Abstract Base Classes

Robert Smallshire
COFOUNDER – SIXTY NORTH

@robsmallshire rob@sixty-north.com

Summary

Summary

Summary

Abstract Base Classes (ABCs)

Summary

Abstract Base Classes (ABCs)

PEP 3119

Summary

Abstract Base Classes (ABCs)

PEP 3119

e.g. collections.abc.Sequence

Summary

Abstract Base Classes (ABCs)

PEP 3119

e.g. collections.abc.Sequence

abc

Summary

Abstract Base Classes (ABCs)

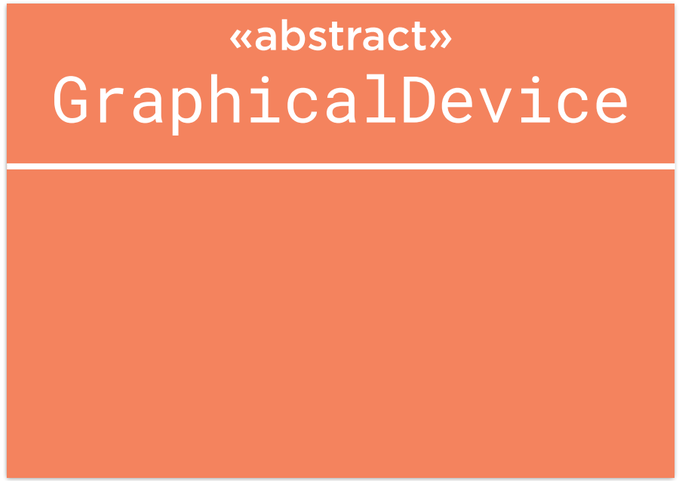
PEP 3119

e.g. collections.abc.Sequence

abc

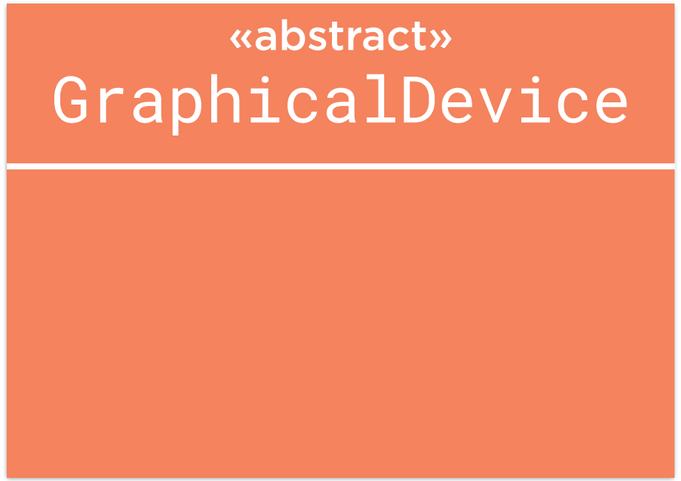
Python is not Java, C++, C#

What Is an Abstract Base Class?



What Is an Abstract Base Class?

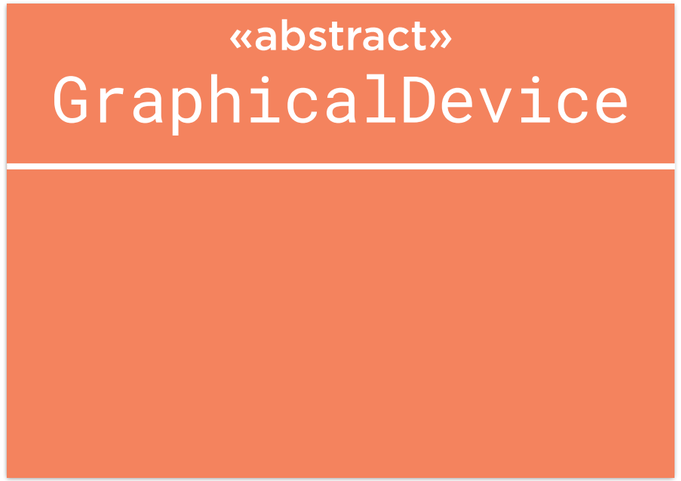
Base



What Is an Abstract Base Class?

Base

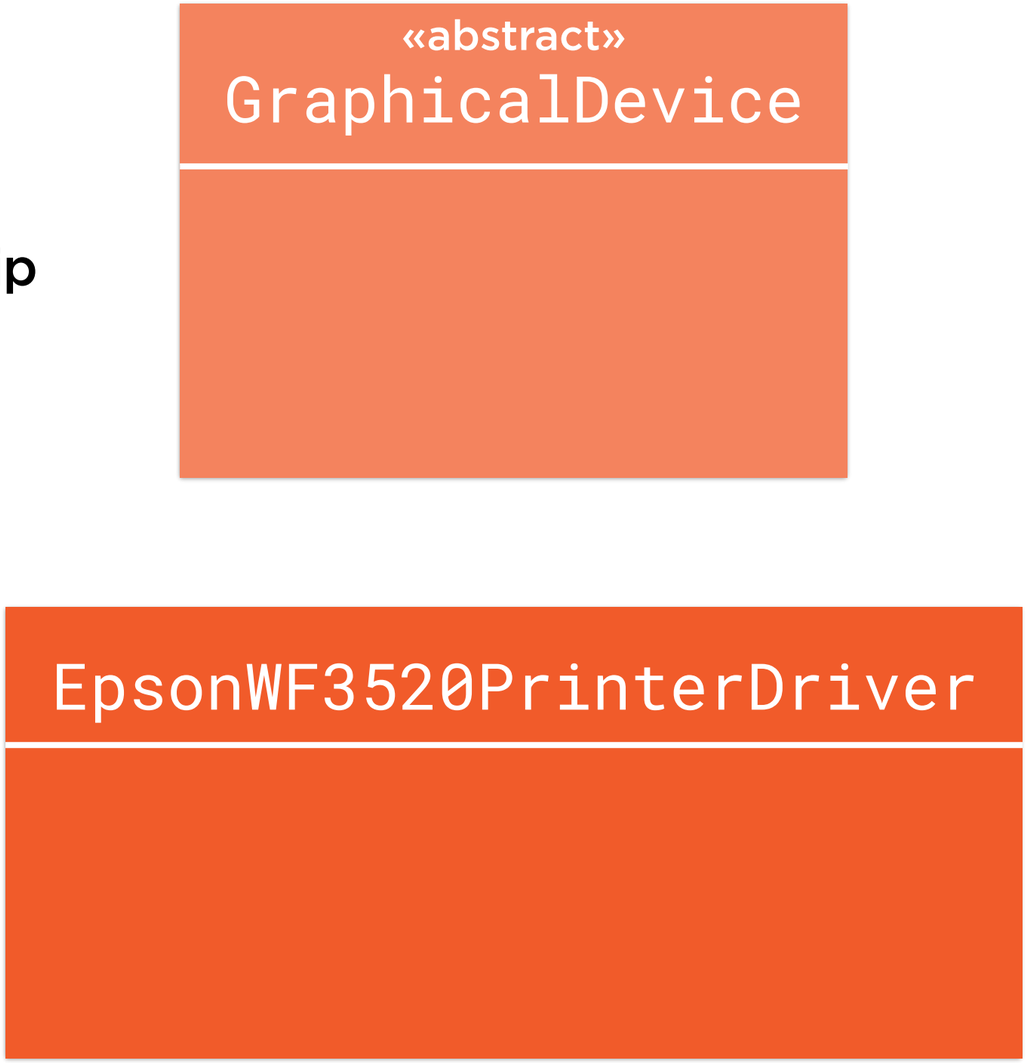
The target of an inheritance relationship from a subclass



What Is an Abstract Base Class?

Base

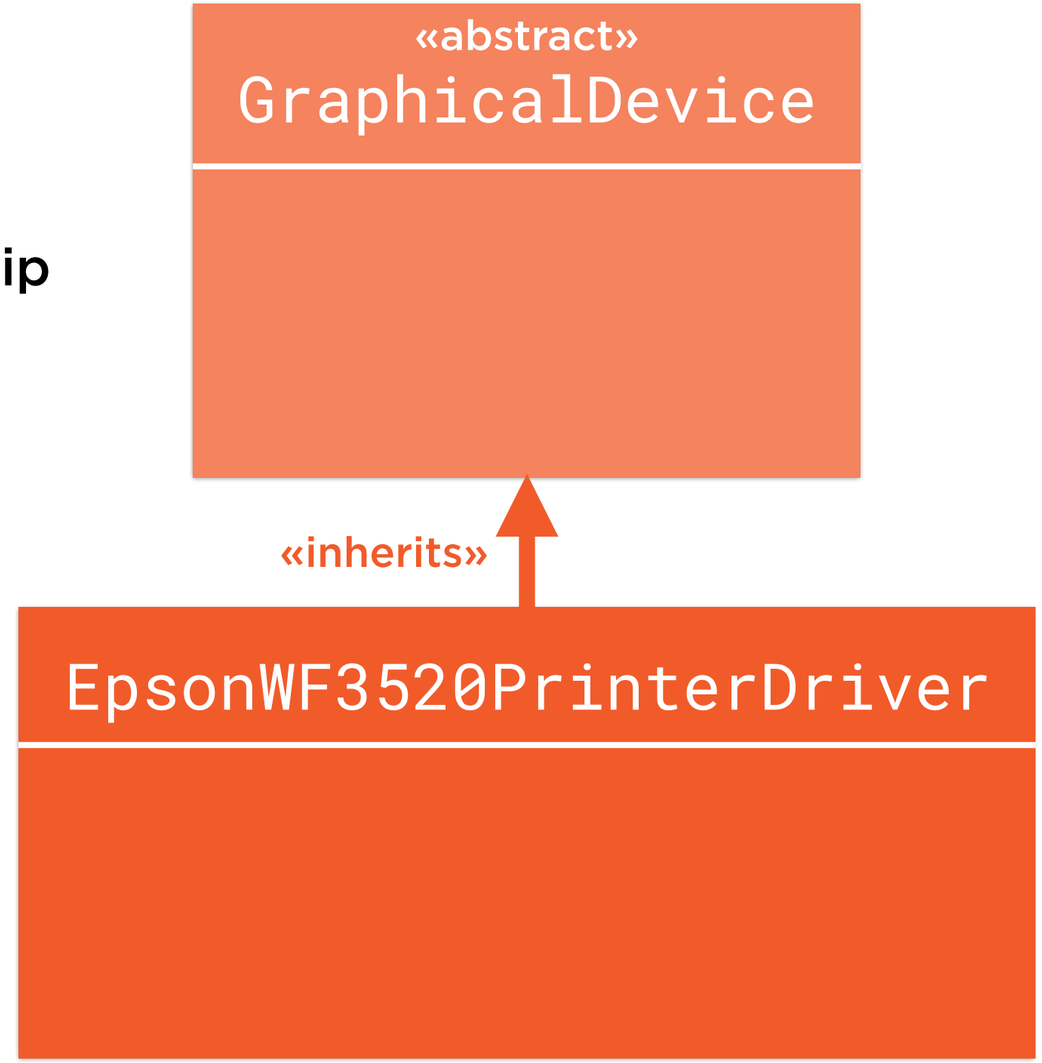
The target of an inheritance relationship from a subclass



What Is an Abstract Base Class?

Base

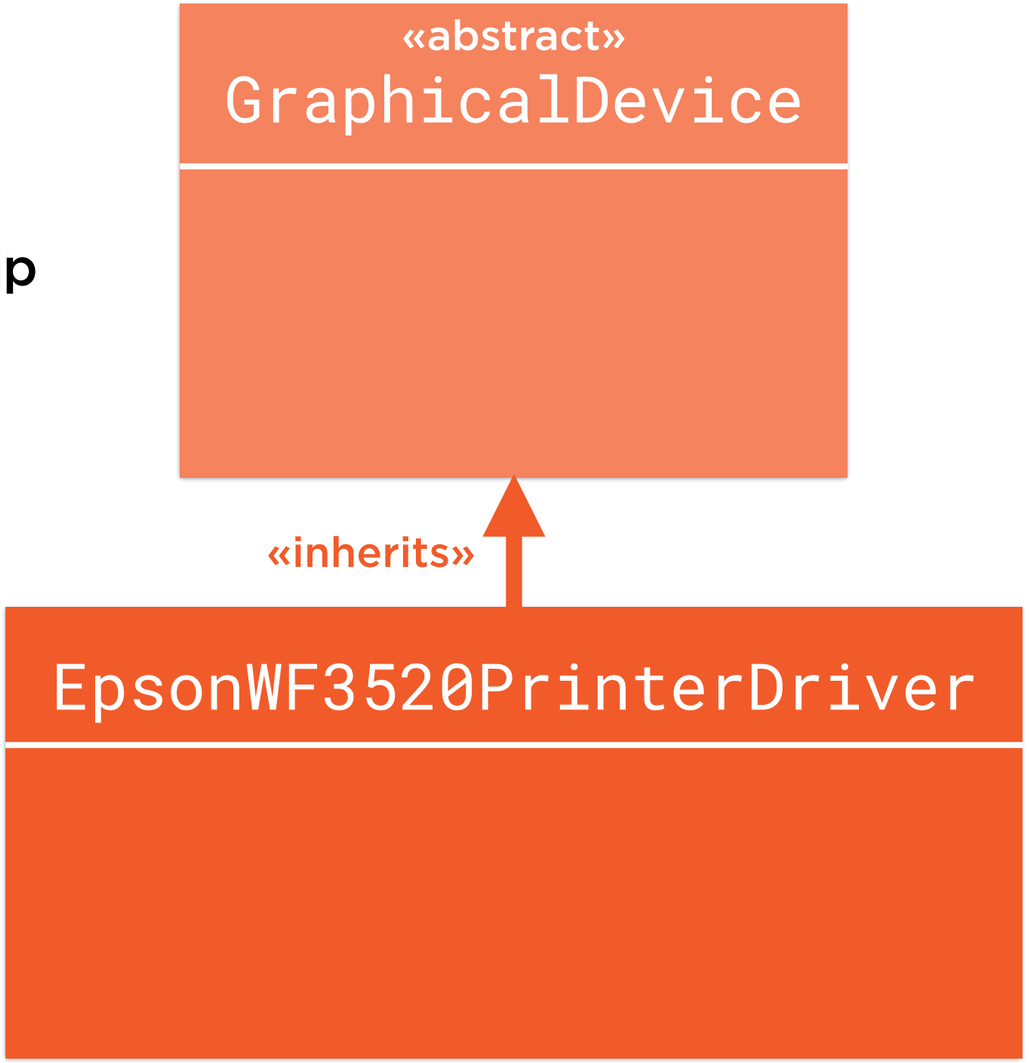
The target of an inheritance relationship from a subclass



What Is an Abstract Base Class?

