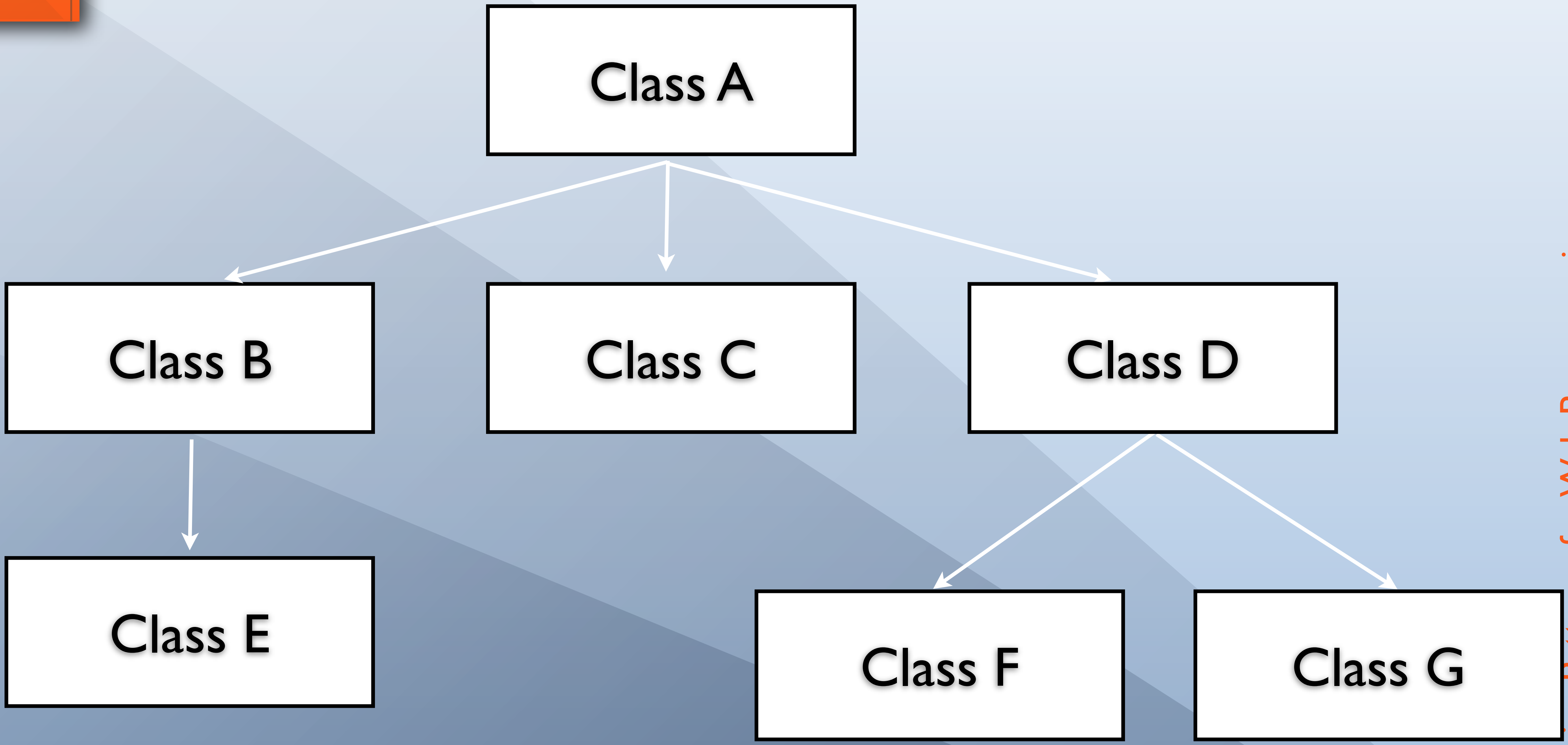
*From General to Specific*

- Subclasses will *inherit* methods, properties and attributes of the superclass
- Think of the superclass as a "template"
- Go from general to specific

Design Patterns for Web Programming
Web Design & Development Bachelor of Science Degree

