How to Write *Actually* Object Oriented Python

Per Fagrell » mango@spotify.com » perfa (github)

Who am I?

OO vs Procedural

Design Principles

Testing

Wrap-up & questions

Who am I?

Per Fagrell

Spotify®

MEDIATEK

ERICSSON

ENEA

Assembly

C

C++

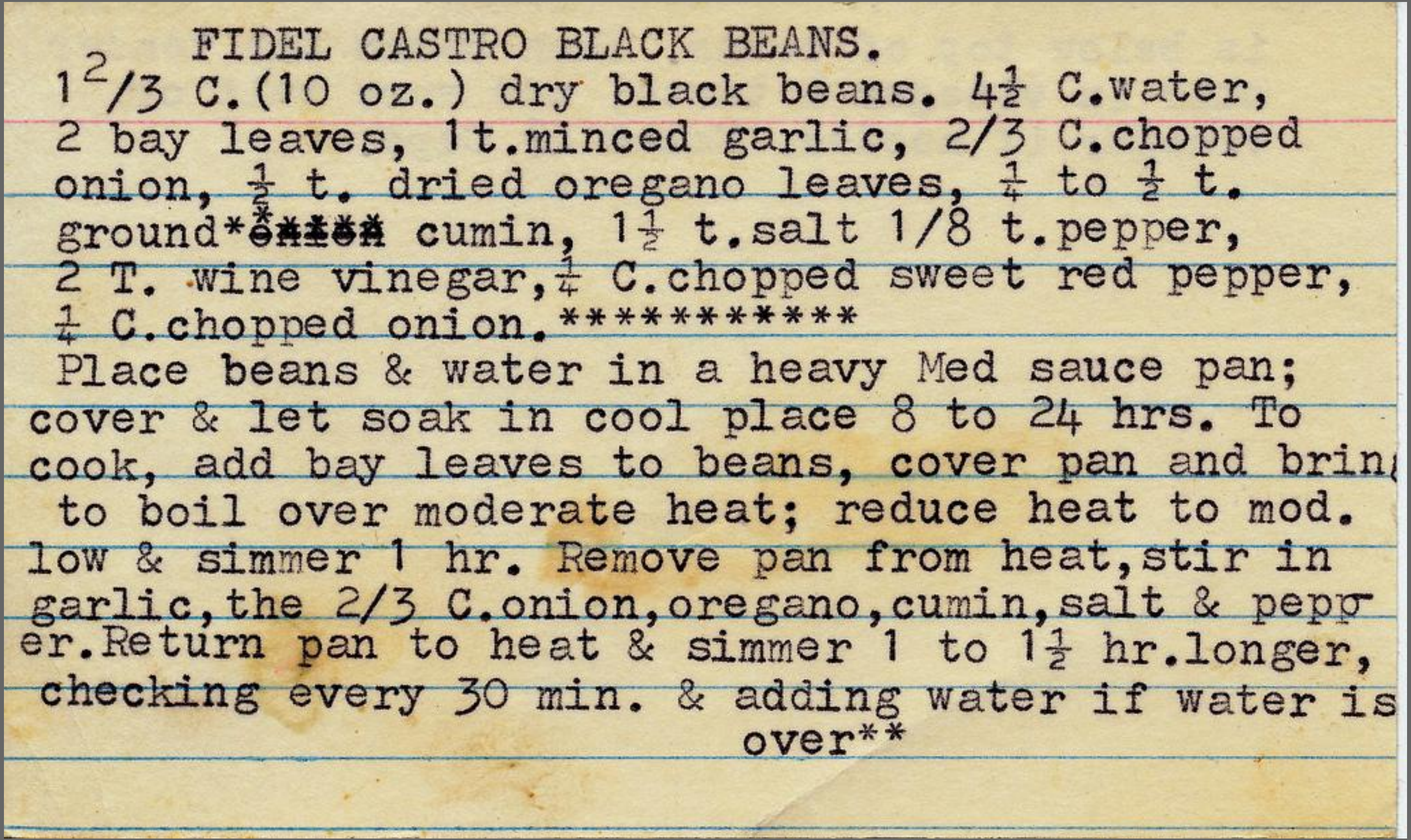
Java

Python

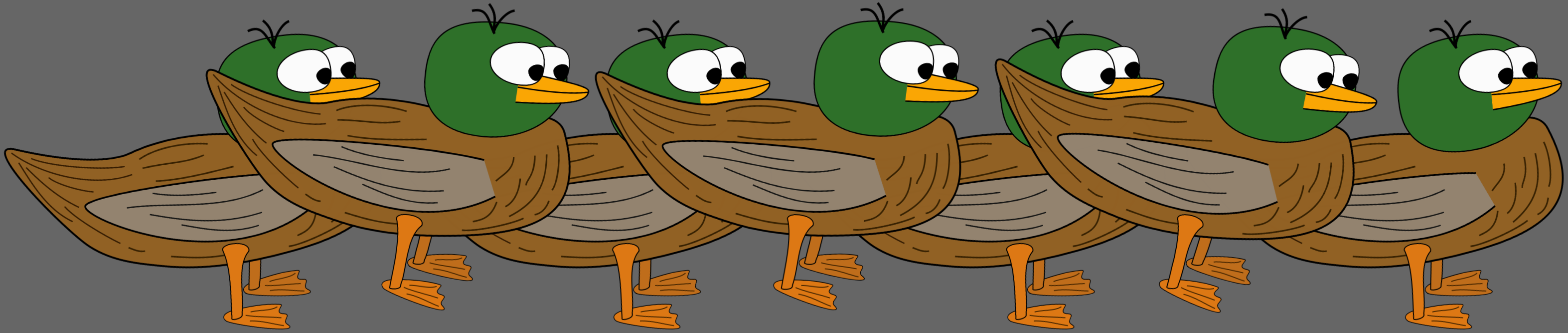
Procedural

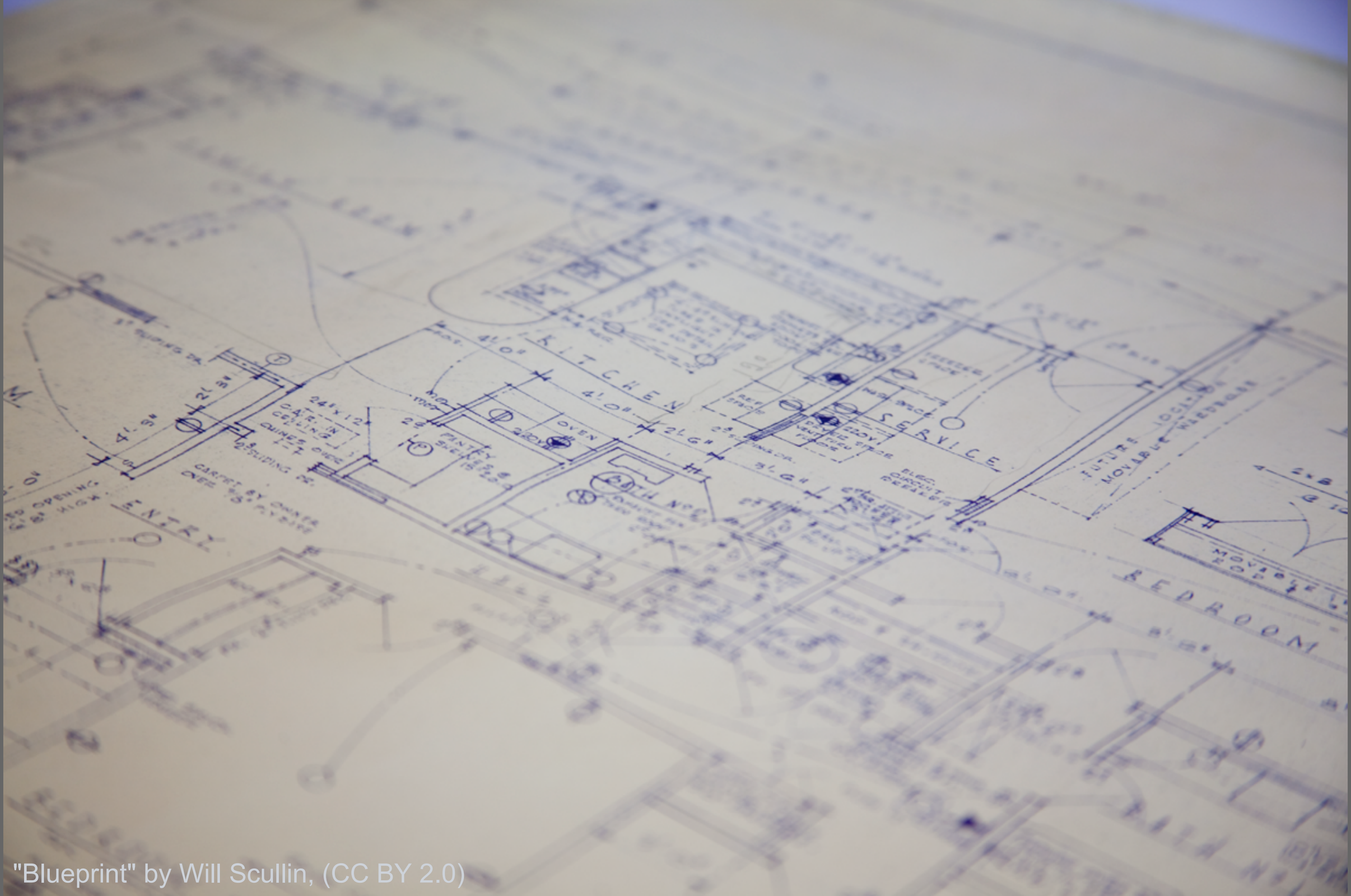
vs

Object-oriented



FIDEL CASTRO BLACK BEANS.
1^{2}/3 C.(10 oz.) dry black beans. 4½ C.water,
2 bay leaves, 1t.minced garlic, 2/3 C.chopped
onion, ½ t. dried oregano leaves, ¼ to ½ t.
ground*~~onion~~ cumin, 1½ t.salt 1/8 t.pepper,
2 T. wine vinegar,¼ C.chopped sweet red pepper,
¼ C.chopped onion.***********
Place beans & water in a heavy Med sauce pan;
cover & let soak in cool place 8 to 24 hrs. To
