# Polymorphism

Damian Gordon

# Polymorphism

- *Polymorphism* simply means that we can call the same method name with parameters, and depending on the parameters, it will do different things. For example:

```
>>> print(6 * 5)

>>> print(“Hello” * 5)
```

# Polymorphism

- *Polymorphism* simply means that we can call the same method name with parameters, and depending on the parameters, it will do different things. For example:

```
>>> print Mult (6, 5)
```

```
>>> print Mult (“Hello”, 5)
```

# Polymorphism

- *Polymorphism* simply means that we can call the same method name with parameters, and depending on the parameters, it will do different things. For example:

```
>>> print Mult (6, 5)
30
>>> print Mult (“Hello”, 5)
HelloHelloHelloHelloHello
```

# Polymorphism

- A more complicated example to consider would be to think about creating a method called `play()` to play an audio file.
- A media player will be needed to load the `AudioFile` object.
- The instruction to play the file might be as simple as:

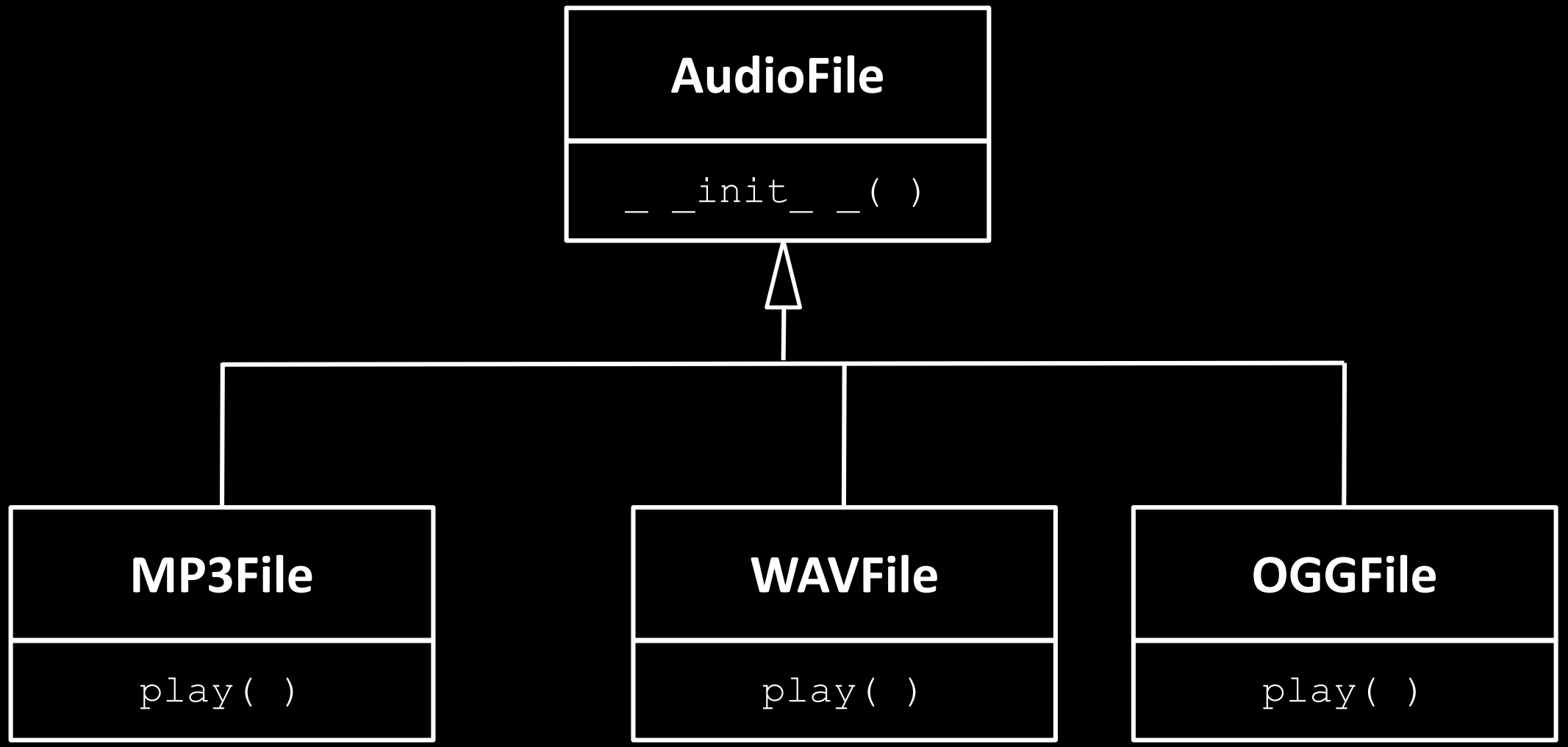
```
>>>  audio_file.play( )
```

# Polymorphism

- However, different audio files use different compression algorithms (e.g. .mp3, .wma, .ogg), and some aren't stored as compressed at all (e.g. .wav).

- We can use inheritance with polymorphism to simplify the design. Each filetype is represented as a different subclass of `AudioFile`, and each of those has a `play( )` method.

# Polymorphism



# Polymorphism

```
class AudioFile:

    def __init__(self, filename):
        if not filename.endswith(self.ext):
            # THEN
            raise Exception("Invalid format")
        # ENDIF;
        self.filename = filename
    # END init()

# END CLASS.
```

# Polymorphism

```
class AudioFile:

    def __init__(self, filename):
        if not filename.endswith(self.ext):
            # THEN
            raise Exception("Invalid format")
        # ENDIF;
        self.filename = filename
    # END init()

# END CLASS.
```

Check if the file extension of the audio being played is a known extension, `self.ext` is set in each of the subclasses.

Raise an exception if it's an unknown file extension

If it's a known file extension, then assign the filename passed in to `self.filename`

# Polymorphism

```
class MP3File(AudioFile):

    ext = “mp3”

    def play(self):
        print(“playing {} as mp3”.format(self.filename))
    # END play

# END CLASS.
```

# Polymorphism

```
class WAVFile(AudioFile):

    ext = “wav”

    def play(self):
        print(“playing {} as wav”.format(self.filename))
    # END play

# END CLASS.
```

# Polymorphism

```
class OGGFile(AudioFile):

    ext = “ogg”

    def play(self):
        print(“playing {} as ogg”.format(self.filename))
    # END play

# END CLASS.
```

# Polymorphism

- Here's how we run it:

```
>>> mp3 = MP3File("myfile.mp3")
>>> mp3.play()
playing myfile.mp3 as mp3
```

# Polymorphism

- Here's another one:

```
>>> wav = WAVFile("myfile.wav")
>>> wav.play()
playing myfile.wav as wav
```

# Polymorphism

- Here's an error:

```
>>> ogg_declared_as_mp3 = MP3File("myfile.ogg")
```

```
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
File "polymorphic_audio.py", line 4, in __init__
raise Exception("Invalid format")

Exception: Invalid format
```


TO BE
CONTINUED...