Base

The target of an inheritance relationship from a subclass

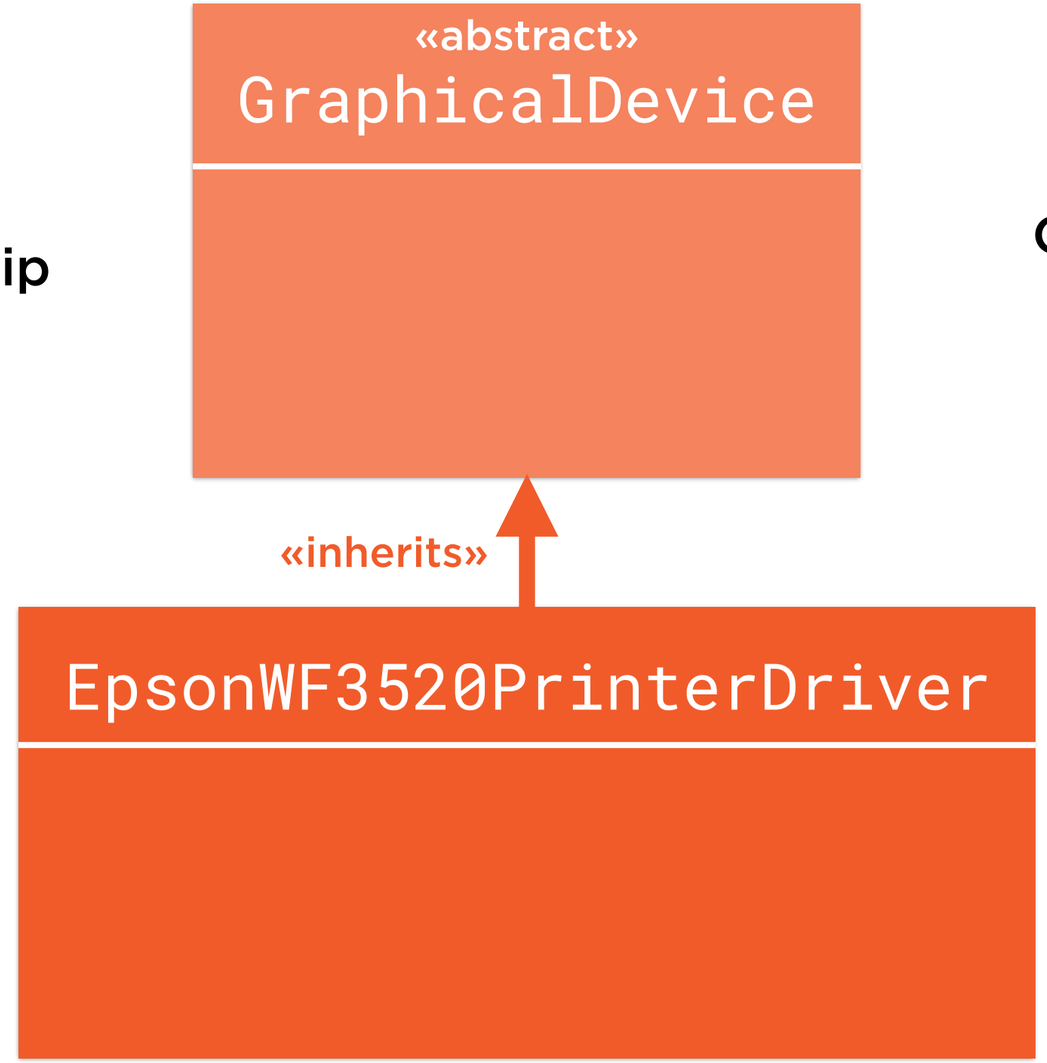


Abstract

What Is an Abstract Base Class?

Base

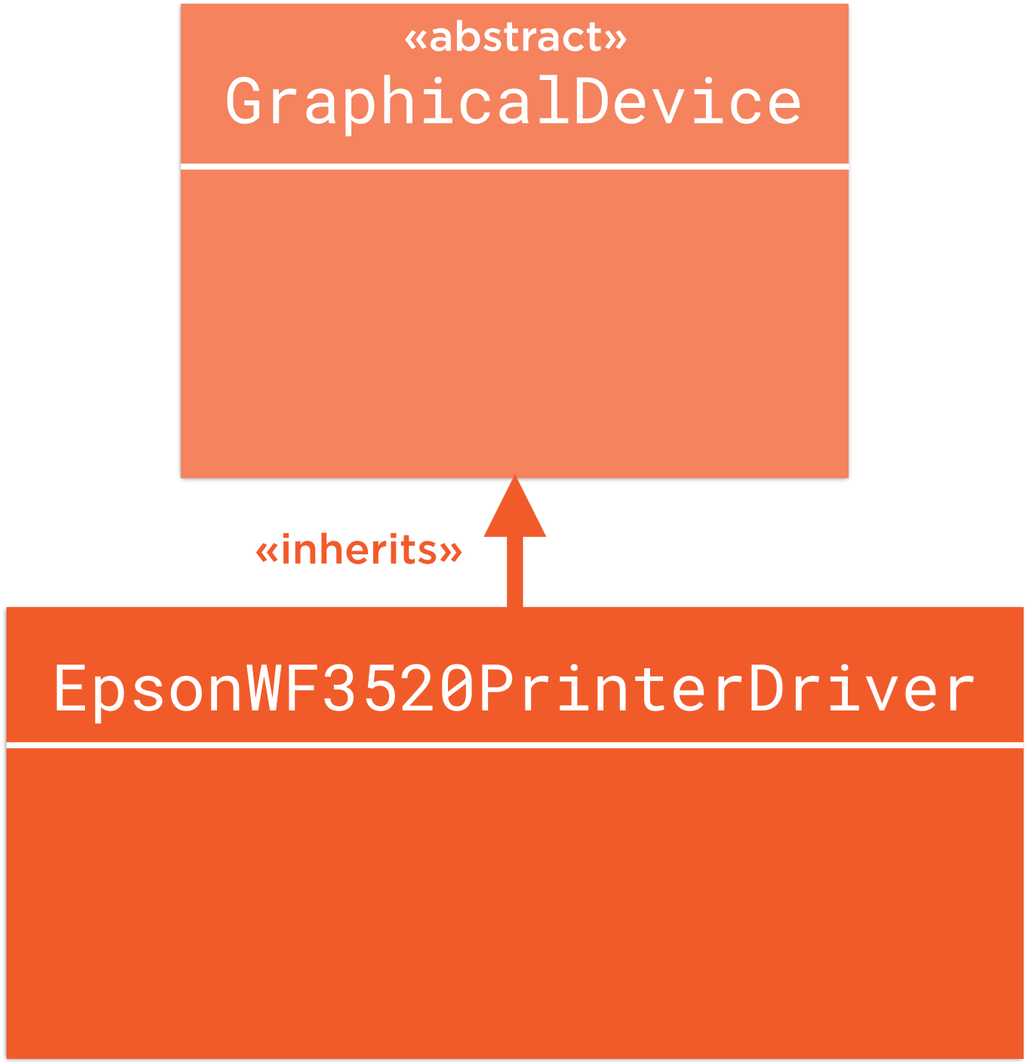
The target of an inheritance relationship from a subclass



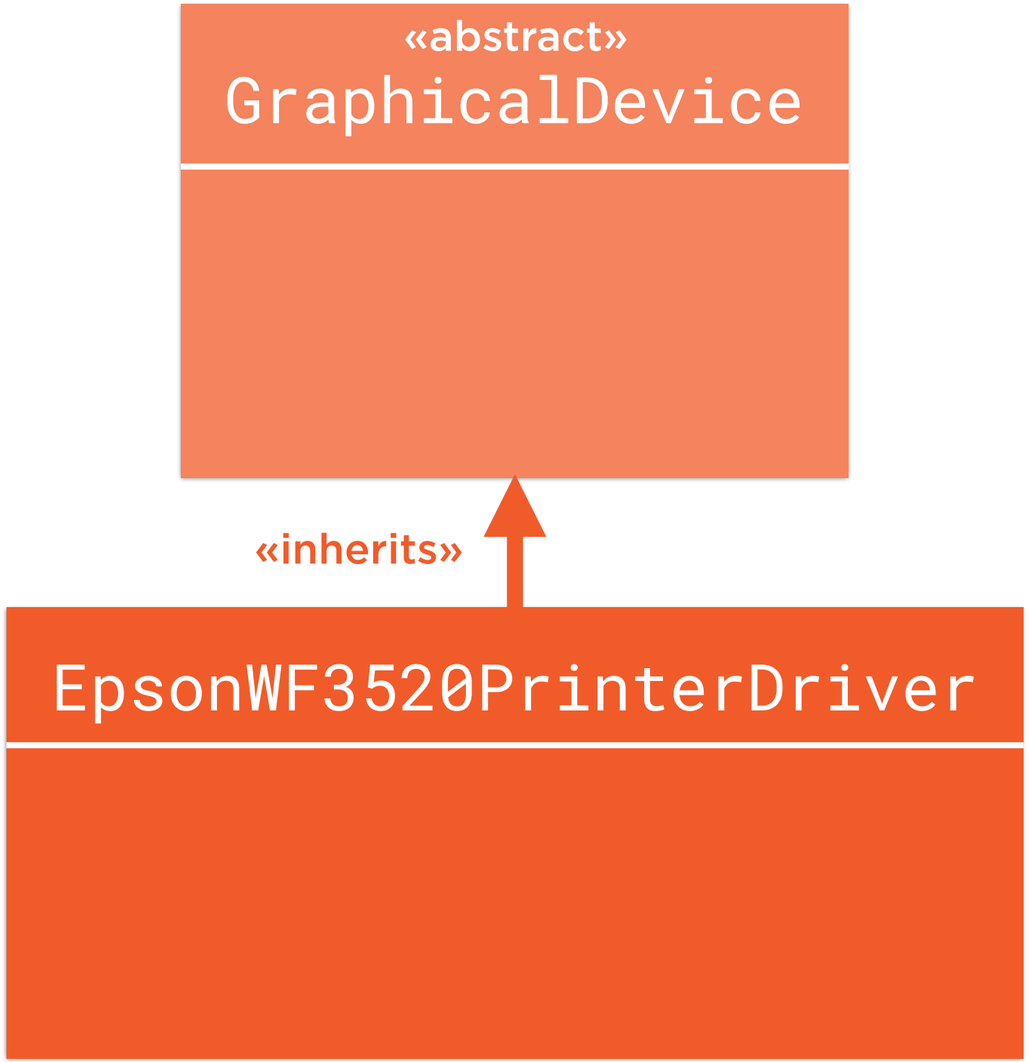
Abstract

Cannot be meaningfully instantiated

Why Are Abstract Base Classes Useful?