cook, add bay leaves to beans, cover pan and brin
to boil over moderate heat; reduce heat to mod.
low & simmer 1 hr. Remove pan from heat,stir in
garlic,the 2/3 C.onion,oregano,cumin,salt & pepp-
er.Return pan to heat & simmer 1 to 1½ hr.longer,
checking every 30 min. & adding water if water is
over**





"Blueprint" by Will Scullin, (CC BY 2.0)

"Crowd - Fres Festival 2013" by FresFestival, (CC BY 2.0)

Maps your
mental model
to code

Python

Gives

GREAT

Freedoms

One-off Scripts

Frameworks & Libraries

Servers

Games & applications

```
def func1(...):
    class Helper(object):
        def method1(...):
            def help_func(...):
                if condition:
                    for x in collection:
```

```
class Builder(object):
        def calc_max_coeff(self, x, y):
                h = [i*i for i in x]
                m= [j*j for j in x]
                p1 = [(i,j) for i,j in zip(h[:2:], m)
                try:
                        f = open(y, ‘r’)
                except IOError:
                        return {}
                coeffs = []
                for line in f.readlines():
                        v = int(line)
                        q = h[v]+m[v]
                        s,t = p1[0] * v + p1[1] * v
                        coeffs.append((s, t))
                o = open(result_path, ‘w’):
                for c in coeffs:
                        x, y = c
                        o.write(“x=%s:y=%s;”)
                self.coeff_max = max(coeffs)

        def compare_max_coeff(q):
                ….
```

