COMP10020 Introduction to Programming II **Combining Objects 1**

Dr. Brian Mac Namee
brian.macnamee@ucd.ie
School of Computer Science
University College Dublin

References

Some of the material in this lecture is based on "Gentle Object Oriented Programming in Python by Nicholas H.Tollervey & Naomi Ceder

http://ntoll.org/static/presentations/oopy/index.html

We Talked About Cows!

Back To Language

Class = ?

Object = ?

Method = ?

Attribute = ?

(This is not entirely accurate but is close enough for today)

Back To Language

Class = Noun

Object = Proper noun

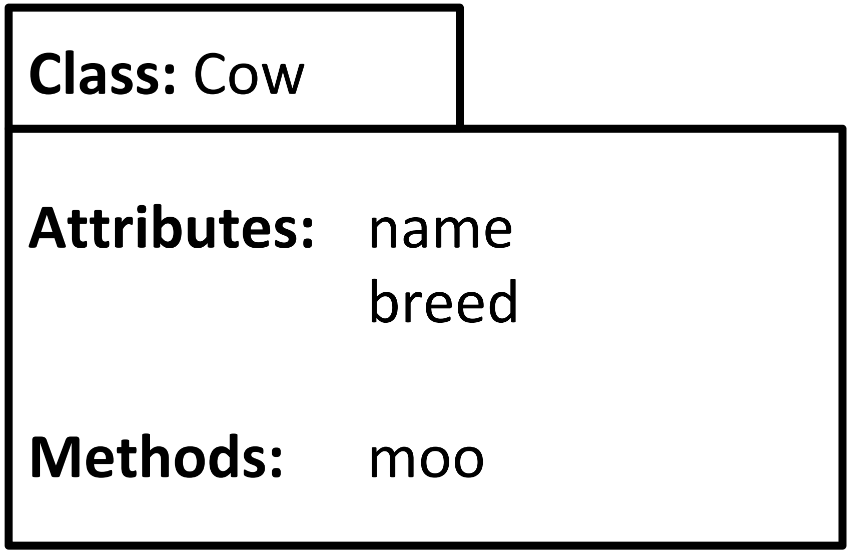
Method = Verb

Attribute = Adjective

(This is not entirely accurate but is close enough for today)

Defining a Class

The first step in defining a class is to define its attributes and methods



Then we can write Python code to define it

Defining a Class

The first step in defining a class is to define its attributes and methods

Class: Cow

ABSTRACTION

Methods: moo

Then we can write Python code to define it

A Simple Pythonic Cow

```
class Cow:

    def __init__(self, name, breed):
        self.name = name
        self.breed = breed

    def moo(self, message):
        print self.name +  " says " + message
```

A Simple Pythonic Cow

```
class Cow:

    def __init__(self, name, breed):
        self.name = name
        self.breed = breed

    def moo(self, message):
        print self.name +  " says " + message
```

This is a special function.
What does it do?

A Simple Pythonic Cow

```
class Cow:

    def __init__(self, name, breed):
        self.name = name
        self.breed = breed

    def moo(self, message):
        print self.name +  " says " + message
```

What are these?

A Simple Pythonic Cow

```
class Cow:

    def __init__(self, name, breed):
        self.name = name
        self.breed = breed

    def moo(self, message):
        print self.name +  " says " + message
```

What type of function is this?

A Simple Pythonic Cow

```
myFirstCow = Cow("Daisy", "Friesan")
myFirstCow.moo("Hello")

mySecondCow = Cow("Buttercup", "Belgian Blue")
mySecondCow.moo("Moooooooo!")
```

Pythonic Cards

```
# The card class
class Card:

    # A constructor called when an object of
     # the class is instantiated.
    def __init__(self, suit, face):
        self.suit = suit
        self.face = face

    # A class method that prints a card
    def show(self):
        print(self.face + " of " + self.suit)
```

Pythonic Cards

```
myCard1 = Card('Hearts','A')
myCard1.show()

myCard2 = Card('Diamonds','K')
myCard2.show()
```

Key Ideas in OOP

In diving deeper into OO programming we will look at the four major principles:

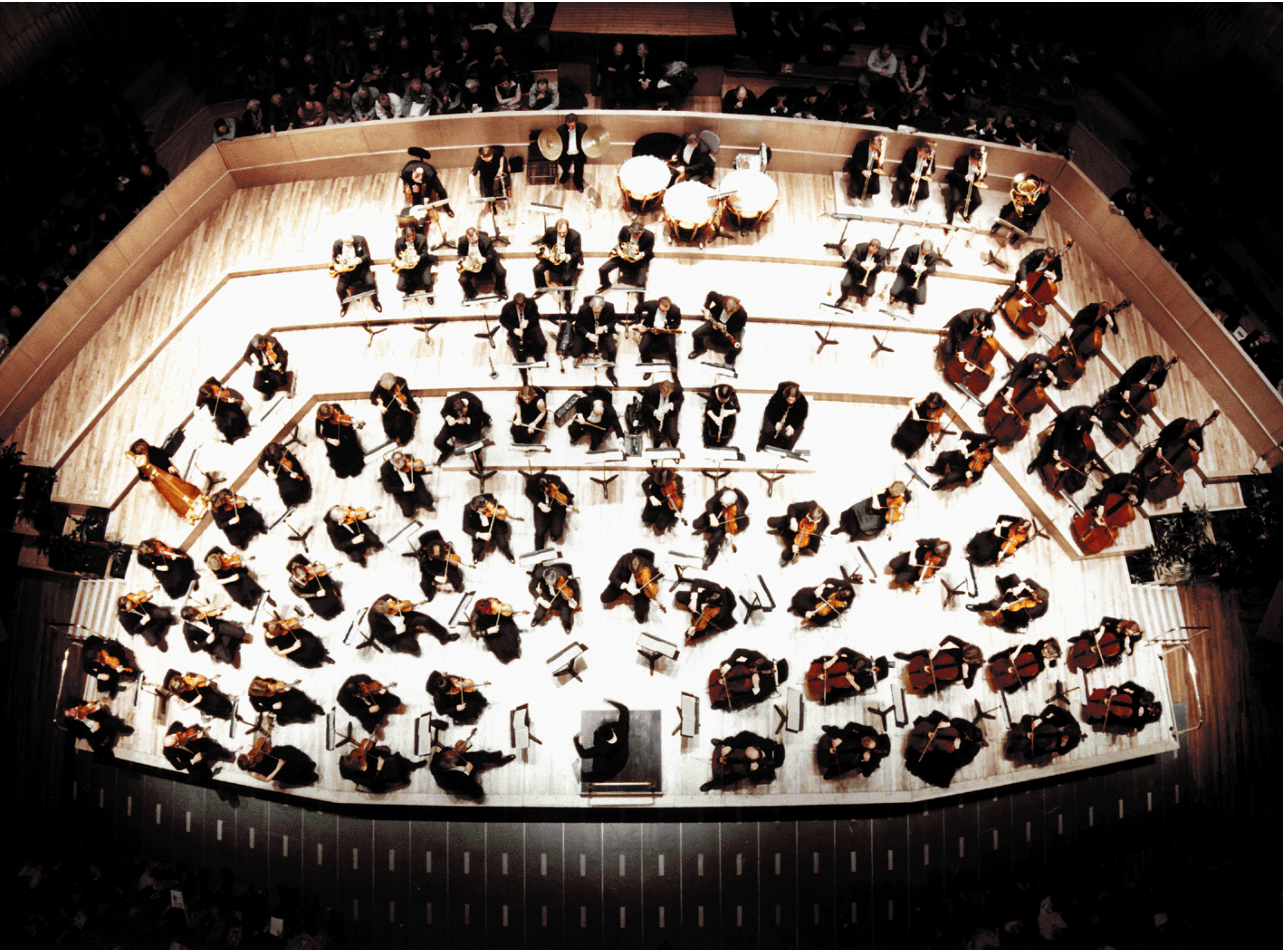
- **Encapsulation**
- Composition
- Inheritance
- Polymorphism