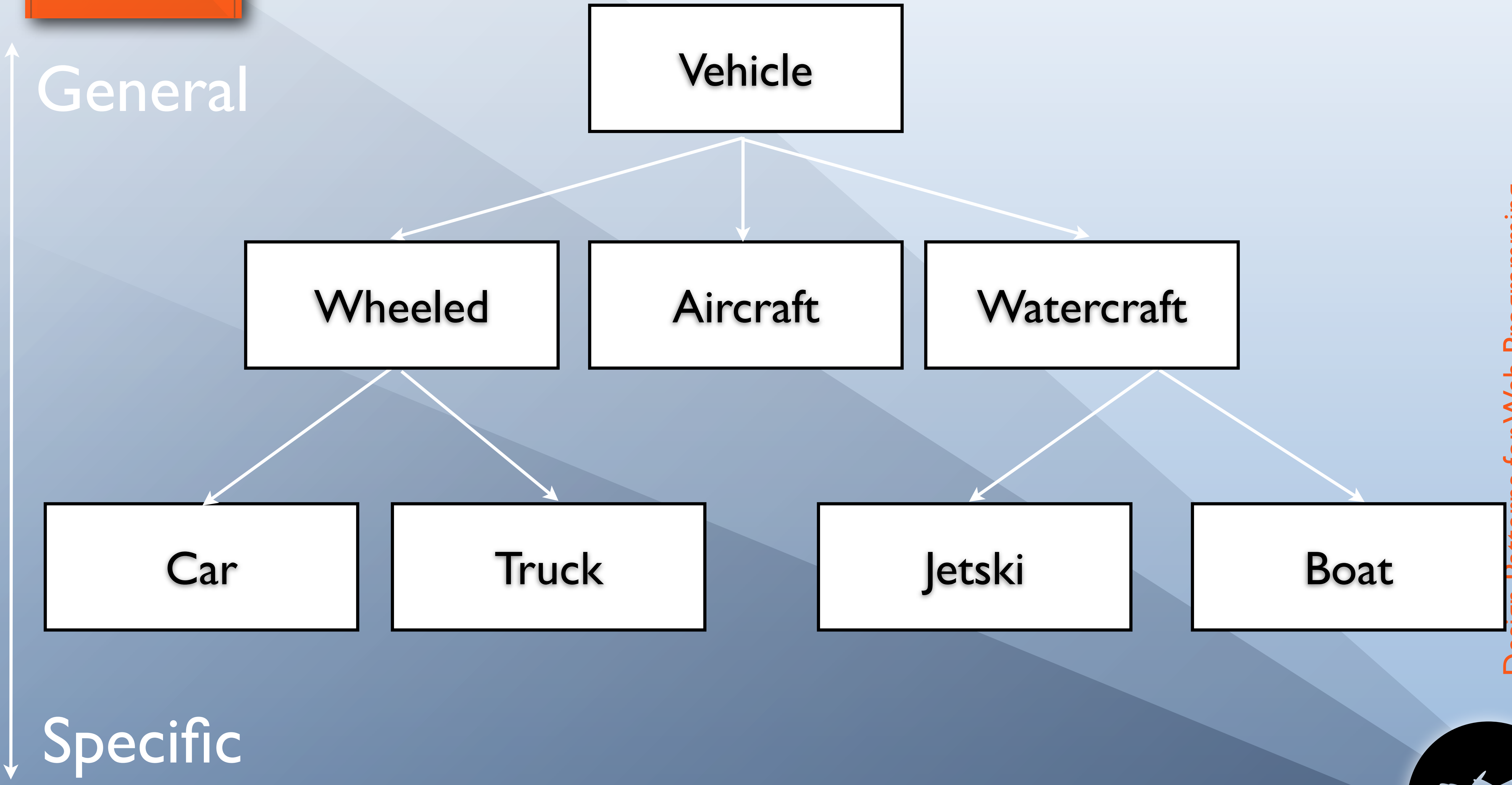
# Inheritance

*Abstract Example*


$$\begin{tikzcd}
 & \boxed{\text{Class A}} \arrow[dl] \arrow[d] \arrow[dr] & \\
\boxed{\text{Class B}} \arrow[d] & \boxed{\text{Class C}} & \boxed{\text{Class D}} \arrow[d] \arrow[dr] & \\
\boxed{\text{Class E}} & & \boxed{\text{Class F}} & \boxed{\text{Class G}}
\end{tikzcd}$$