Consistency

$$\begin{tikzcd}\text{Consistency} \arrow[d] \\ \text{Discipline}\end{tikzcd}$$

“Doorman, Upper East Side, Manhattan” Jeffrey Zeldman **CC BY 2.0**

“Steampunk Robot” Sougent Harrop **CC BY-SA 2.0**

Consistency improves maintainability

Consistency simplifies testing

Consistency

Simplifies

Communication

Dry

Don’t Repeat Yourself

```
value = remap((input + self.old_value) * 2.3)
if value < threshold:
    raise InputRangeError("Input out of range, adjusted input: %f",
                          (input + self.old_value) * 2.3)
log.info("Setting adjusted by %f", (input + self.old_value) * 2.3)
```

```
adjusted = (input + self.old_value) * ADJUSTMENT_COEFF
value = remap(adjusted)
if value < threshold:
    raise InputRangeError("Input out of range, adjusted input: %f",
                          adjusted)
log.info("Setting adjusted by %f", adjusted)
```

```
def load(self):
    with open(BASE_SETTINGS, 'r') as settings:
        try:
            load_base_settings(settings)
        except LoadError:
            log.error(“Failed to load %s”, BASE_SETTINGS)
    with open(PLUGIN_SETTINGS, 'r') as settings:
        try:
            load_plugin_settings(settings)
        except LoadError:
             log.error(“Failed to load %s”, PLUGIN_SETTINGS)
    with open(EXTENSION_SETTINGS, 'r') as settings:
        …
```

```python
def load(self):
    try_to_load(BASE_SETTINGS, load_base_settings)
    try_to_load(PLUGIN_SETTINGS, load_plugin_settings)
    try_to_load(EXTENSION_SETTINGS, load_extension_settings)
```

```python
CONFIG_LOAD_MAP = [(BASE_SETTINGS, load_base_settings),
                   (PLUGIN_SETTINGS, load_plugin_settings),
                   (EXTENSION_SETTINGS, load_extension_settings)]

def load(self):
    for settings_file, loader in CONFIG_LOAD_MAP:
        try_to_load(settings_file, loader)
```

```python
def test_should_do_x(self):
    …
    self.assertEqual(user, testobject.user)
    self.assertEqual(project, testobject.project)
    self.assertEqual(owner, testobject.owner)

def test_should_do_y(self):
    …
    self.assertEqual(user, testobject.user)
    self.assertEqual(project, testobject.project)
    self.assertEqual(owner, testobject.owner)
```

```python
def test_should_do_x(self):
    …
    self.assertValidTestobject(testobject)

def test_should_do_y(self):
    …
    self.assertValidTestobject(testobject)
```

Don’t Repeat Yourself

Solid

Single Responsibility Principle

Code should only have ONE reason to change

“Be Responsible” Sam Howzit (CC BY 2.0)

```
class Modem(object):
      def call(self, number):
            …
      def disconnect(self):
            …
      def send_data(self, data):
            …
      def recv_data(self):
            …
```

```
class ConnectionManager(object):
    def call(self, number):
        …
    def disconnect(self):
        …
class DataTransciever(object):
    def send_data(self, data):
        …
    def recv_data(self):
        …
```

```python
class Person(object):
     def calculate_pay(self):
         …
     def save(self):
         …
```

```python
class Person(object):
     def calculate_pay(self):
         …

class Persistor(object):
     def save(self, person):
         …
```

```
class Person(object, DbPersistMixin):
    def calculate_pay(self):
        …

class DbPersistMixin(object):
    def save(self):
        …
```

```python
def process_frame(self):
    frame = self.input_processor.top()

    start_addr = frame.addr
    pow2_size = 1
    while pow2_size < frame.offs:
        pow2_size <<= 1
    end_addr = start + pow2_size
    o_map = io_map.new_map(start_addr, end_addr)

    self.output_processor.flush(o_map)
```

```python
def process_frame(self):
    frame = self.input_processor.top()
    o_map = self.memory_mapper.map(frame)
    self.output_processor.flush(o_map)
```

Single Responsibility Principle

Open/Closed Principle

Code should open to extension but closed to modification

“Craftsman” A. Davey (CC BY 2.0)

How can you know?

```
def validate_link(self, links):
    for link in links:
        track = Track(link)
        self.validate(track)
```

```python
def validate_link(self, links):
    for link in links:
        if link.startswith("spotify:album:"):
            uri = Album(link)
        else:
            uri = Track(link)
        self.validate(uri)
```

```python
def validate_link(self, links):
    for link in links:
        self.validate(uri_factory(link))
```

Song Album Playlist UserDefinedRadio

Open/Closed Principle

Liskov Substitutability Principle

“Duck” Gábor Kovács (CC BY 2.0)

Anywhere you use a base class, you should be able to use a subclass and not know it

“Mandarin Duck” Tambako The Jaguar (CC BY-ND 2.0)

Liskov Substitutability Principle

Interface Segregation Principle

Don’t force clients to use interfaces they don’t need

?