COMPOSITION

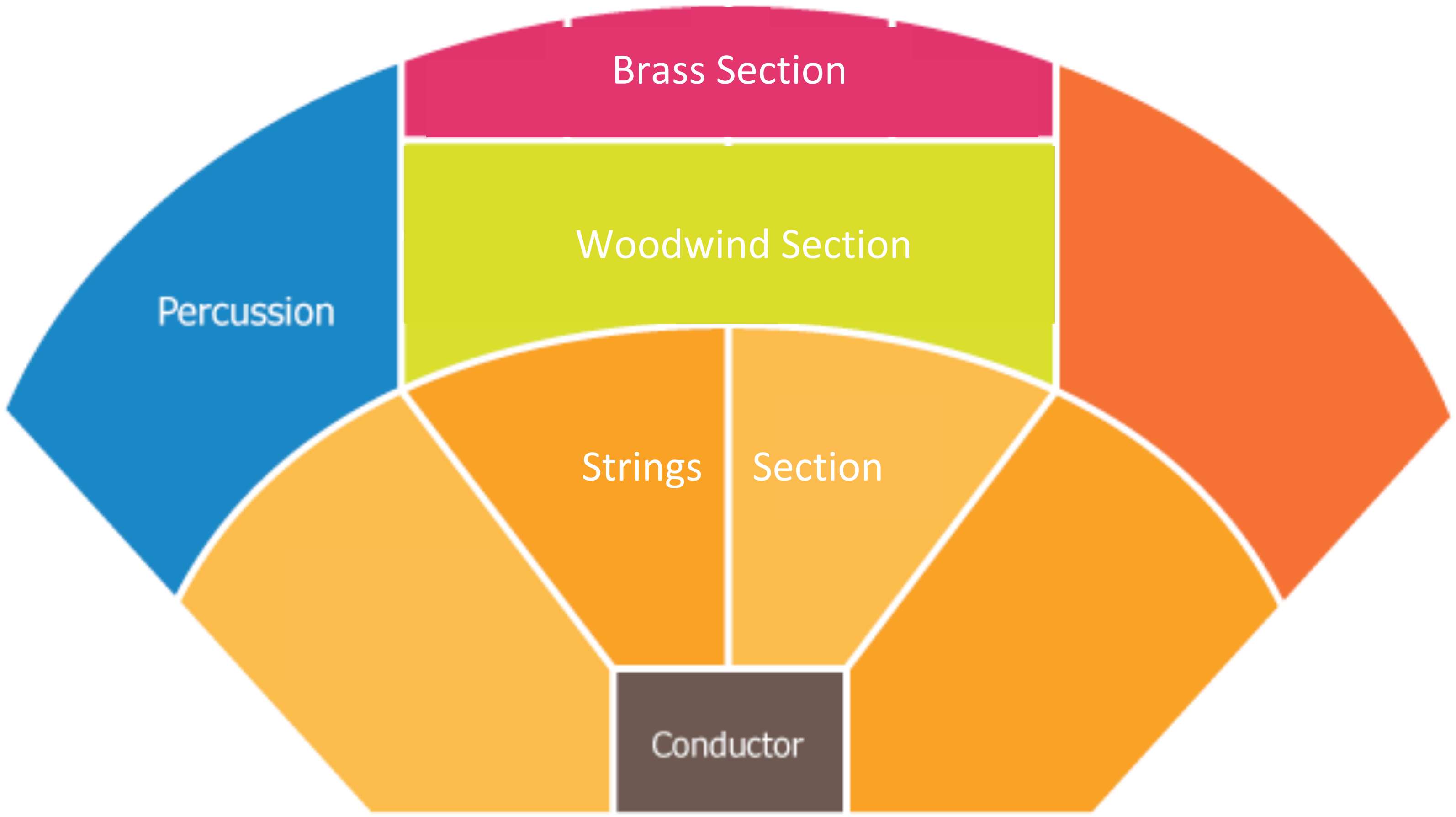
Key Ideas in OOP

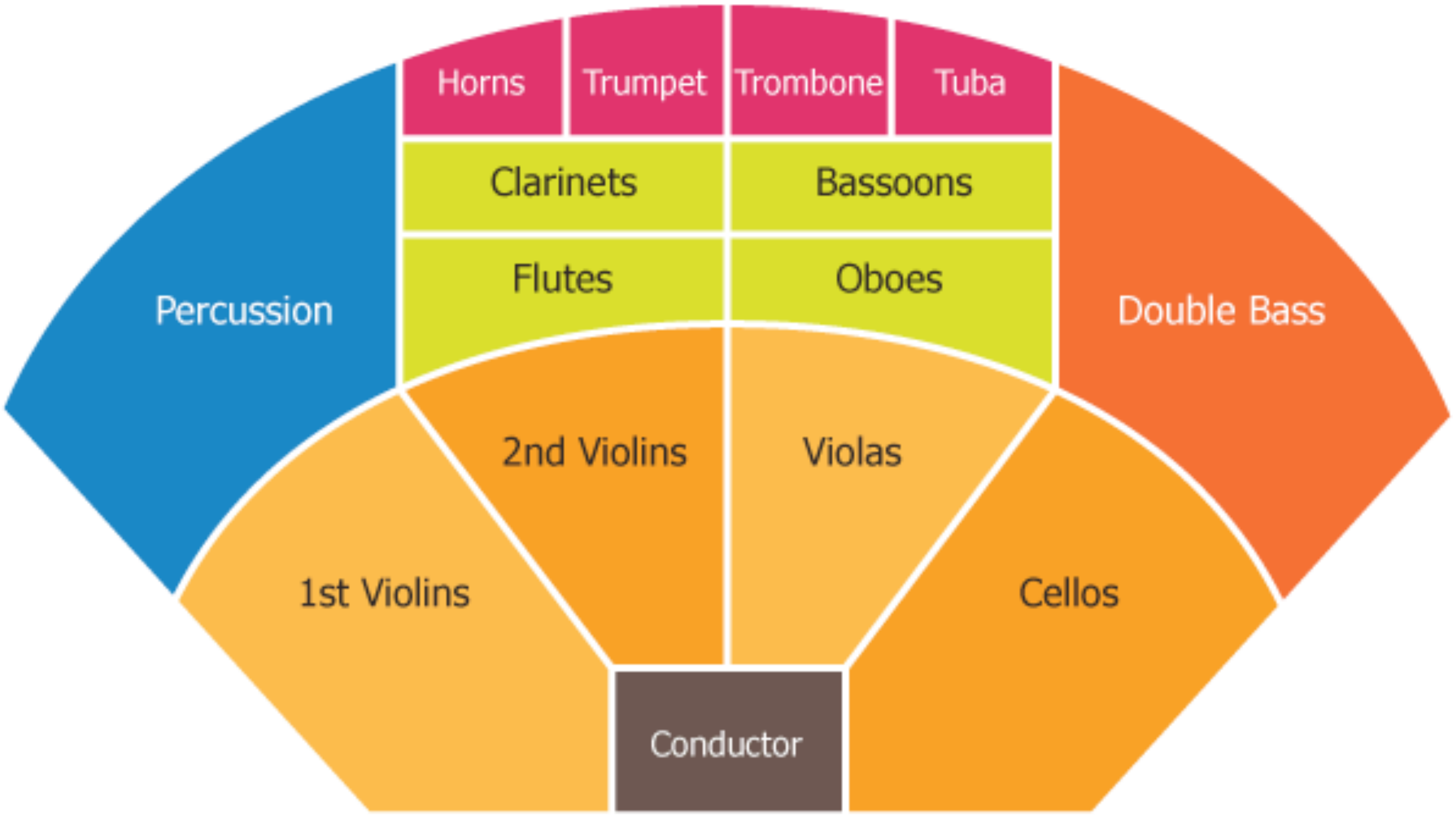
In diving deeper into OO programming we will look at the four major principles:

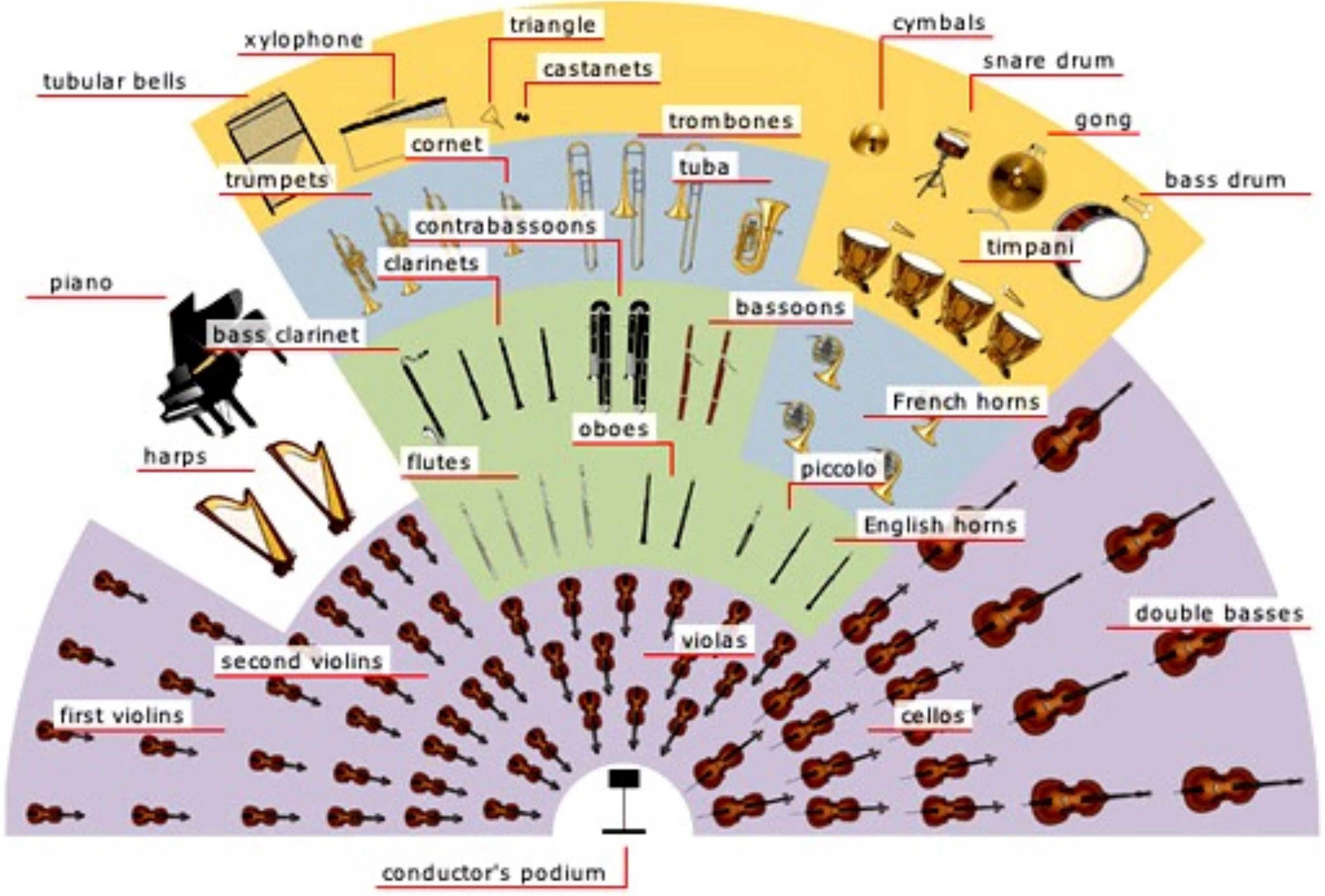
- Encapsulation
- **Composition**
- Inheritance
- Polymorphism