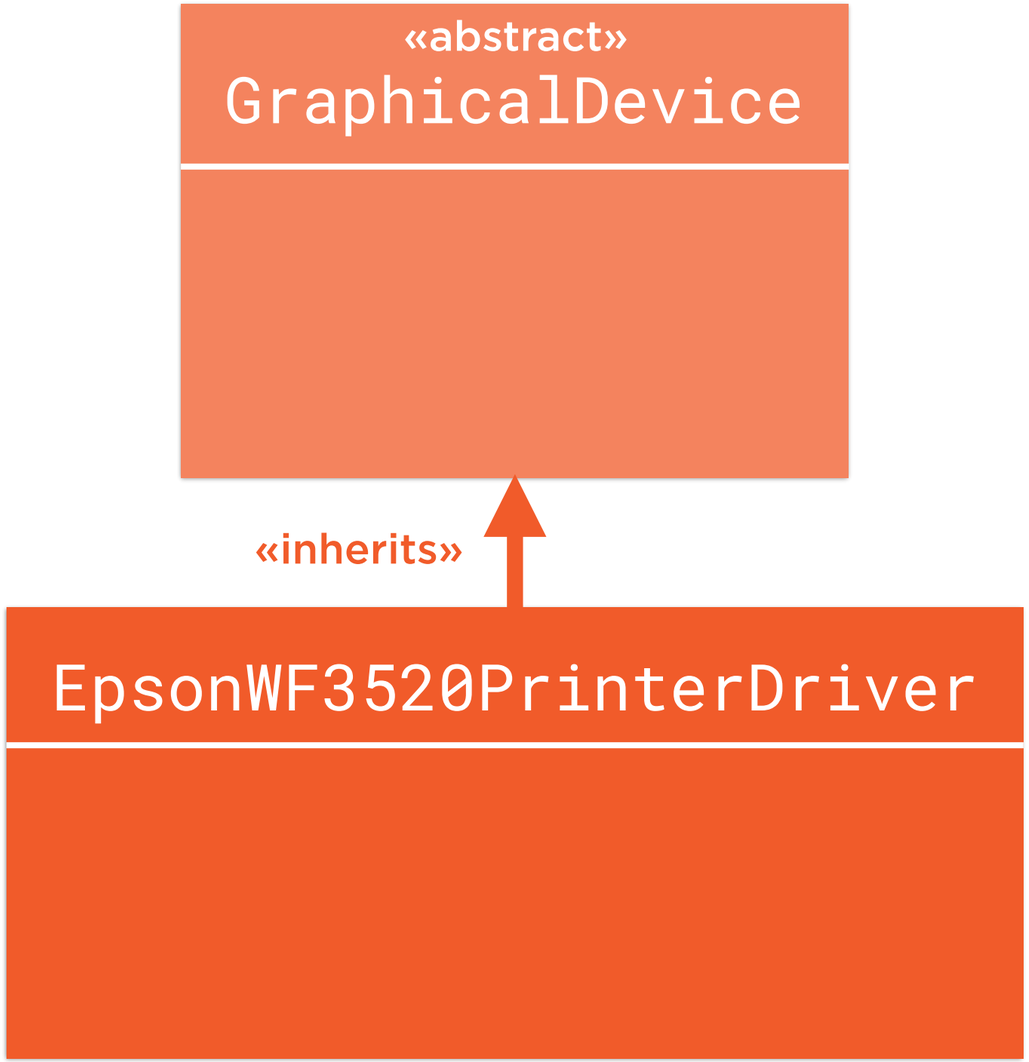


Why Are Abstract Base Classes Useful?



Interface Definition

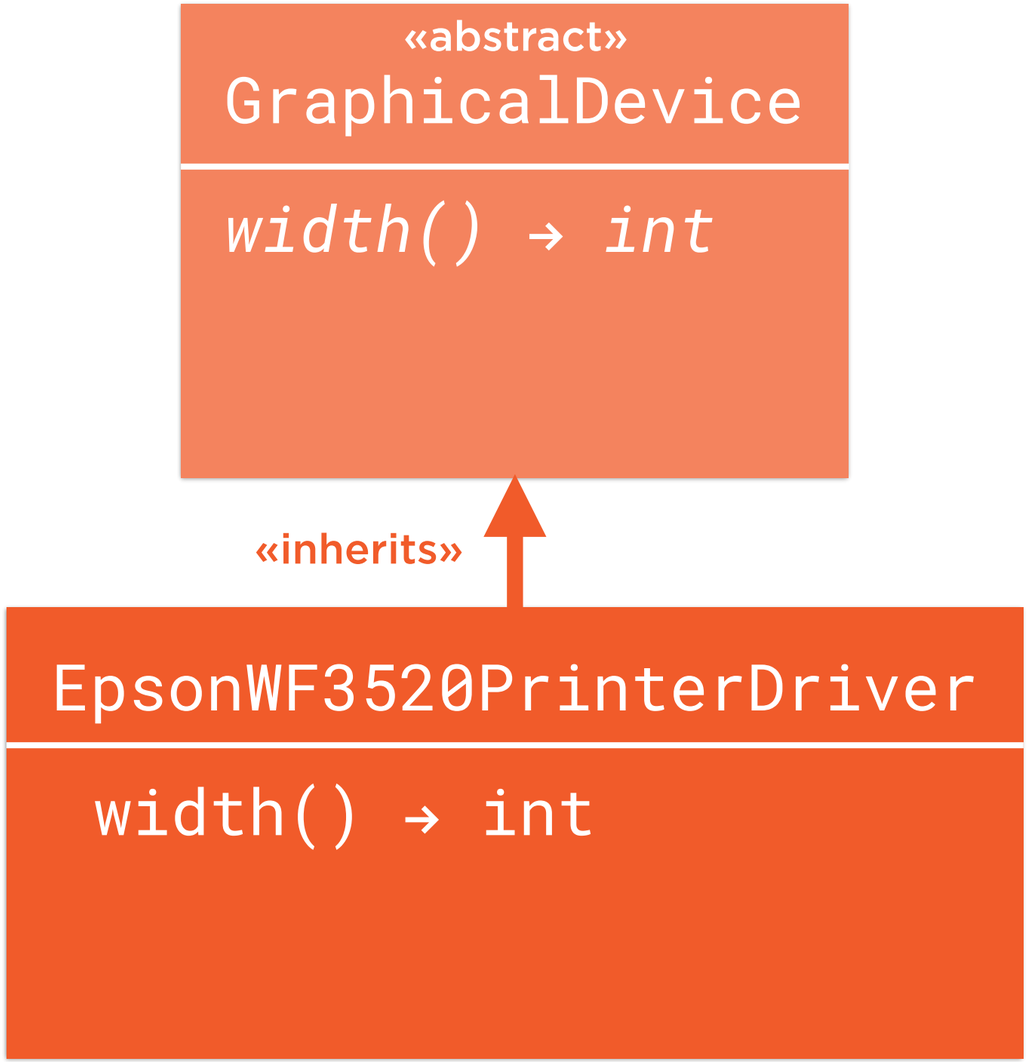
Why Are Abstract Base Classes Useful?



Interface Definition

The base class defines the interface for clients of any and all subclasses

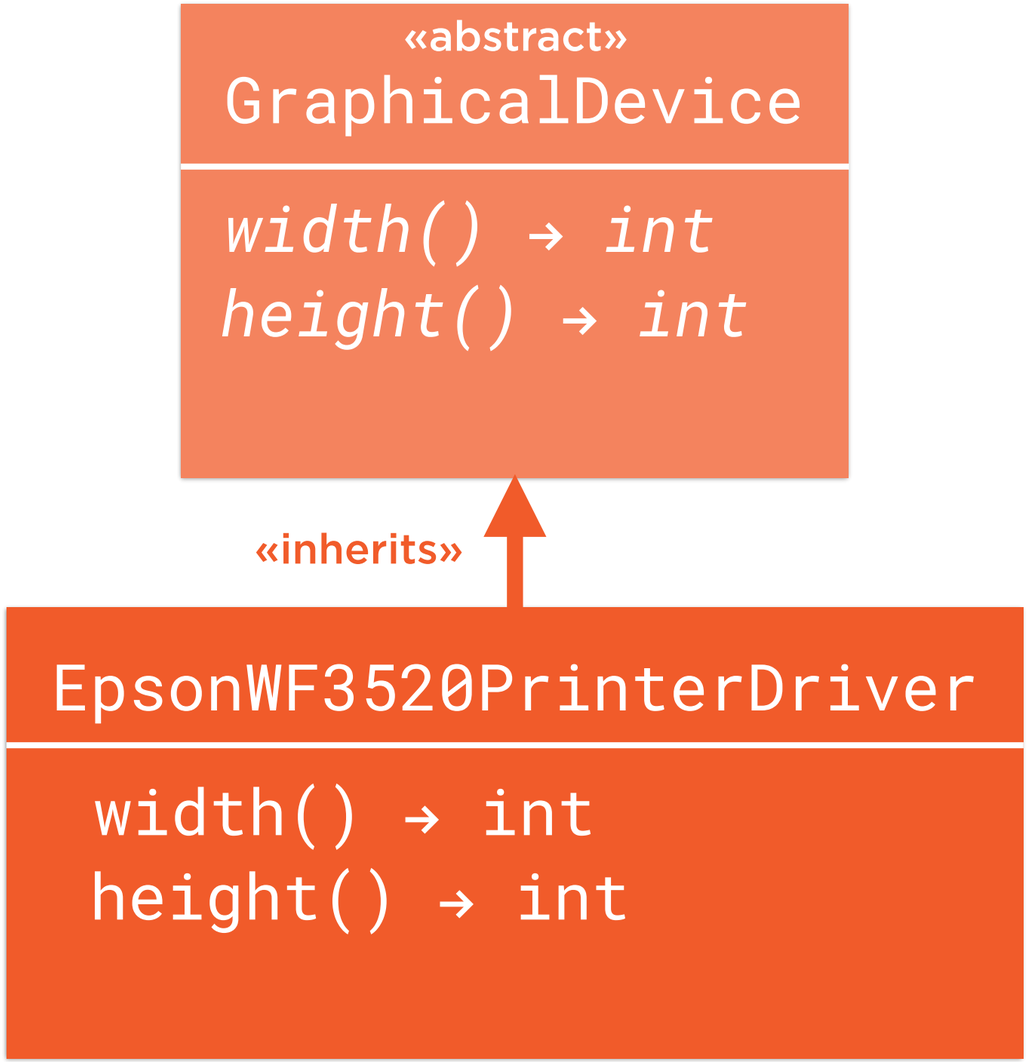
Why Are Abstract Base Classes Useful?



Interface Definition

The base class defines the interface for clients of any and all subclasses

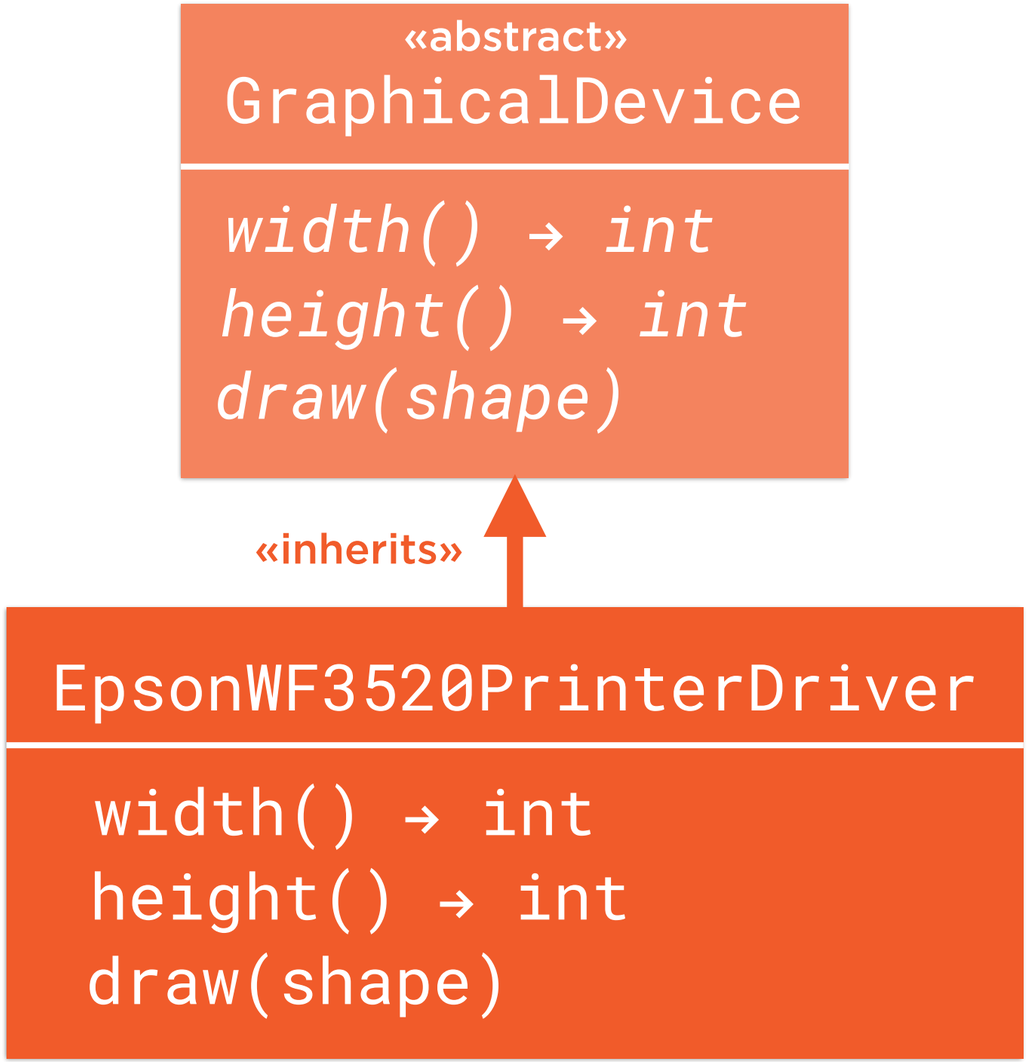
Why Are Abstract Base Classes Useful?