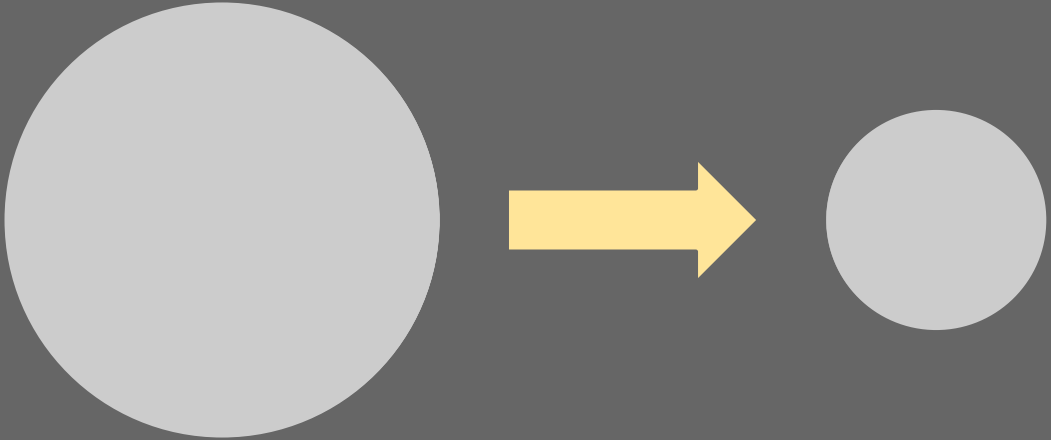
```
public interface IOStream {
    string read(count=-1);
    void write(string data);
}
```

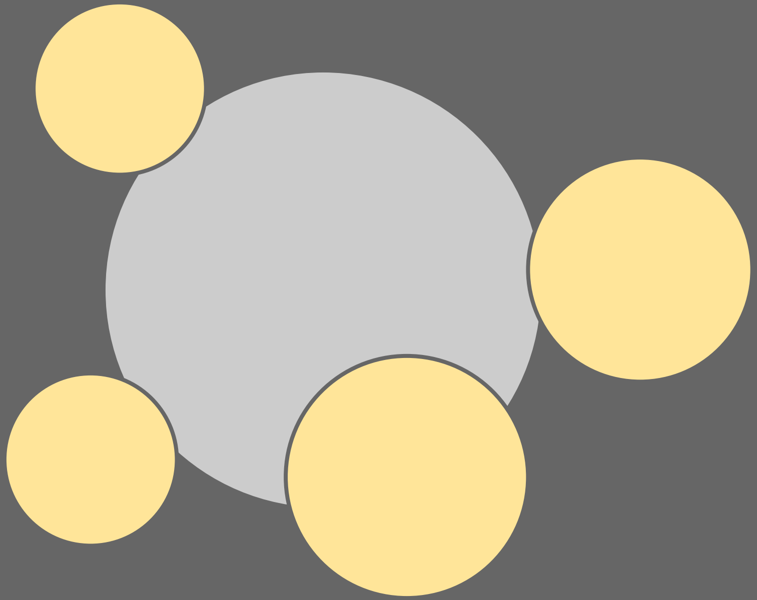
```
public interface InputStream {
    string read(count=-1);
}
```