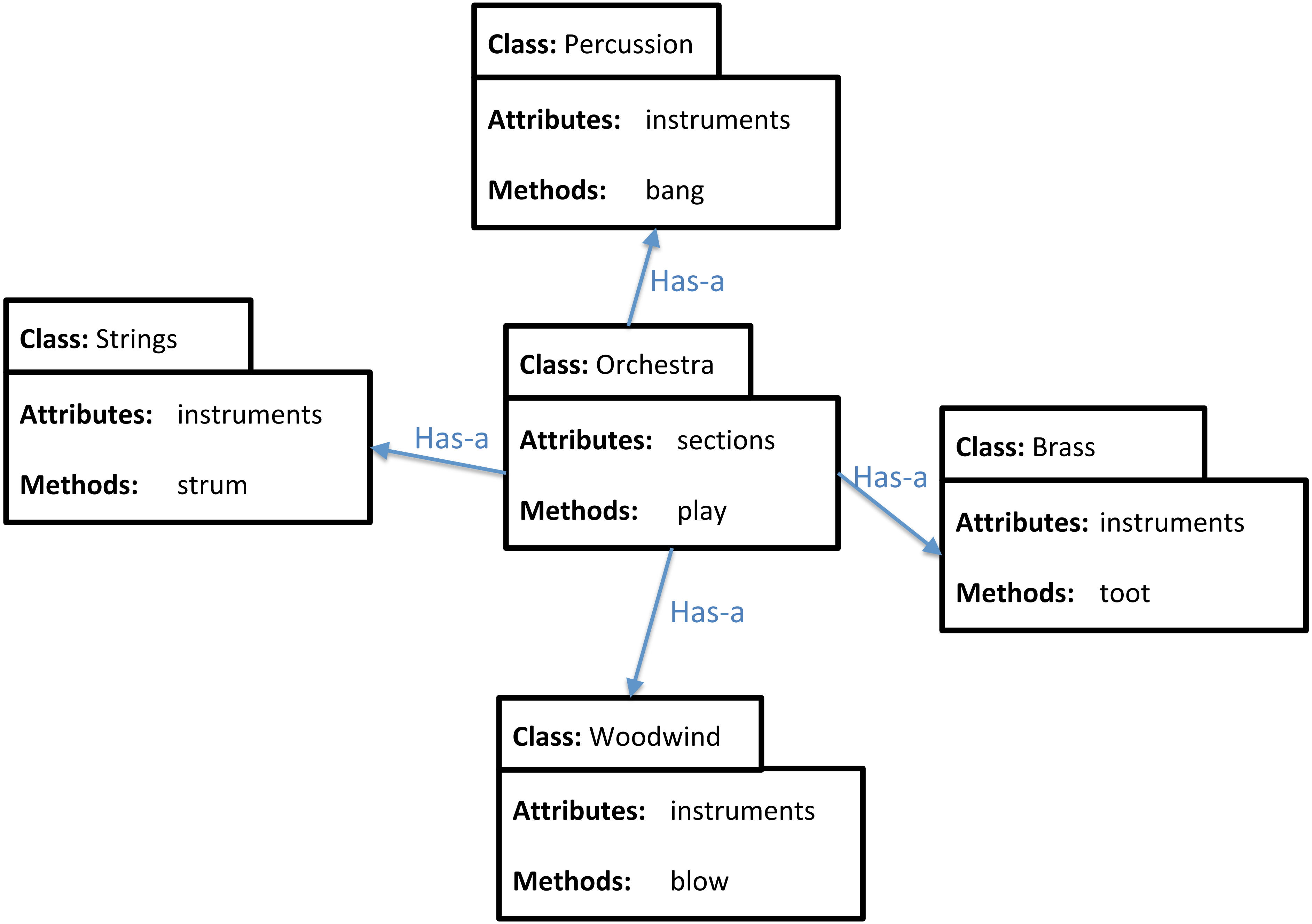
Orchestra

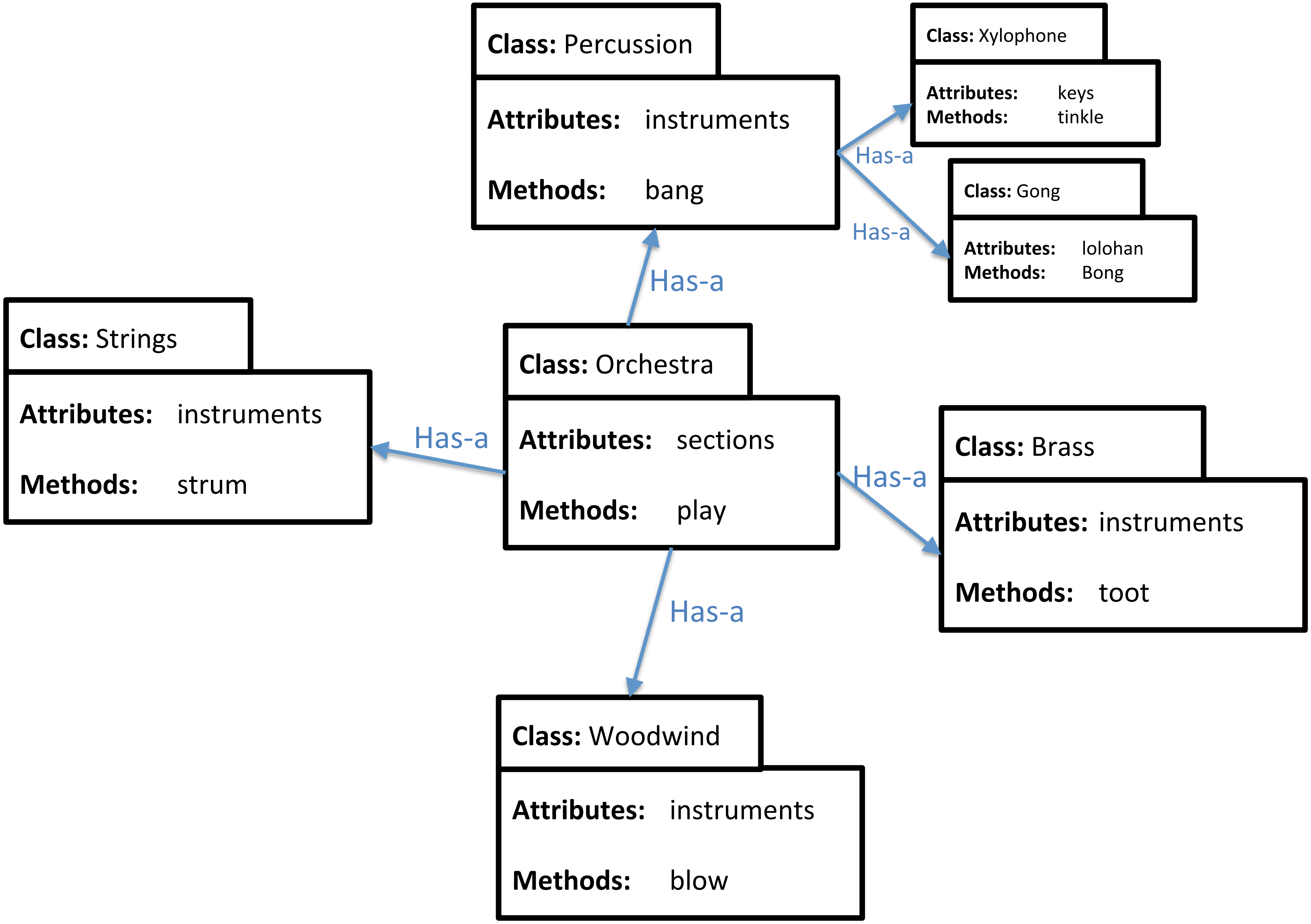

Brass Section
Woodwind Section
Percussion
Strings
Section
Conductor


Horns
Trumpet
Trombone
Tuba
Clarinets
Bassoons
Flutes
Oboes
Percussion
Double Bass
2nd Violins
Violas
1st Violins
Cellos
Conductor

