class attributes

versus

instance attributes


CARGO
CONTAINER
CARGO
CONTAINER
CARGO
CONTAINER
CARGO
CARGO
CG 044 233 01
MAX. LOAD
TARE
MAX SPEED
100 km/h

Monday, 24

static methods

with the

@staticmethod

decorator

The static terminology is a relic from C and C++

```
// static_member_functions.cpp
#include <stdio.h>

class StaticTest
{
private:
    static int x;
public:
    static int count()
    {
        return x;
    }
};

int StaticTest::x = 9;

int main()
{
    printf_s("%d\n", StaticTest::count());
}
```

Text

class methods

with the

@classmethod

decorator

Choosing

@staticmethod or @classmethod

No access needed to either *class* or *instance* objects.

Most likely an implementation detail of the class.

May be able to be moved to become a module-scope function

Requires access to the class object to call other class methods or the constructor.

class methods

for named constructors

static methods

with

inheritance


CAXU
HJCU
8281 98 8
22G1
MAX GROSS
TARE
PAYLOAD
CUBE
MSCU
M. G. W.
TARE
NET
CU.CAP.
HANJIN
SHIPPING
CO.,LTD.

Monday, 24

class methods

with

inheritance

encapsulation

using the

@property

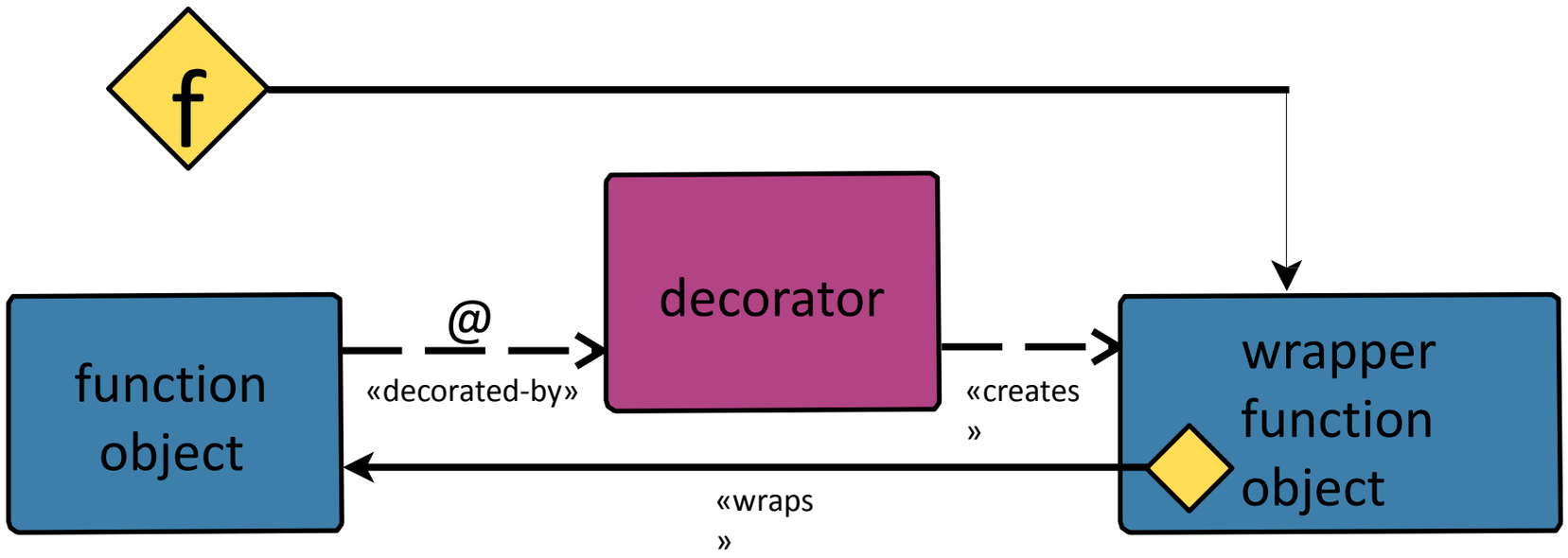
decorator


python™

≠

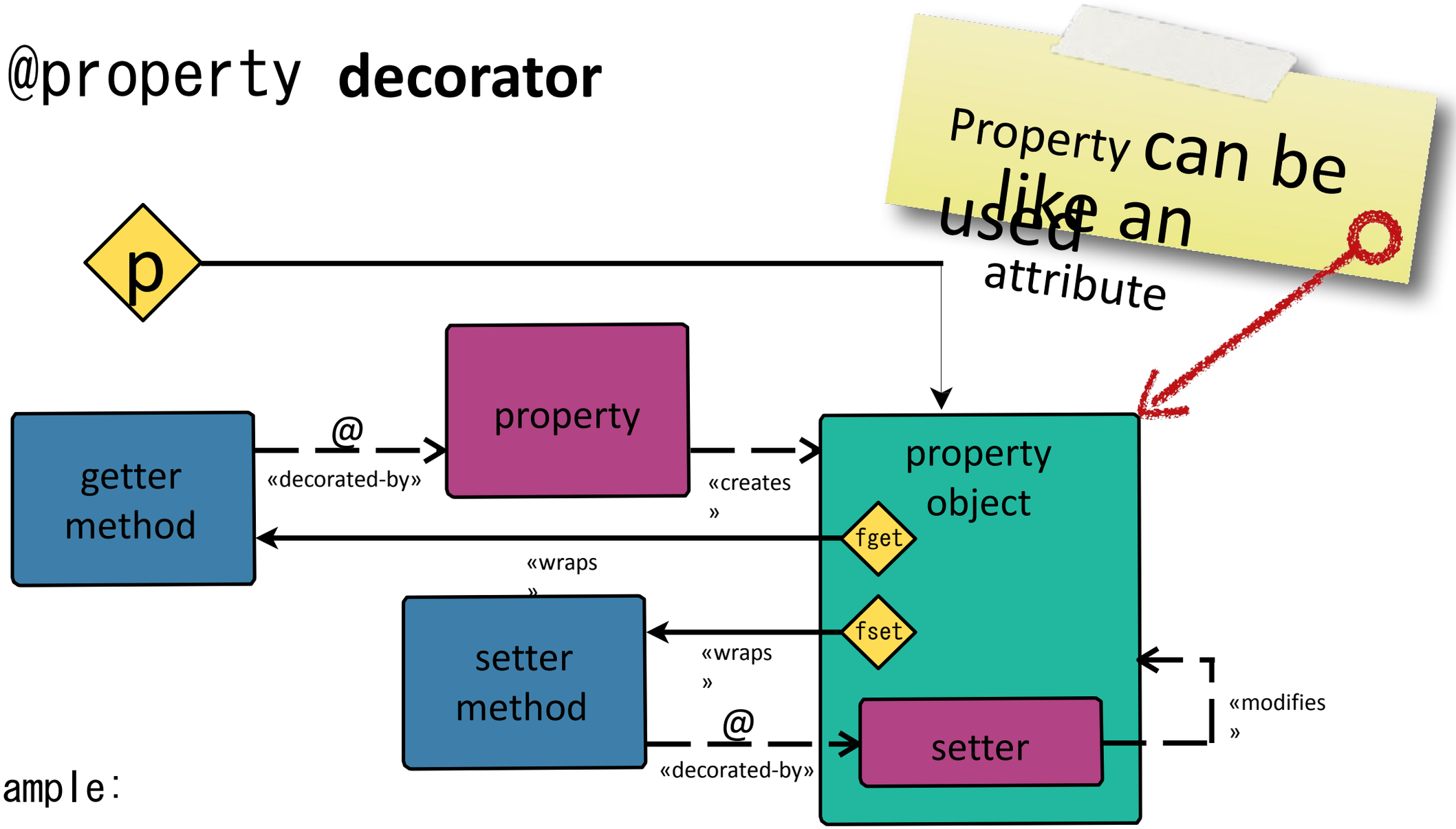

Java™

Decorator recap




```
@decorator
def f():
    do_som
    ething
    ()
```

The @property decorator




```
class Example:

    @property
    def
    p(self):
        return self._p

    @p.setter
    def     p(self,
        value): self._p
            =
        value
```

Inheritance

interaction with the

@property

decorator

Summary: Properties and Class Methods

- Shown the differences between class attributes and instance attributes.
- Illustrated how class attributes are shared amongst all instances.
- Demonstrated how to access class attributes by fully qualifying with the class name.
- Shown that attempting to assign to a class attribute through the instance reference `self` actually creates a new instance attribute.
- Used `@staticmethod` to decorate methods within a class which depend on neither class nor instance objects.
- Used `@classmethod` to decorate methods which operate on the class object.
- Implemented named constructors using class methods.
- Shown how inheritance interacts with static and class methods.
- Demonstrated class- and static-method polymorphism by invoking through the `self` instance.

Summary: Properties and Class Methods

- **Shown how to create read-only and read-write properties using the `@property` decorator.**
- **Used the *template method* design pattern to override properties without recourse to esoteric syntactical constructs.**