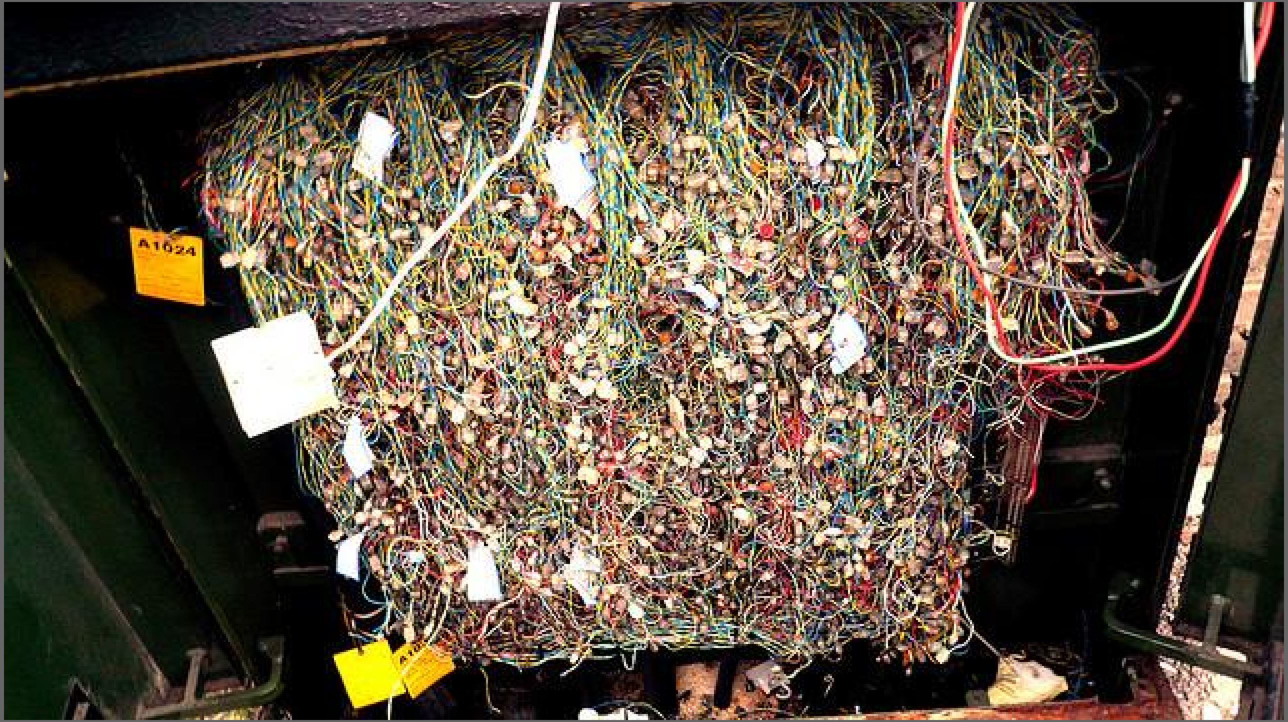
```
public interface OutputStream {
    void write(string data);
}
```

? ?

? ?







“Phone-wire tangle” Cory Doctorow (CC BY-SA 2.0)

Interface Segregation Principle

Dependency Inversion Principle

High-level modules shouldn’t rely on low-level modules

Both should rely on abstractions

```python
class MusicPlayer(object):
    def play(self): ...
    def pause(self): ...
    def stop(self): ...
    def next(self): ...
    def previous(self): ...
```

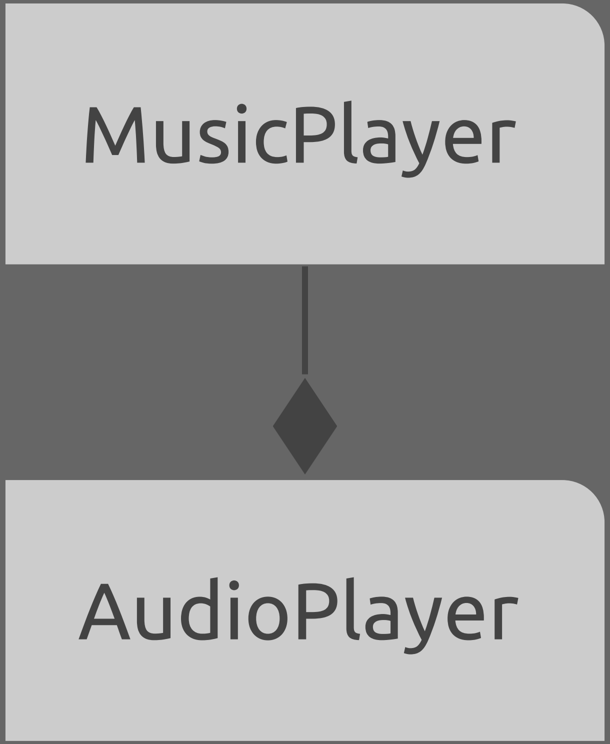
```python
class MusicPlayer(object):
    def play(self):
        song_file = self.playlist.current
        song_data = open(song_file, ‘r’).read()
        audio.load_data(song_data)
        audio.play_sound(0)
        self.state = MusicPlayer.PLAYING
    def next(self):
        ...
```