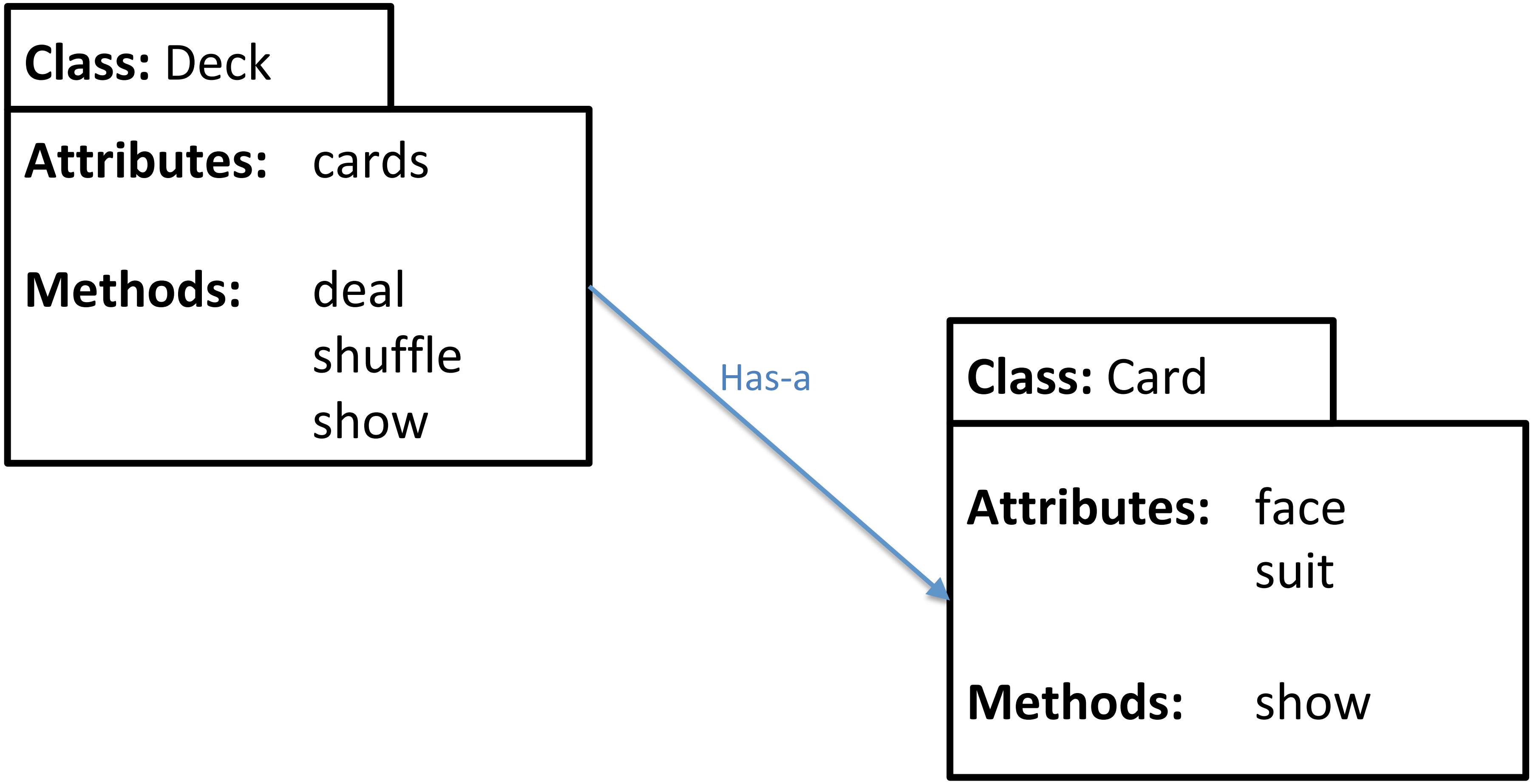


🟩	🟦	🟧	🟪
woodwind family	brass family	percussion instruments	violin family


Class: Percussion
Attributes: instruments
Methods: bang
Has-a
Class: Strings
Attributes: instruments
Methods: strum
Has-a
Class: Orchestra
Attributes: sections
Methods: play
Has-a
Class: Brass
Attributes: instruments
Methods: toot
Has-a
Class: Woodwind
Attributes: instruments
Methods: blow


Class: Percussion
Attributes: instruments
Methods: bang
Class: Xylophone
Attributes: keys
Methods: tinkle
Class: Gong
Attributes: lolohan
Methods: Bong
Has-a
Has-a
Has-a
Class: Strings
Attributes: instruments
Methods: strum
Has-a
Class: Orchestra
Attributes: sections
Methods: play
Has-a
Class: Brass
Attributes: instruments
Methods: toot
Has-a
Class: Woodwind
Attributes: instruments
Methods: blow

A simpler Composition Example



SUMMARY

Summary

In diving deeper into OO programming we will look at the four major principles:

- – Encapsulation
- – **Composition** <- looked at today
- – Inheritance
- – Polymorphism