Interface Definition

The base class defines the interface for clients of any and all subclasses

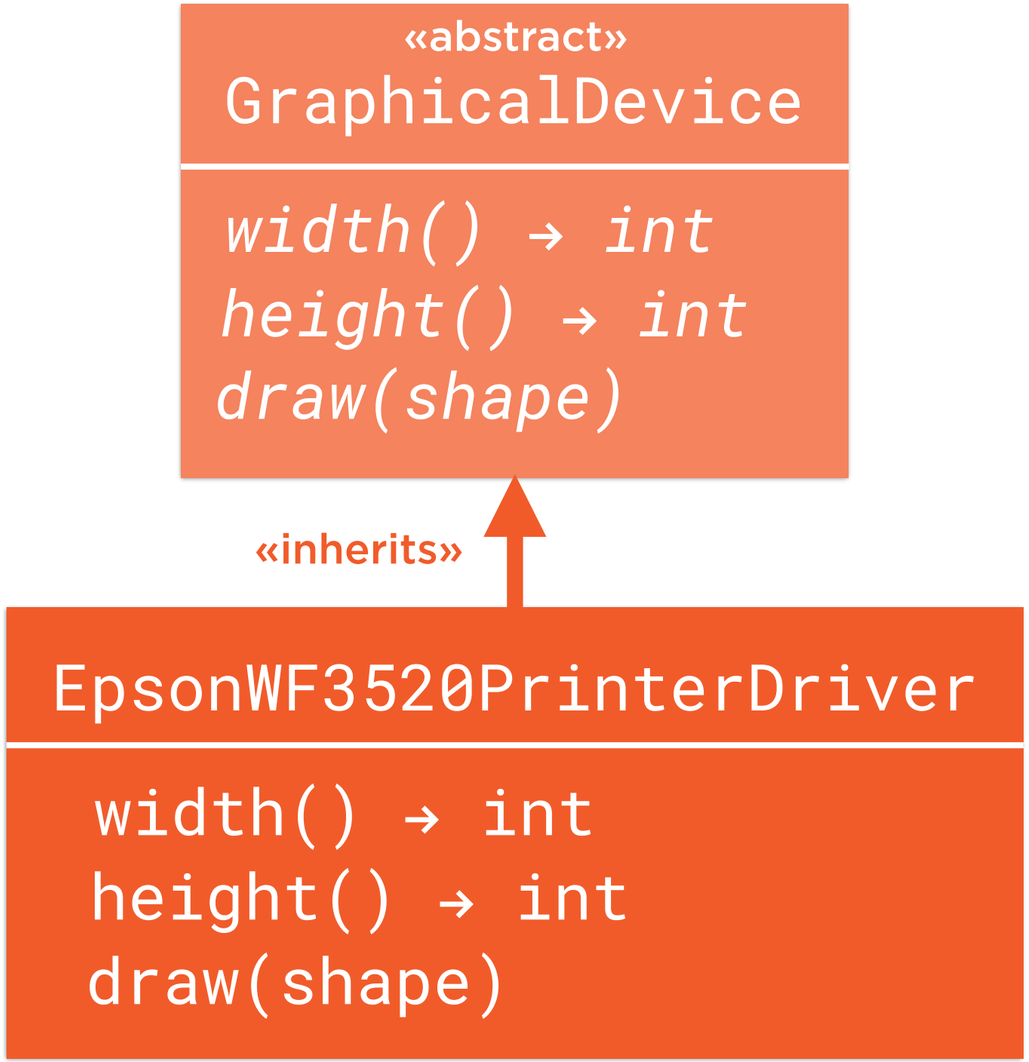
Why Are Abstract Base Classes Useful?



Interface Definition

The base class defines the interface for clients of any and all subclasses

Why Are Abstract Base Classes Useful?

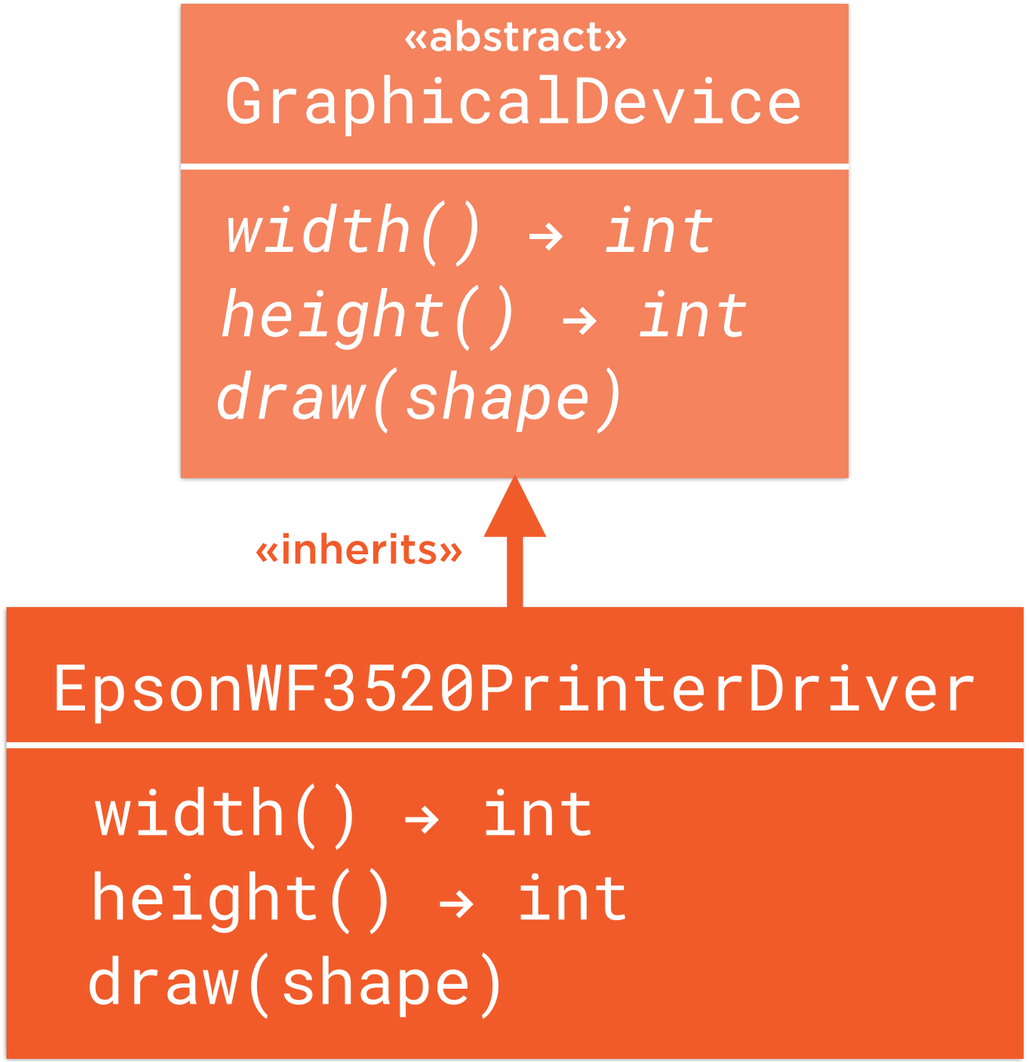


Interface Definition

The base class defines the interface for clients of any and all subclasses

Liskov Substitutability

Why Are Abstract Base Classes Useful?



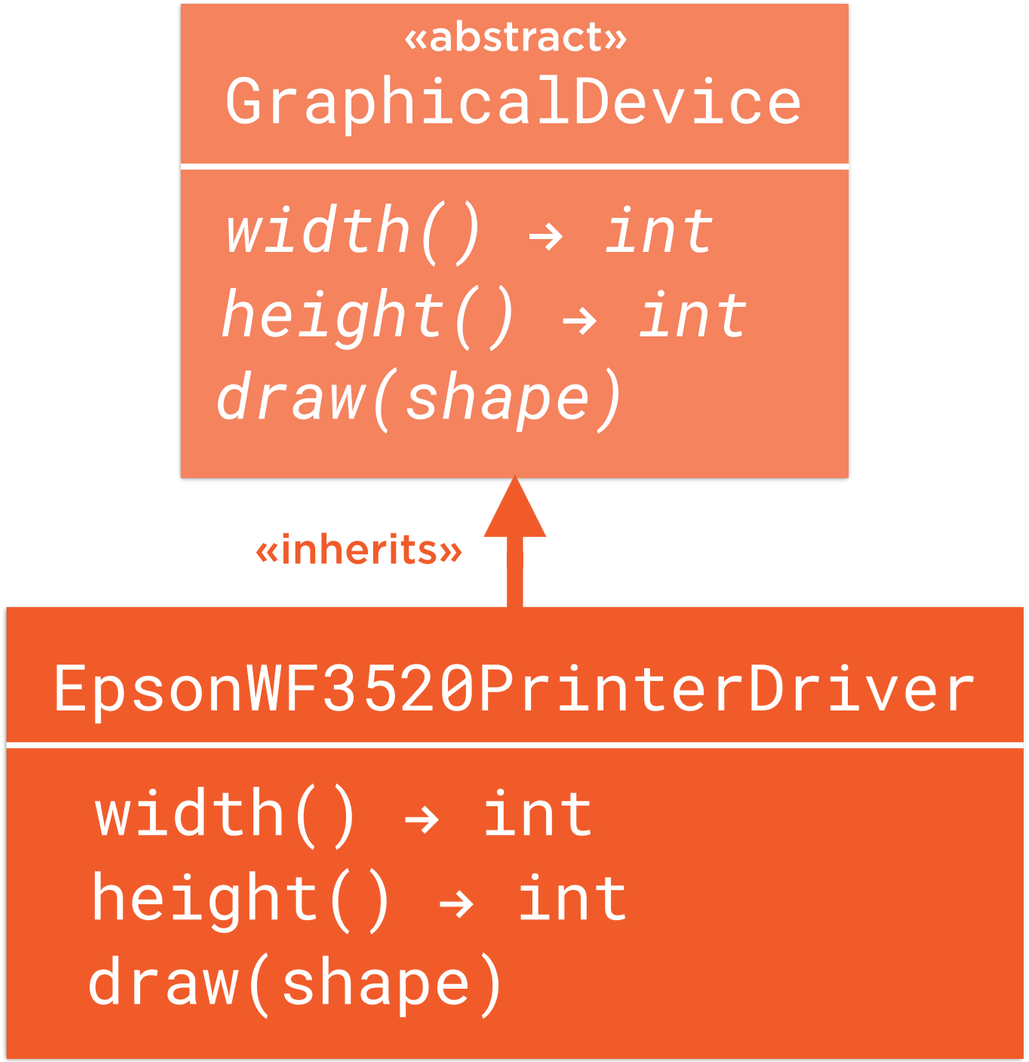
Interface Definition

The base class defines the interface for clients of any and all subclasses

Liskov Substitutability

Code relying only on the base class does not need to be modified for alternative subclasses

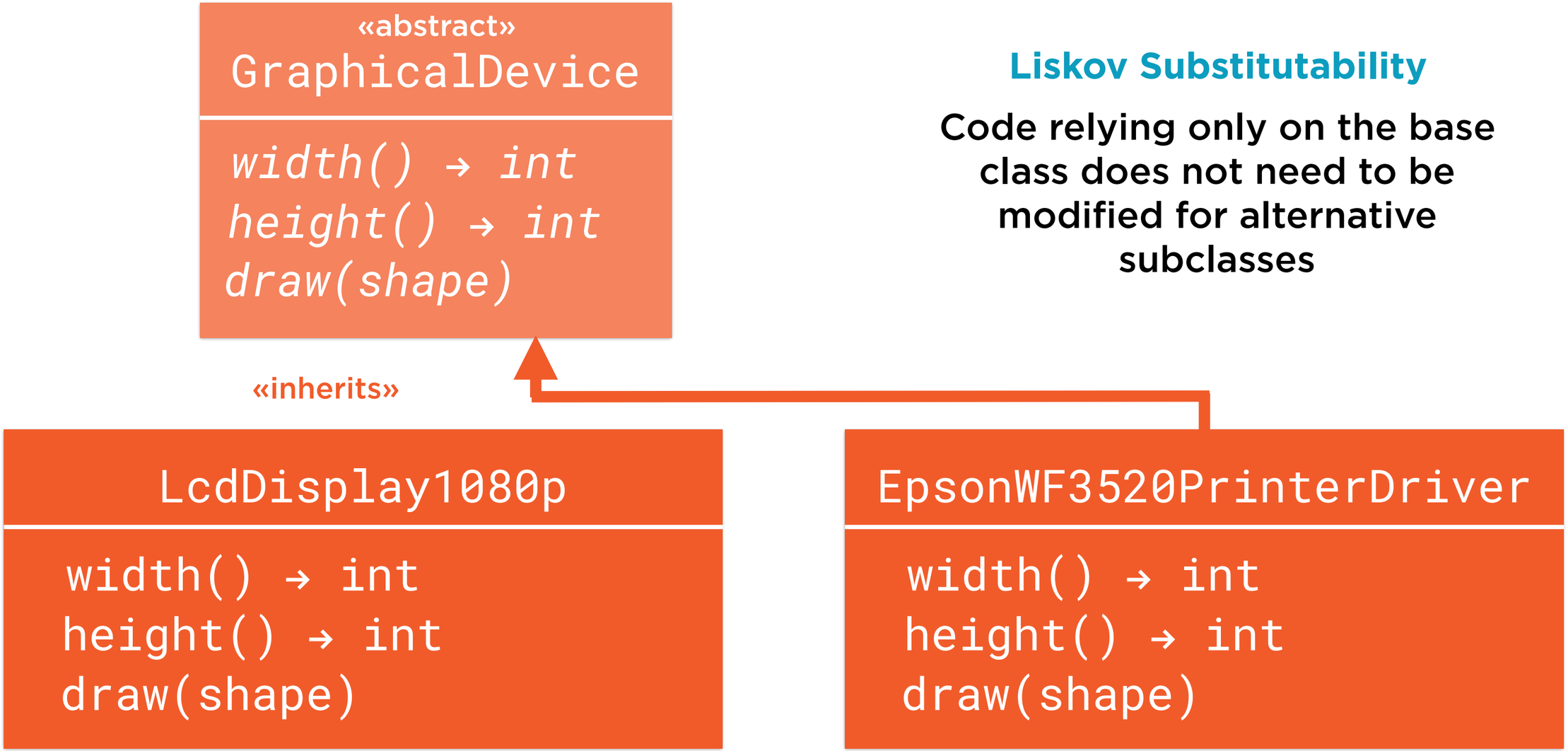
Why Are Abstract Base Classes Useful?