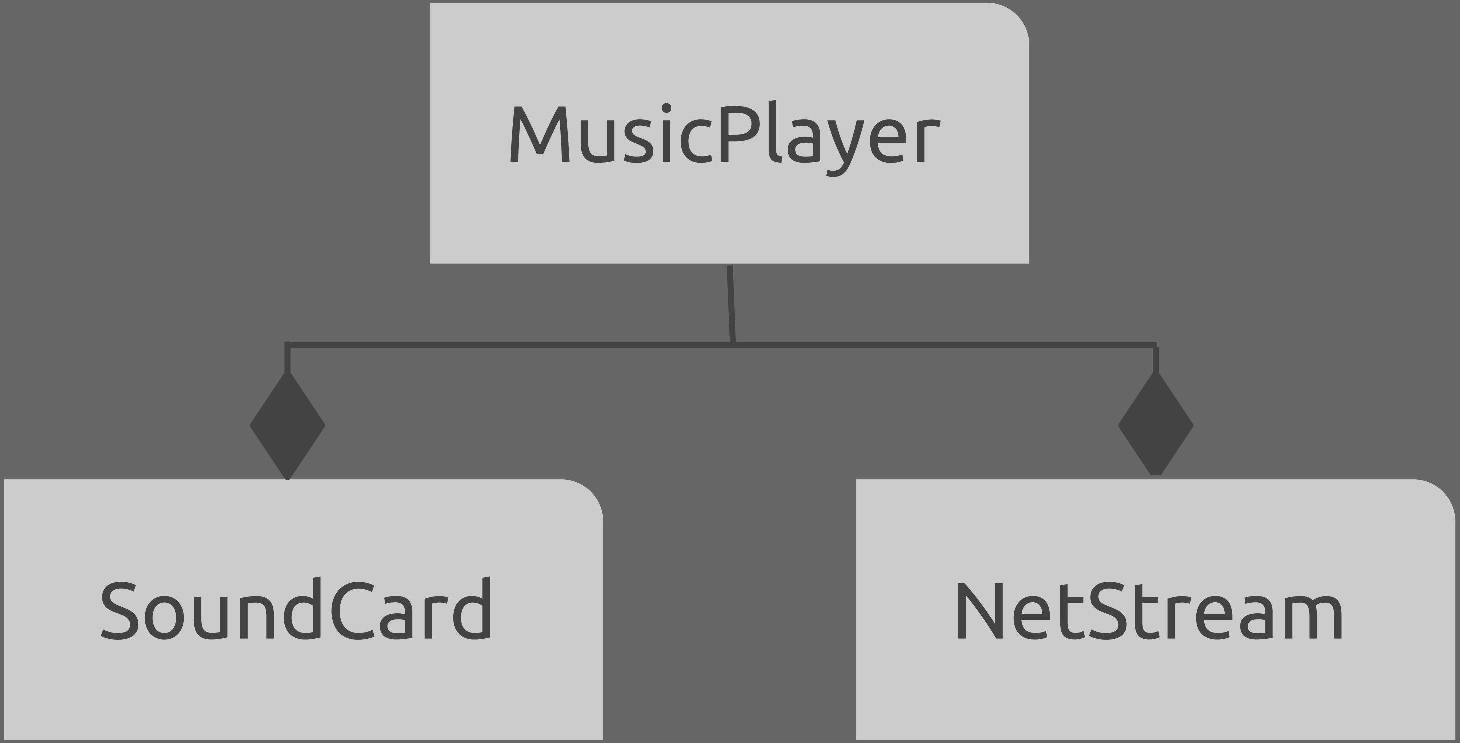
State of playback from playlist

`Reading files, audio subsystem`

“Abstract 1.1” Steve Snodgrass (CC BY 2.0)

MusicPlayer


MusicPlayer
AudioPlayer


MusicPlayer
SoundCard
NetStream

Dependency Inversion Principle

Single responsibility principle

Open/closed Principle

Liskov substitutability principle

Interface segregation principle

Dependency Inversion Principle

Tell, Don't Ask

Tell objects to do
the work,
don’t *ask* them
for their data

```python
def calculate(self):
    cost = 0
    for line_item in self.bill.items:
        cost += line_item.cost
    ...
```

```python
def calculate(self):
    cost = self.bill.total_cost()
    ...
```

```
def calculate(self, pos, vel):
    # Calculate amplitude of velocity
    abs_vel = math.sqrt(sum((vel.x**2,
                             vel.y**2,
                             vel.z**2))
    ...
```

```python
def calculate(self, position, velocity):
    vel = abs(velocity)
    …
```