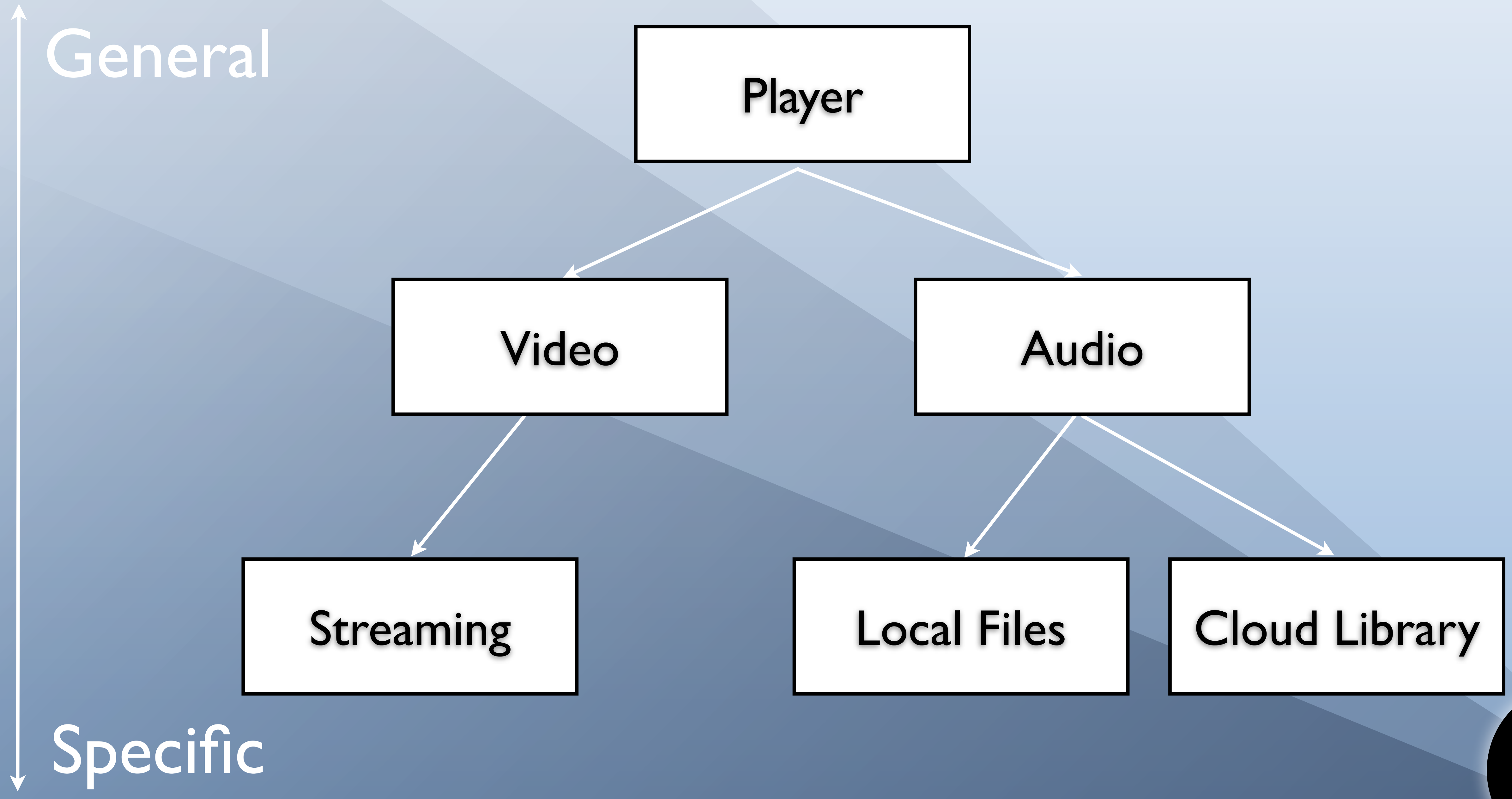
# Inheritance

*Real world analogy*



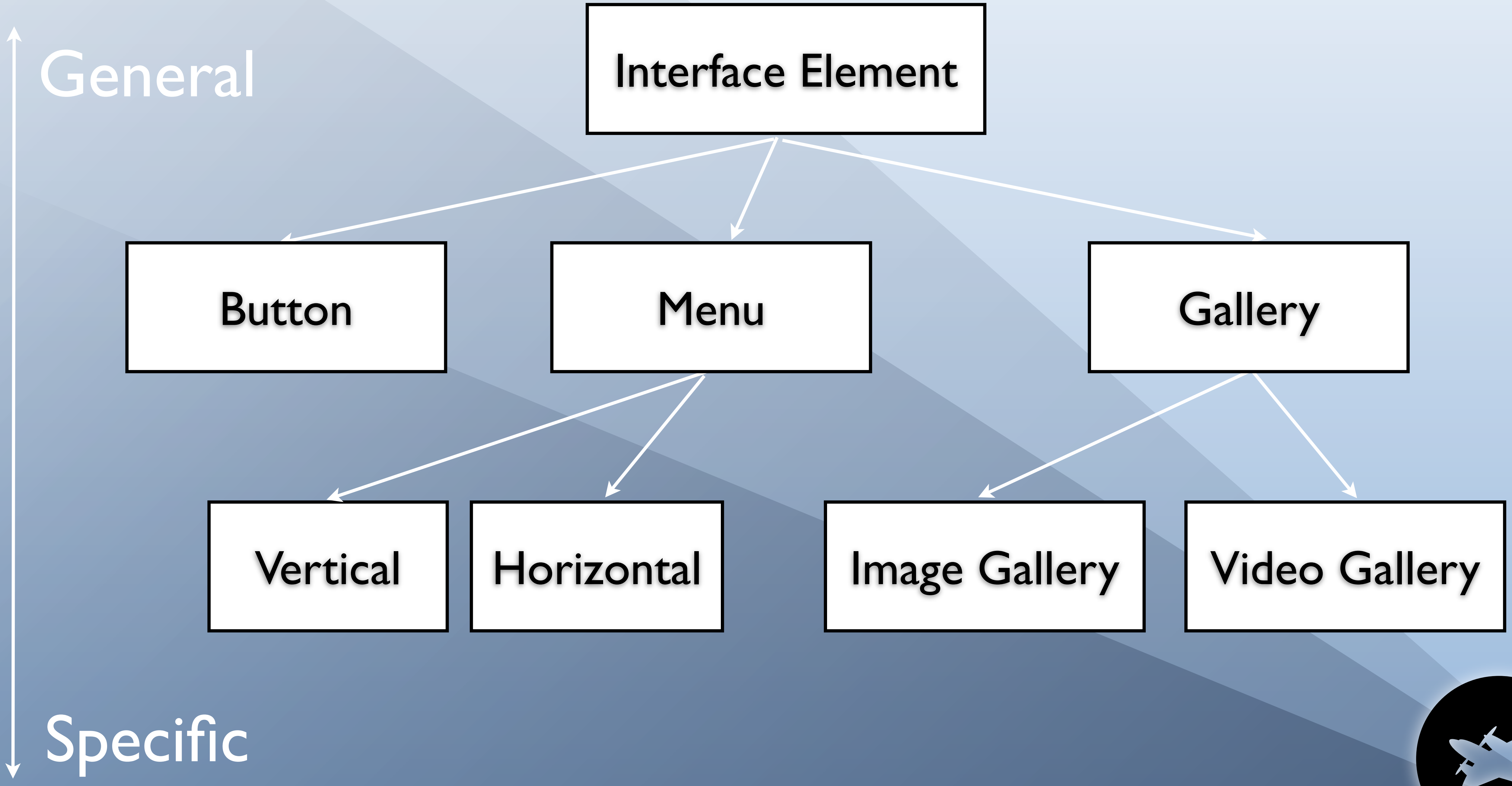
# Inheritance

*Practical Example*



# Inheritance

*Practical Example*



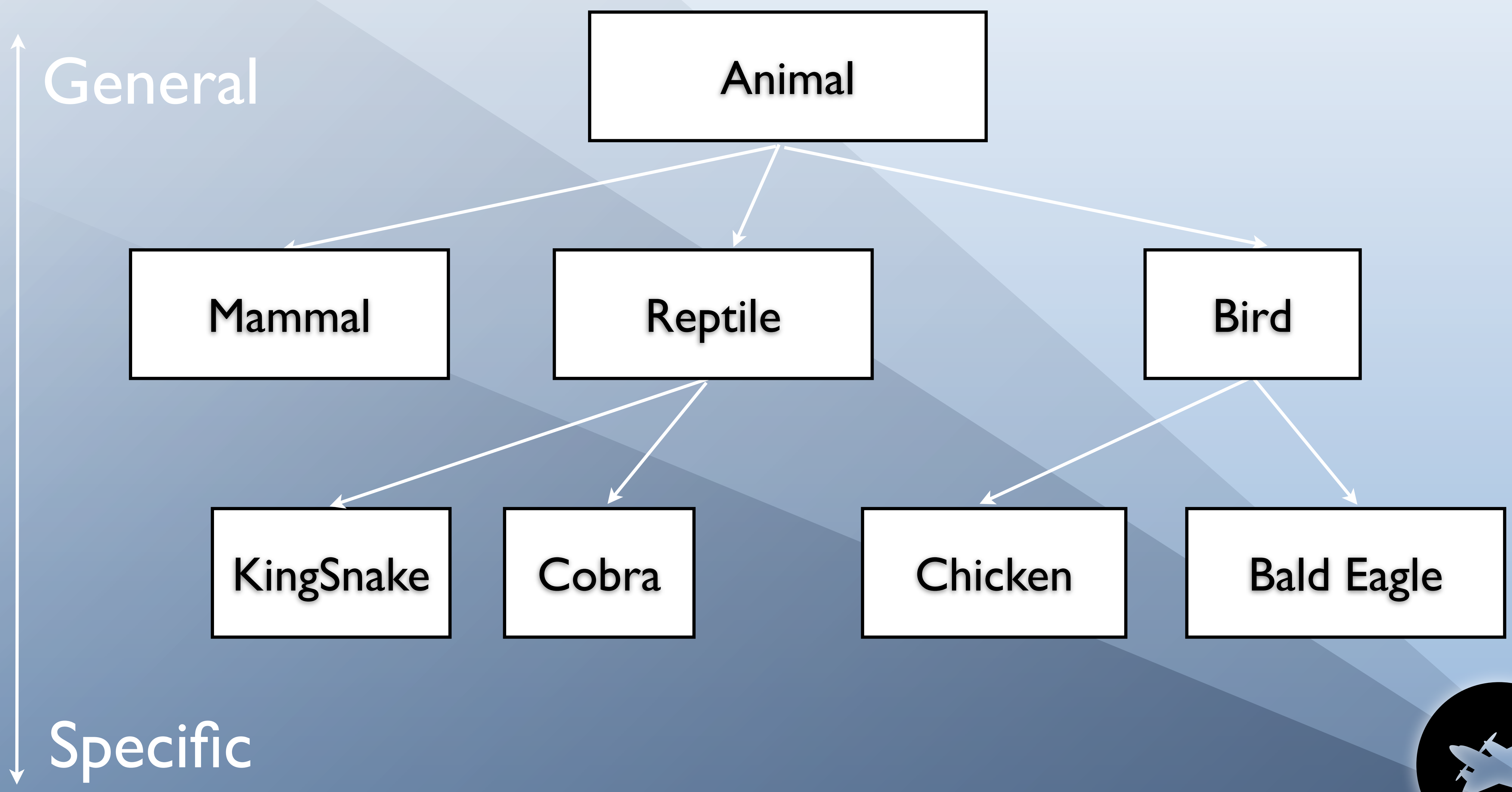
## Litmus Test

*How to know you are doing inheritance right*

- ______ IS ALSO A(N) ________
- subclass goes in first blank
- superclass goes in second blank
- Great way to know you are doing it right!

# Inheritance

*Lions, Tigers and Bears, Oh my!*



# Inheritance in Python

*So, that's what that "object" thing was for!*

```
class Button(object):
    def __init__(self):
        self.__label = "Submit"
        self._user_name = “”

    @property
    def label(self):
        return self.__label

class EmailButton(Button):
    def __init__(self):
        super(EmailButton, self).__init__()