Liskov Substitutability

Code relying only on the base class does not need to be modified for alternative subclasses

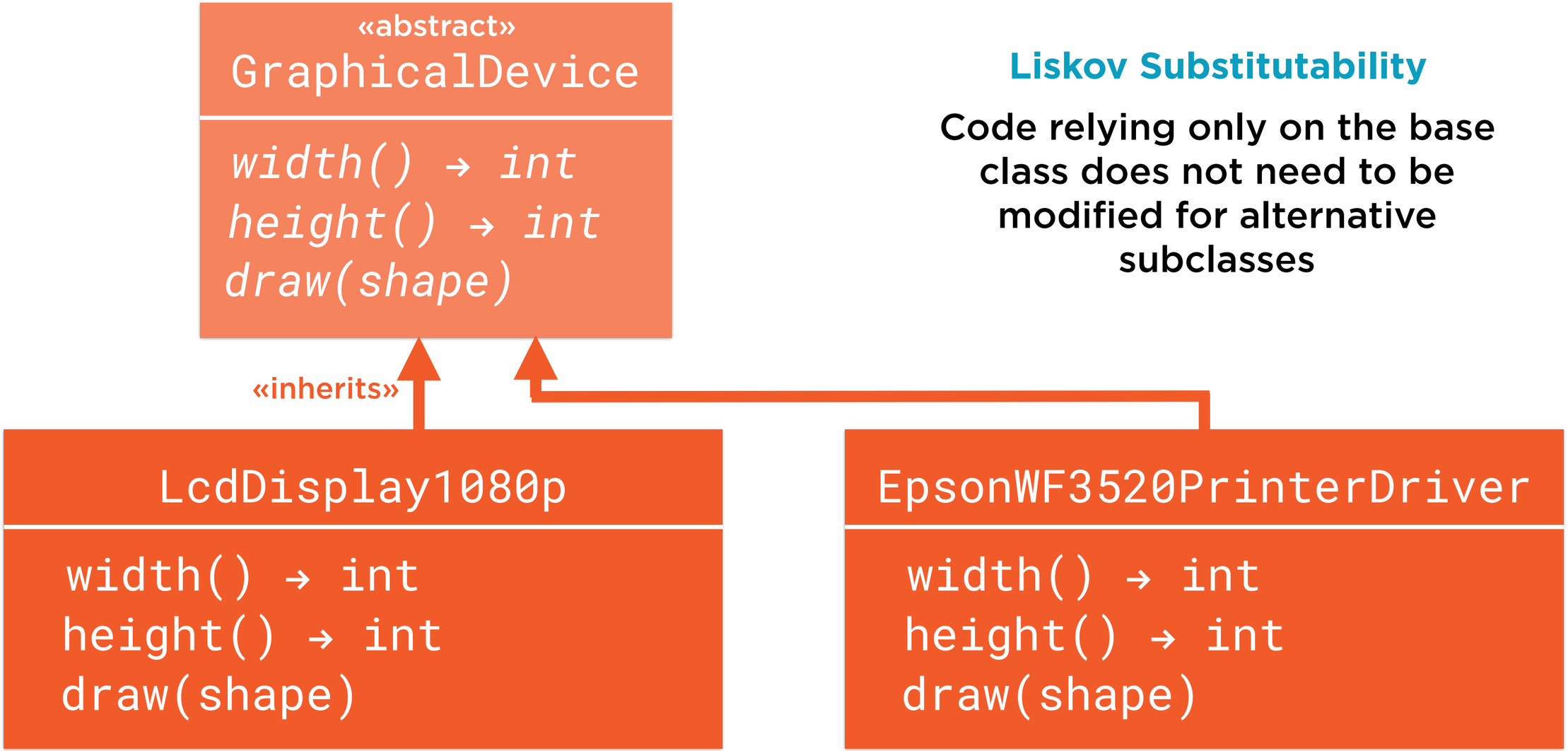
Why Are Abstract Base Classes Useful?



Liskov Substitutability

Code relying only on the base class does not need to be modified for alternative subclasses

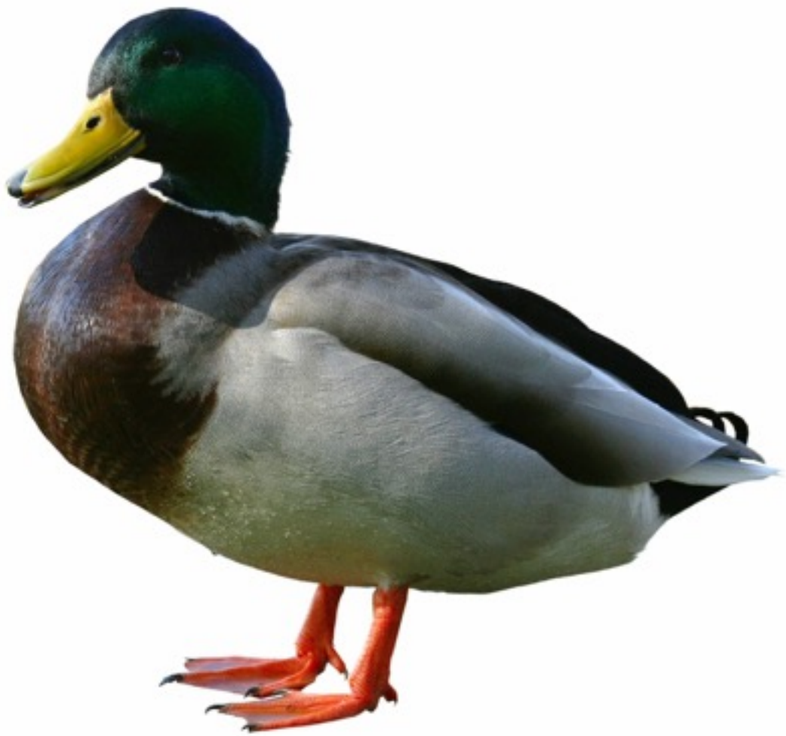
Why Are Abstract Base Classes Useful?



Liskov Substitutability

Code relying only on the base class does not need to be modified for alternative subclasses

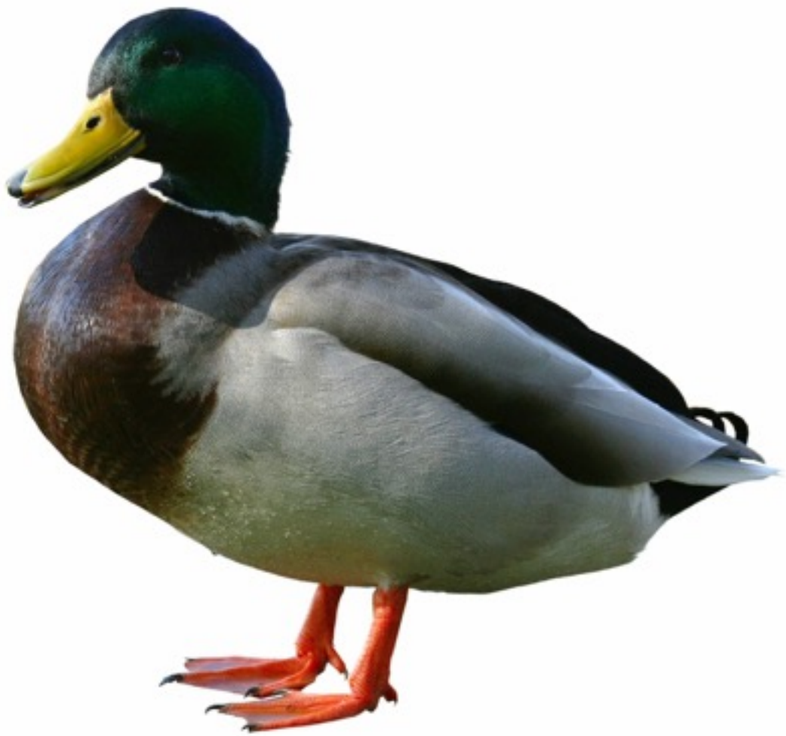
What About Duck Typing?



What About Duck Typing?

"When I see a bird that walks like a duck and swims like a duck and quacks like a duck, I call that bird a duck."

James Whitcomb Riley
American poet and author



A conforming MutableSequence must implement all 16 methods.

«abstract»

MutableSequence

```
__contains__(item) → bool
__delitem__(index)
__iadd__(iterable)
__iter__() → iterator
__getitem__() → object
__len__() → int
__reversed__() → iterator
__setitem__() → object
append(item)
count(item) → int
extend(iterable)
index(item) → int
insert(index, item)
pop() → object
remove(item)
reverse()
```

A conforming MutableSequence must implement all 16 methods.

Abstract Base Classes in Python

Motivations for Abstract Base Classes

Motivations for Abstract Base Classes

Specification

ABCs are effective for specifying interface protocols

Motivations for Abstract Base Classes

Specification

ABCs are effective for specifying interface protocols

Detection

ABCs can be used to detect conforming objects

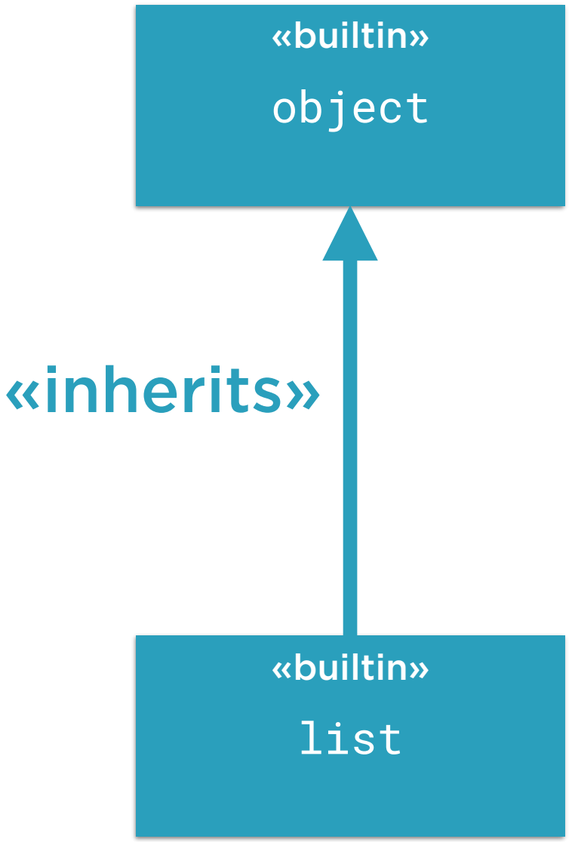
Python Is Not Java, C#, C++, *etc*.

Python Is Not Java, C#, C++, *etc*.

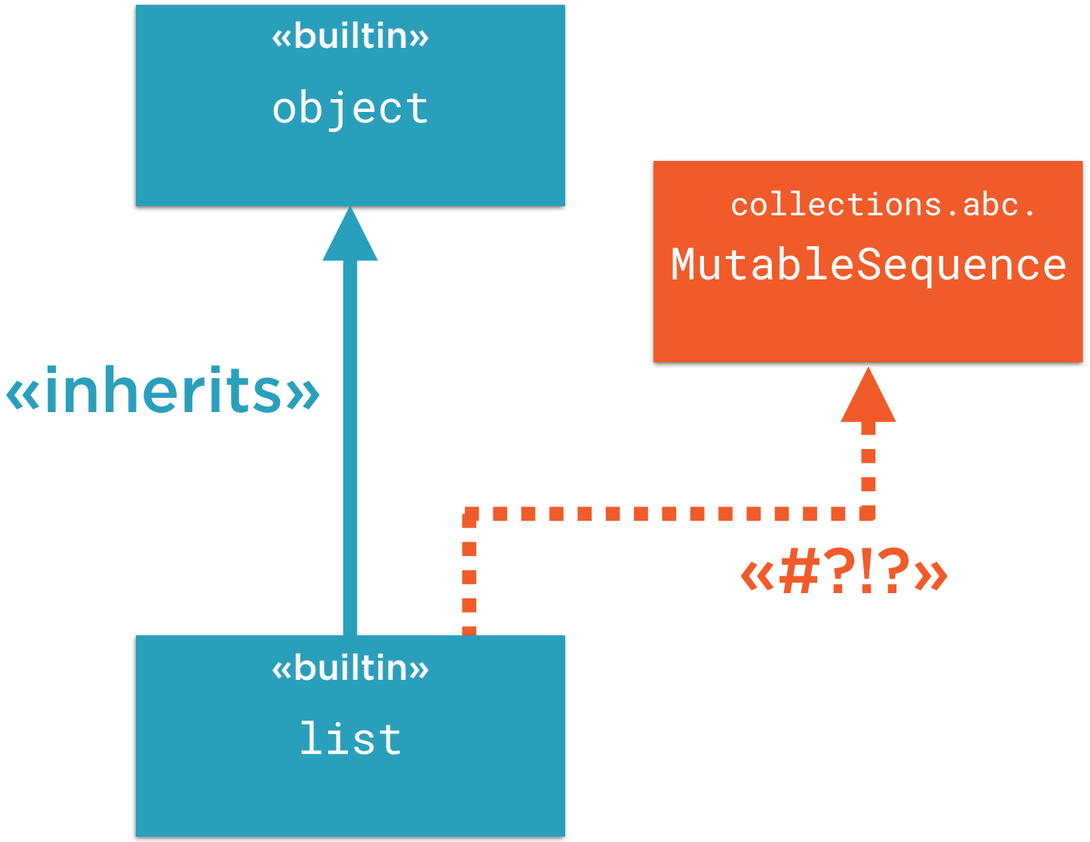
«builtin»

`list`

Python Is Not Java, C#, C++, *etc*.