Tell, Don't Ask

Design Principles

Unit-testing

```
def load(self):
    with open(BASE_SETTINGS, 'r') as settings:
        try:
            load_base_settings(settings)
        except LoadError:
            log.error(“Failed to load %s”, BASE_SETTINGS)
    with open(PLUGIN_SETTINGS, 'r') as settings:
        try:
            load_plugin_settings(settings)
        except LoadError:
             log.error(“Failed to load %s”, PLUGIN_SETTINGS)
    with open(EXTENSION_SETTINGS, 'r') as settings:
        …
```

```
@patch("__builtin__.open")
def test_loading_base_settings(self, mock_open):
    settings_data = [BASE_SETTINGS, PLUGIN_SETTINGS, …]
    mock_open.side_effect = lambda: StringIO(settings_data.pop())
    self.testobject.load()

    self.assertEquals(self.testobject.property1, …)
    self.assertEquals(self.testobject.property2, …)
    self.assertEquals(self.testobject.property3, …)
    …

@patch("__builtin__.open")
def test_loading_bad_base_settings(self, mock_open):
    settings_data = [BAD_BASE_SETTINGS, PLUGIN_SETTINGS, …]
    …
```

```python
def load(self):
    for settings_file, loader in CONFIG_LOAD_MAP:
        try_to_load(settings_file, loader)
```

```python
@patch("__builtin__.open")
def test_loading_settings(self, mock_open):
    mock_file = Mock()
    mock_open.return_value = mock_file

    self.testobject.load()

    mock_open.assert_called_once_with(FILE_PATH, "r")
    self.mock_loader.assert_called_once_with(mock_file)

@patch("__builtin__.open")
def test_loading_bad_settings(self, mock_open):
    mock_open.side_effect = IOError()
    self.testobject.load()  # Catches IOError

    self.assertEqual(False, self.testobject.loaded)
```

```python
def process_frame(self):
    frame = self.input_processor.top()

    start_addr = frame.addr
    pow2_size = 1
    while pow2_size < frame.offs:
        pow2_size <<= 1
    end_addr = start + pow2_size
    o_map = io_map.new_map(start_addr, end_addr)

    self.output_processor.flush(o_map)
```

```python
@patch('iomap')
def test_process_frame_calculates_nearest_pow2_offset(self,
                                                      iomap_mock):
    input_proc = Mock()
    input_prov.addr = 0
    input_prov.offs = 24
    output_proc = Mock()

    uut = FrameProcessor(input_proc, output_proc)
    uut.process_frame()

    iomap_mock.new_map.assert_called_once_with(0, 32)
```

```python
def process_frame(self):
    frame = self.input_processor.top()
    o_map = self.memory_mapper.map(frame)
    self.output_processor.flush(o_map)
```

```python
def test_process_frame_flushes_iomap(self):
    mem_mapper = Mock(MemMapper)
    output_proc = Mock(OutputProcessor)

    uut = FrameProcessor(Mock(InputProcessor),
                         mem_mapper, output_proc)
    uut.process_frame()

    output_proc.flush.assert_called_once_with(
                             mem_mapper.map())
```

Parting Thoughts

Think 'Objects'

“Tortoise” SteveR- (CC BY 2.0)

"why was this difficult?"

“Am I missing an object?”

Objects express the domain

```python
if not valid_user(user):
   return -1

c = netpkg.open_connection("uri://server.path",
port=57100, flags=netpkg.KEEPALIVE)
if c is None:
   return -1

files = [str(f) for f in c.request(netpkg.DIRLIST)]
for source in files:
    local_path = "/home/%s/Downloads/%s" \
                   % (user_name, source)
    data = c.request(netpkg.DATA, source)
    with open(local_path, 'w') as local:
        local.write(data)
```

```
authenticate(user)
connection = connect(user, server)

files = RemoteDirectory(connection)
download = Downloader(files)

download.to(user.downloads_dir)
```

Q&A

Thank you!