        self.__email = “kermit@muppet.com”
        self._user_name = “Kermit4Ever”
```

# Inheritance in Python

*Access parent properties, methods and attributes!*

```
class Button(object):
    def __init__(self):
        self.__label = "Submit"
        self._user_name = “”

    @property
    def label(self):
        return self.__label

class EmailButton(Button):
    def __init__(self):
        super(EmailButton, self).__init__()
        self.__email = “kermit@muppet.com”
        self._user_name = “Kermit4Ever”
```

## Inheritance in Python

*Invoking the Superclass's constructor function*

```
class Button(object):
    def __init__(self):
        self.__label = "Submit"
        self._user_name = “”

    @property
    def label(self):
        return self.__label

class EmailButton(Button):
    def __init__(self):
        Button.__init__()
        self.__email = “kermit@muppet.com”
        self._user_name = “Kermit4Ever”
```

# Inheritance in Python

*Invoking the Superclass's constructor function*

```
class Button(object):
    def __init__(self):
        self.__label = "Submit"
        self._user_name = “”

    @property
    def label(self):
        return self.__label

class EmailButton(Button):
    def __init__(self):
        super(EmailButton, self).__init__()
        self.__email = “kermit@muppet.com”
        self._user_name = “Kermit4Ever”
```

FULL SAIL
UNIVERSITY

## Inheritance Review

*So what was all that now?*

- Inheritance allows us to use classes as templates for other classes
- ______ IS A(N) ______ litmus test
- In the code:
  - Superclass within parenthesis
  - Invoke superclass's constructor

Design Patterns for Web Programming
Web Design & Development Bachelor of Science Degree