Python Is Not Java, C#, C++, *etc*.



«abstract»
MutableSequence

```
__contains__(item) → bool
__delitem__(index)
__iadd__(iterable)
__iter__() → iterator
__getitem__() → object
__len__() → int
__reversed__() → iterator
__setitem__() → object
append(item)
count(item) → int
extend(iterable)
index(item) → int
insert(index, item)
pop() → object
remove(item)
reverse()
```

A conforming MutableSequence must implement all **16** methods…

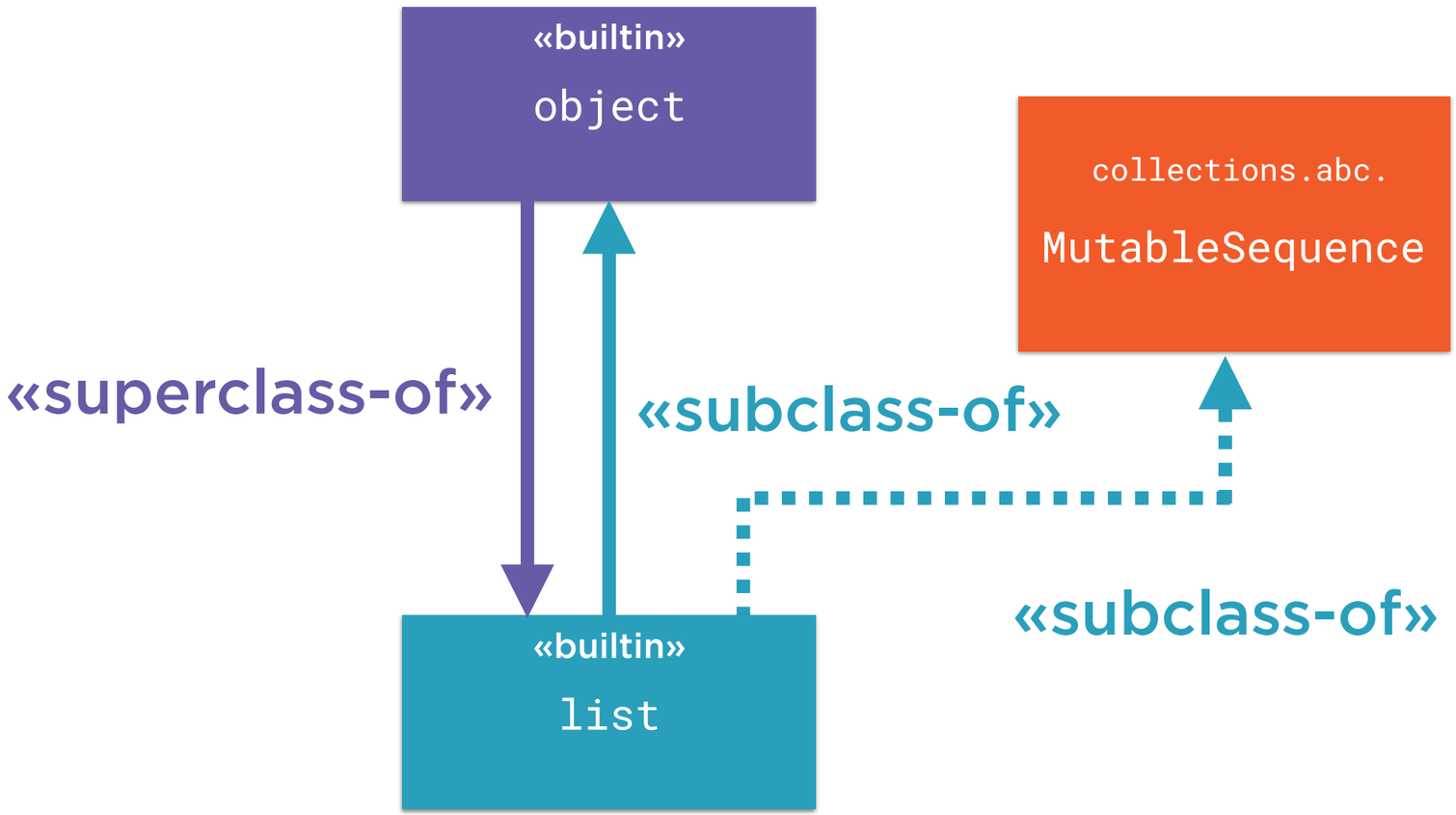
«abstract»
MutableSequence

```
__contains__(item) → bool
__delitem__(index)
__iadd__(iterable)
__iter__() → iterator
__getitem__() → object
__len__() → int
__reversed__() → iterator
__setitem__() → object
append(item)
count(item) → int
extend(iterable)
index(item) → int
insert(index, item)
pop() → object
remove(item)
reverse()
```

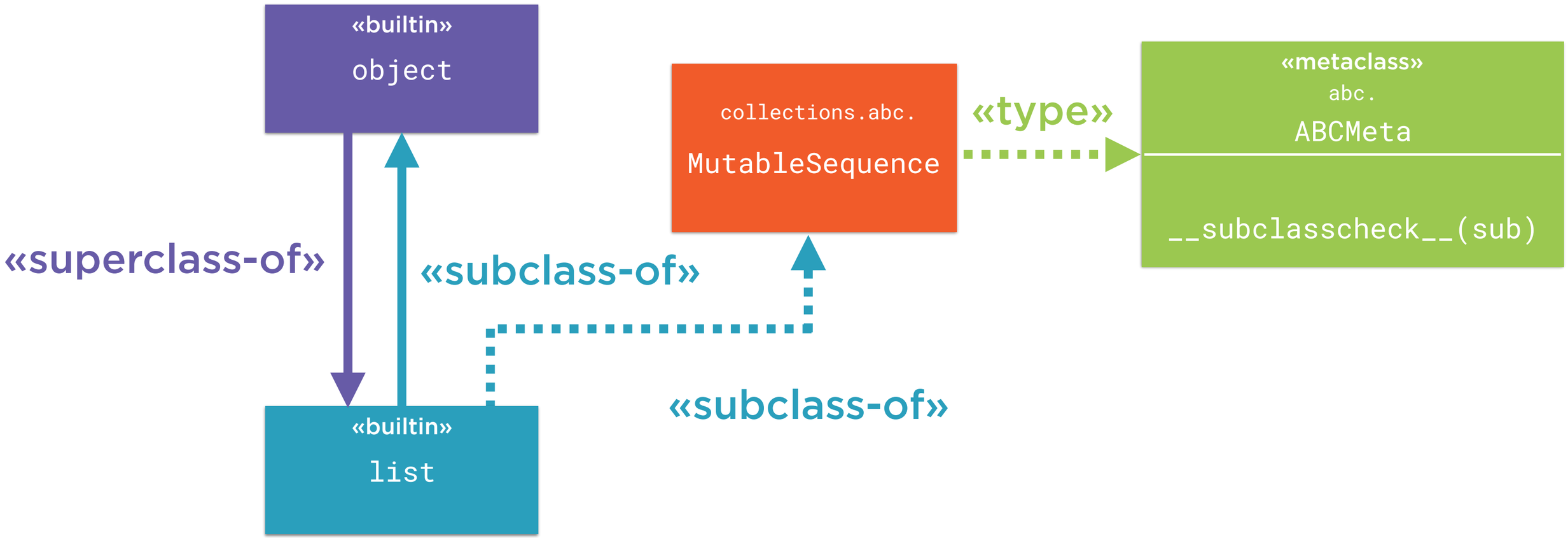
A conforming MutableSequence must implement all **16** methods…

…but the base class provides rudimentary implementations of many in terms of just **5**.

issubclass(list, MutableSequence)

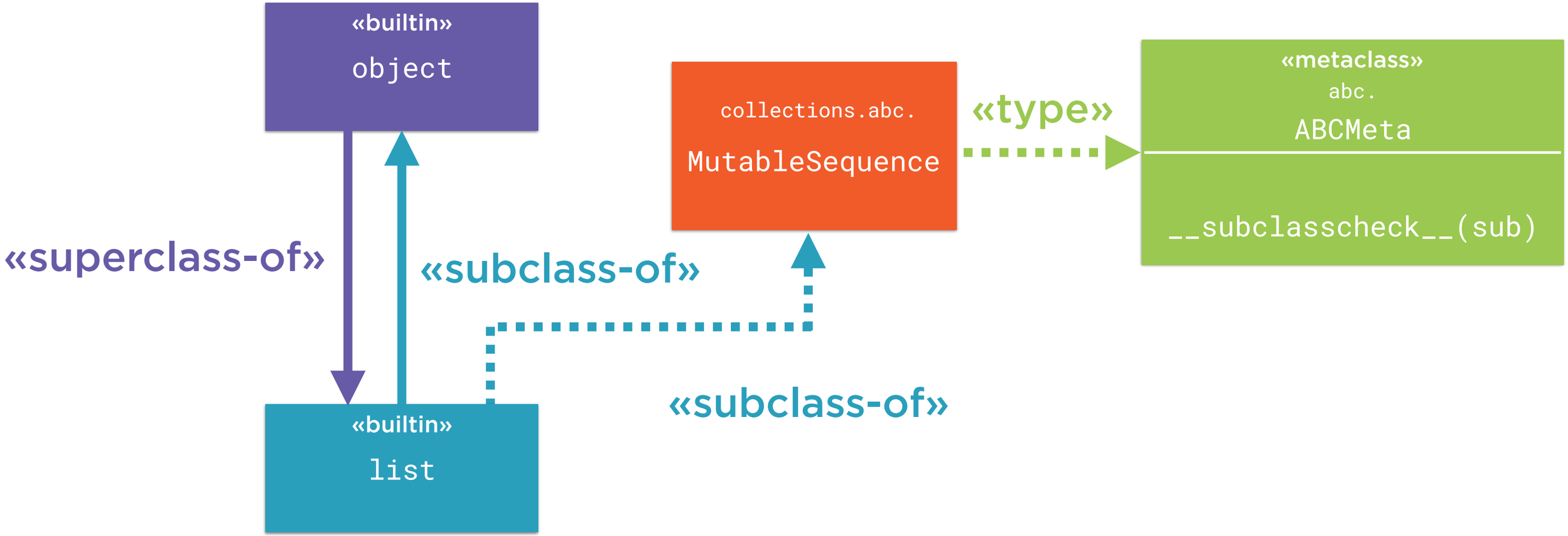


issubclass(list, MutableSequence)



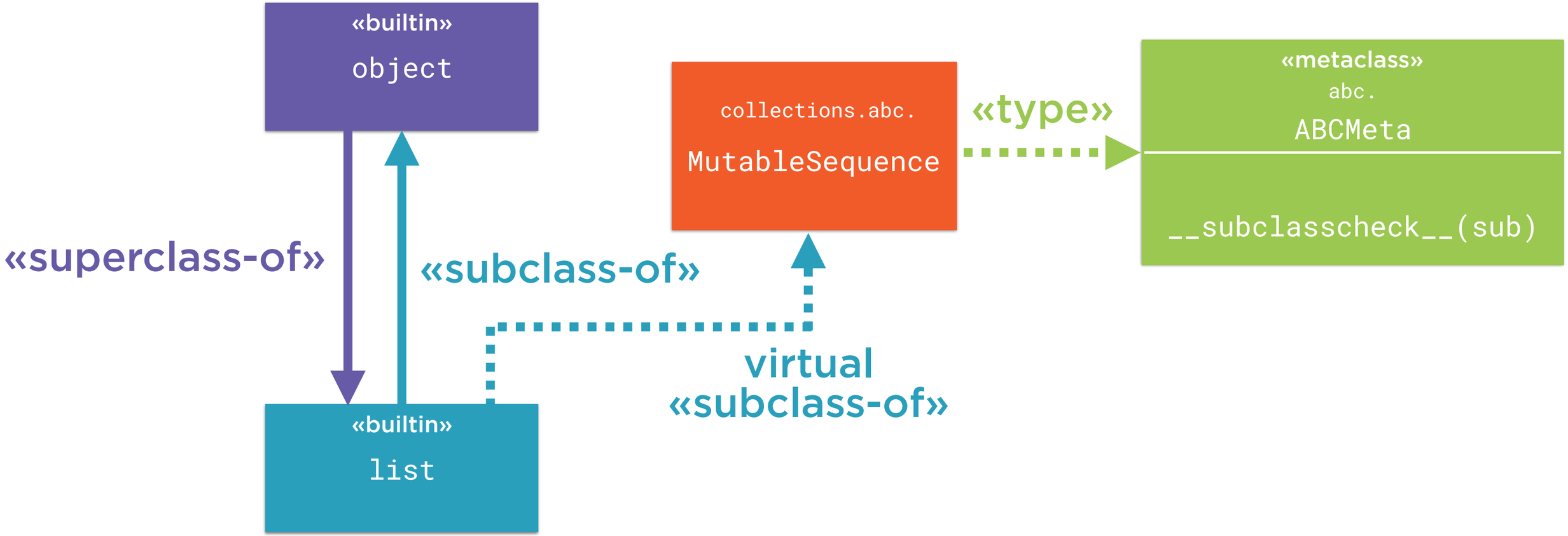
issubclass(list, MutableSequence)

```
if hasattr(type(MutableSequence), '__subclasscheck__'):
    return type(MutableSequence).__subclasscheck__(list)
# normal issubclass() behaviour...
```



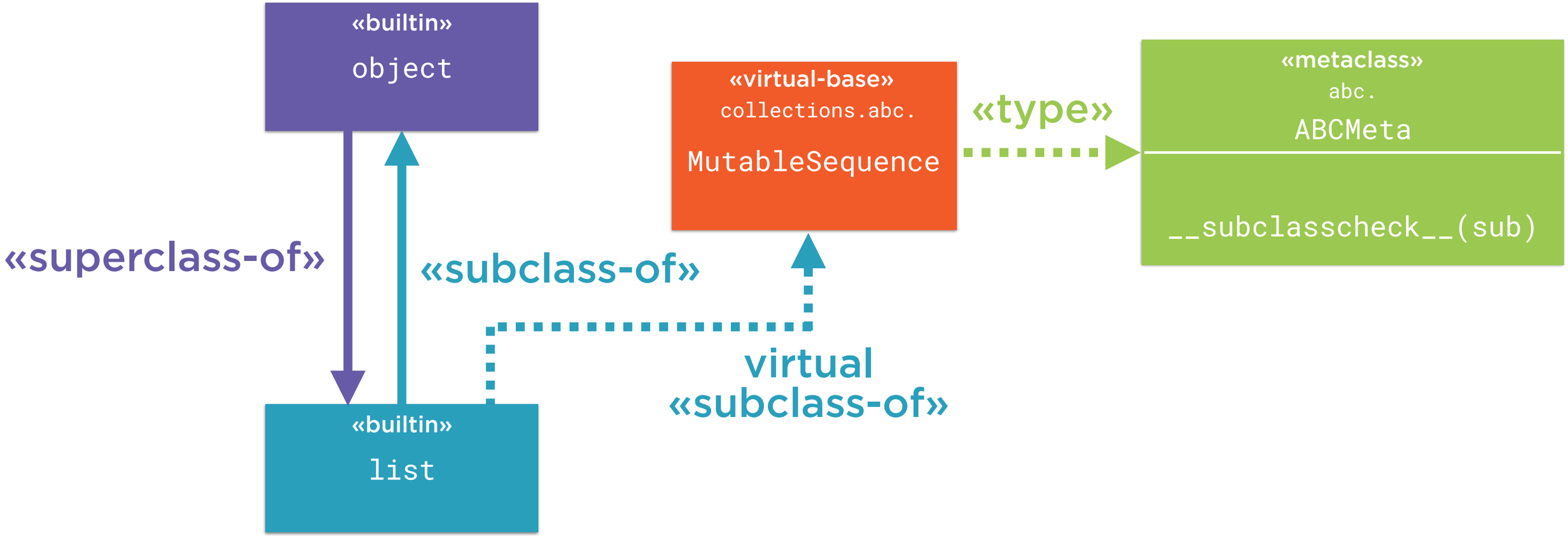
issubclass(list, MutableSequence)

```
if hasattr(type(MutableSequence), '__subclasscheck__'):
    return type(MutableSequence).__subclasscheck__(list)
# normal issubclass() behaviour...
```



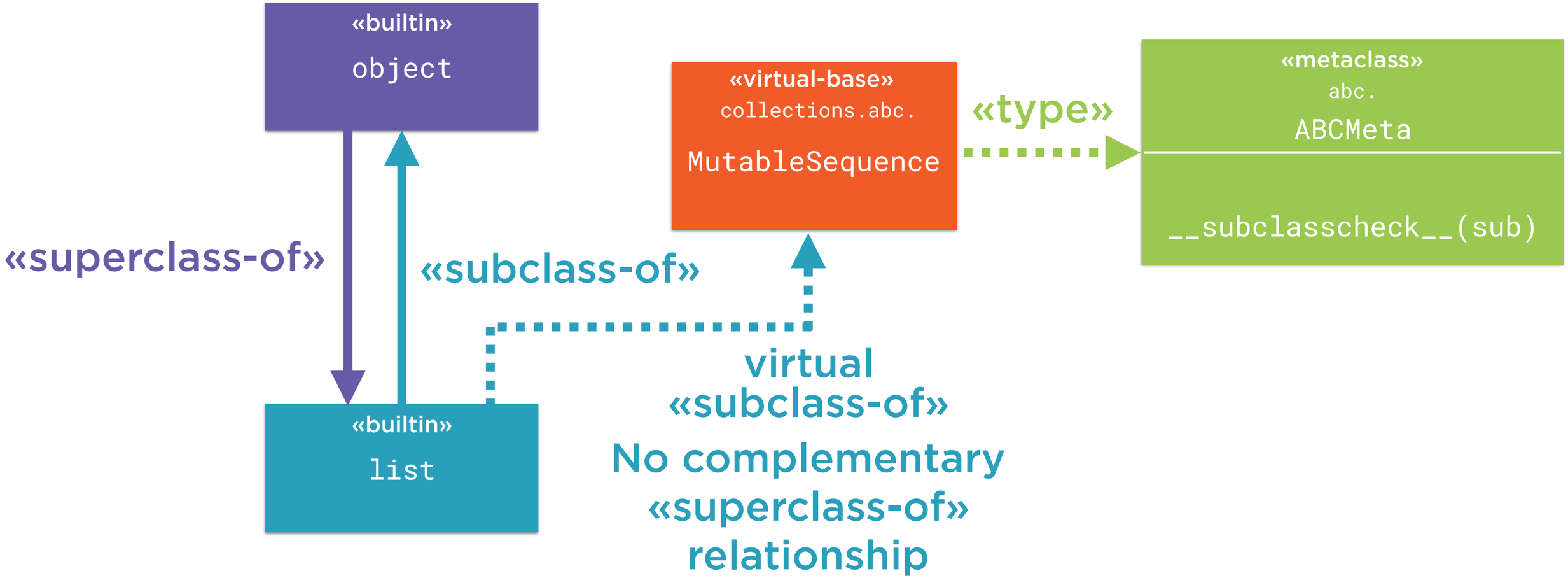
issubclass(list, MutableSequence)

```
if hasattr(type(MutableSequence), '__subclasscheck__'):
    return type(MutableSequence).__subclasscheck__(list)
# normal issubclass() behaviour...
```



issubclass(list, MutableSequence)

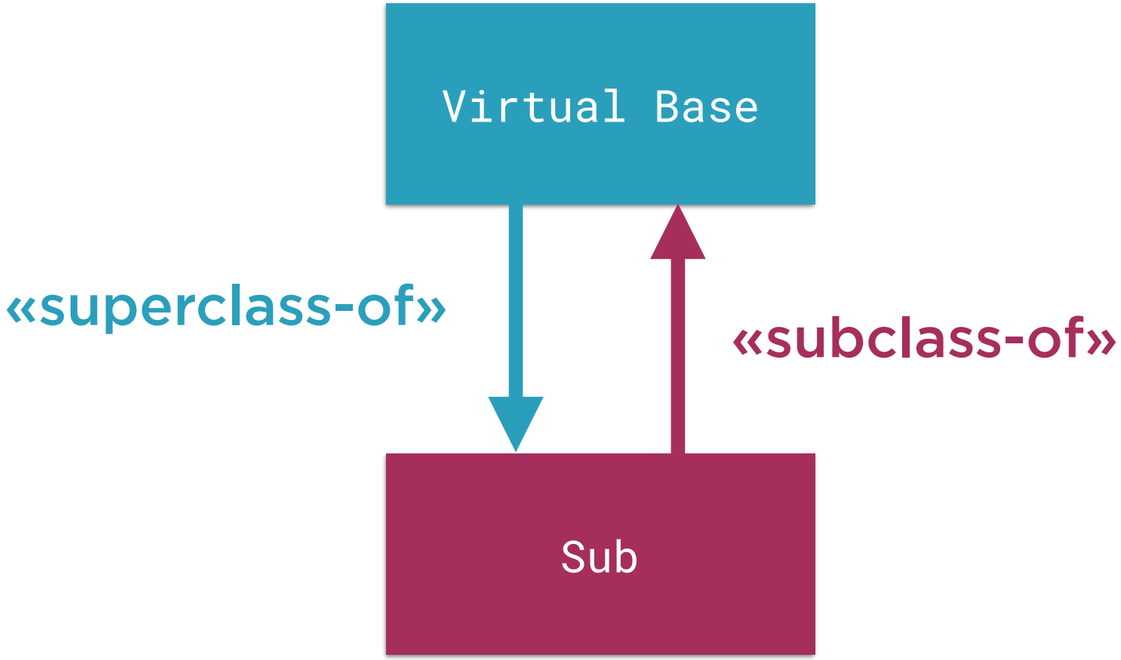
```
if hasattr(type(MutableSequence), '__subclasscheck__'):
    return type(MutableSequence).__subclasscheck__(list)
# normal issubclass() behaviour...
```



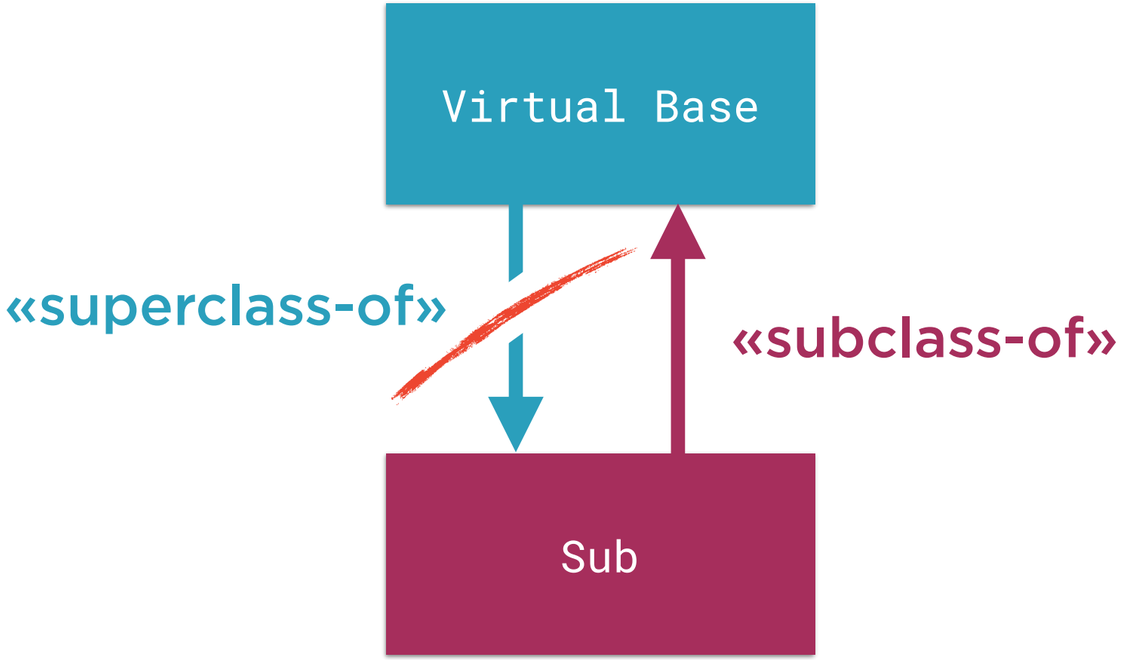
Abstract Base Classes in Practice

Non-transitive Subclass Relationships

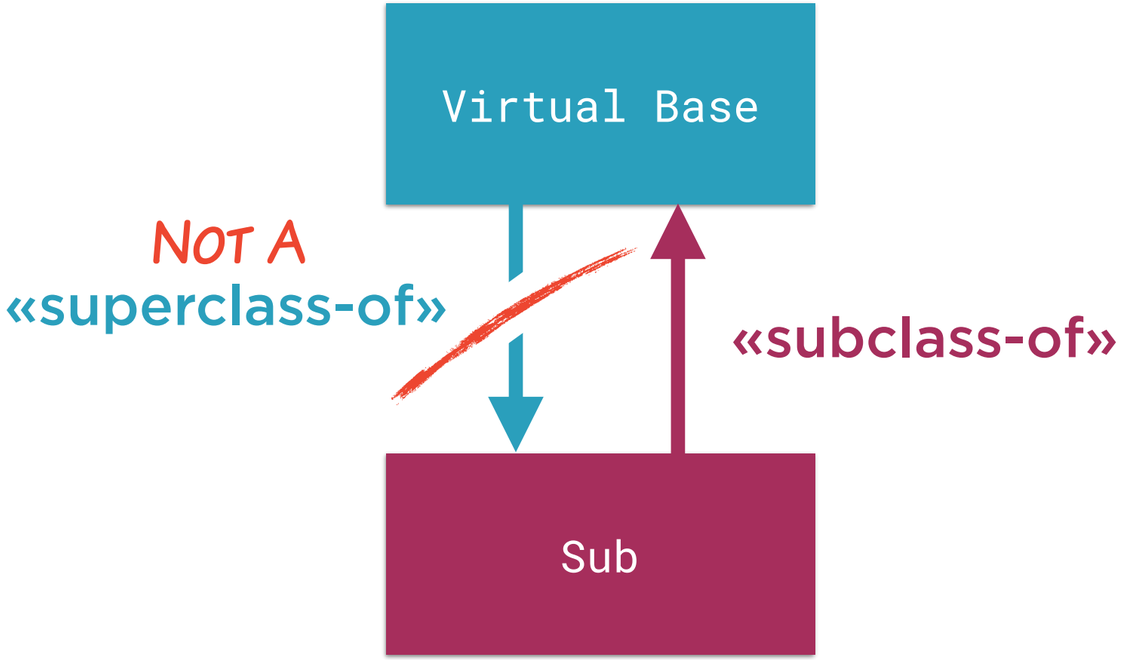
Non-transitive Subclass Relationships



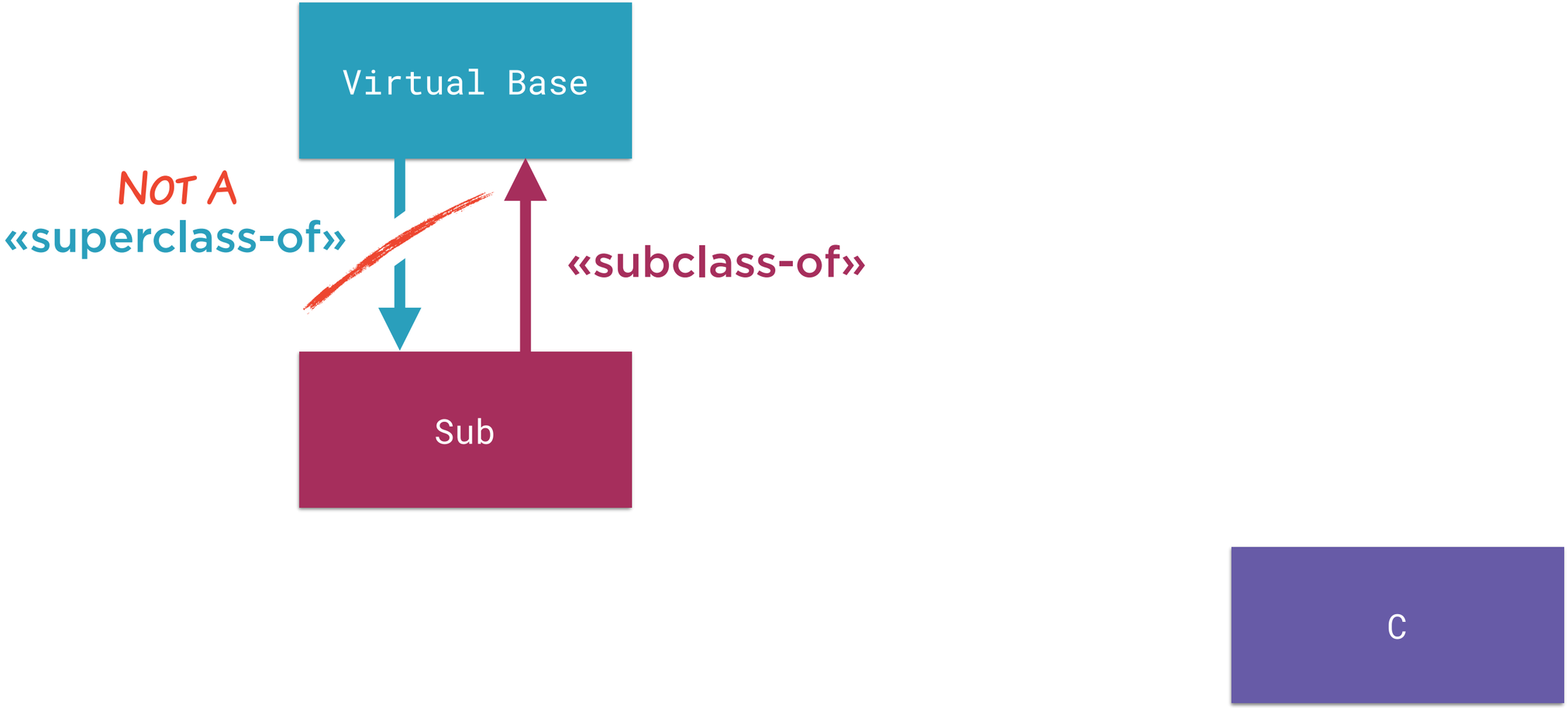
Non-transitive Subclass Relationships



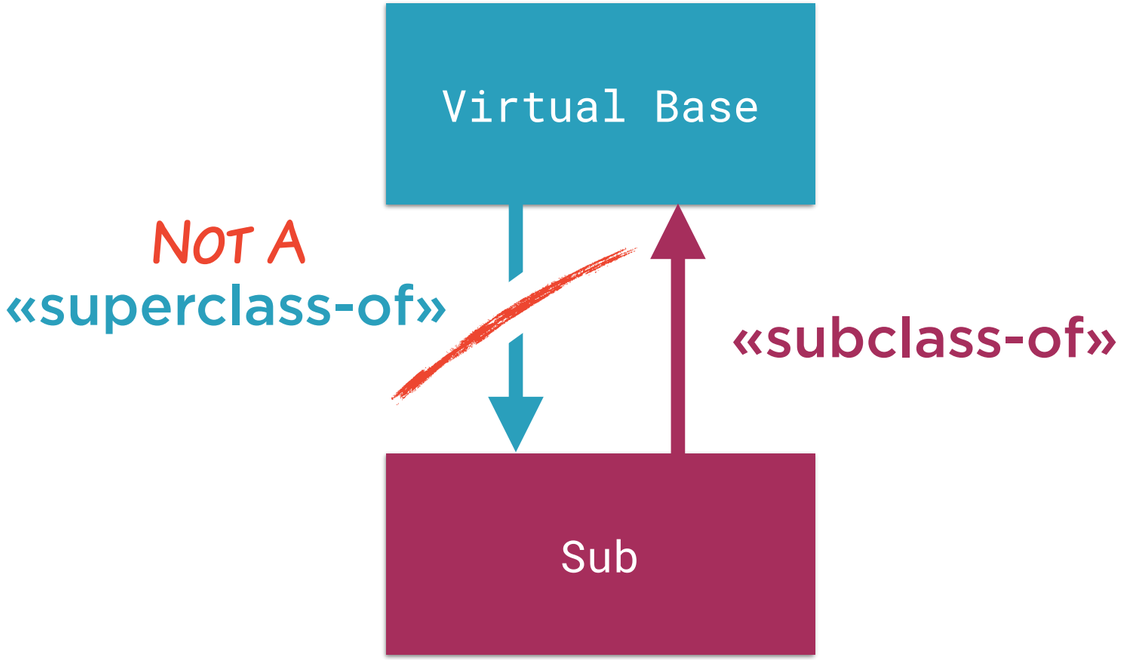
Non-transitive Subclass Relationships

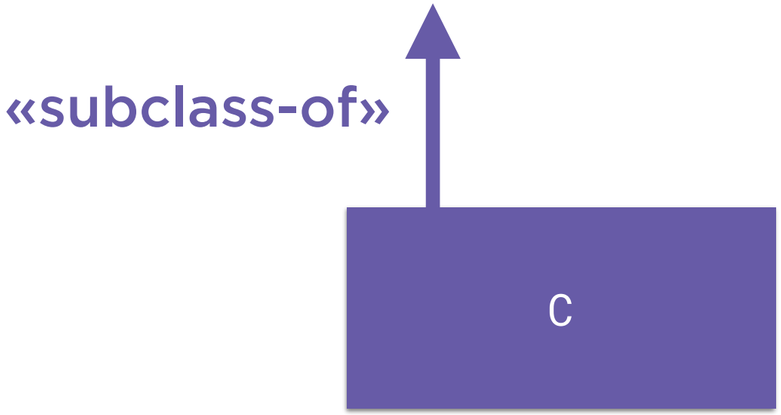


Non-transitive Subclass Relationships

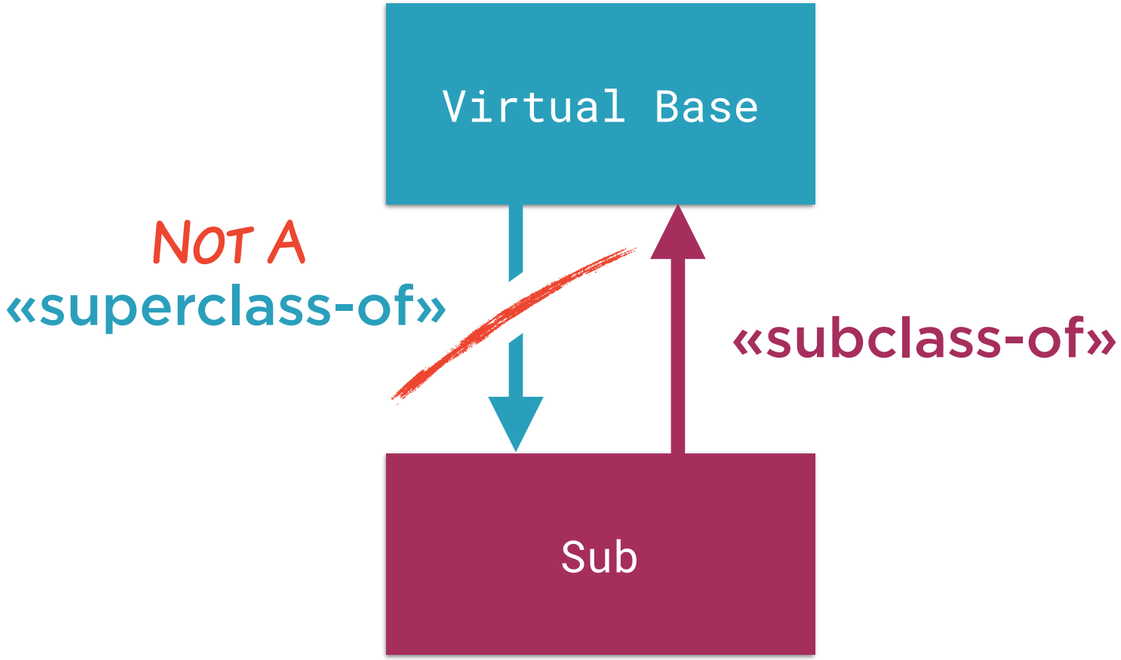


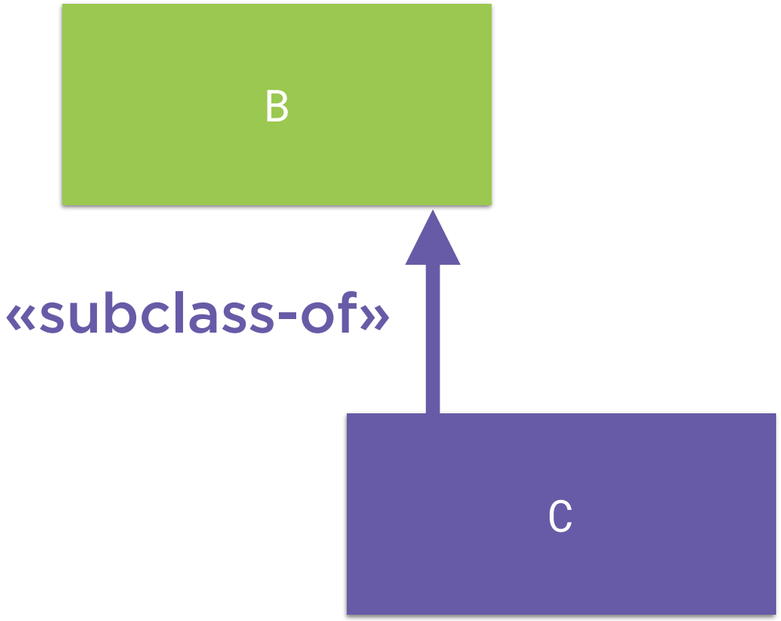
Non-transitive Subclass Relationships



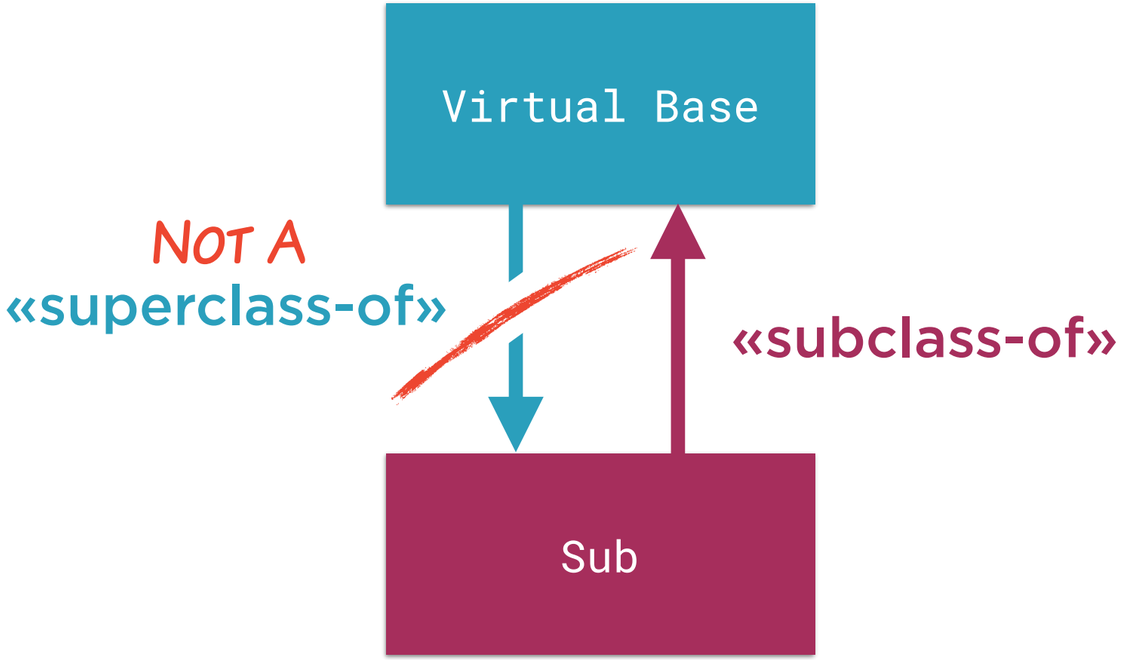


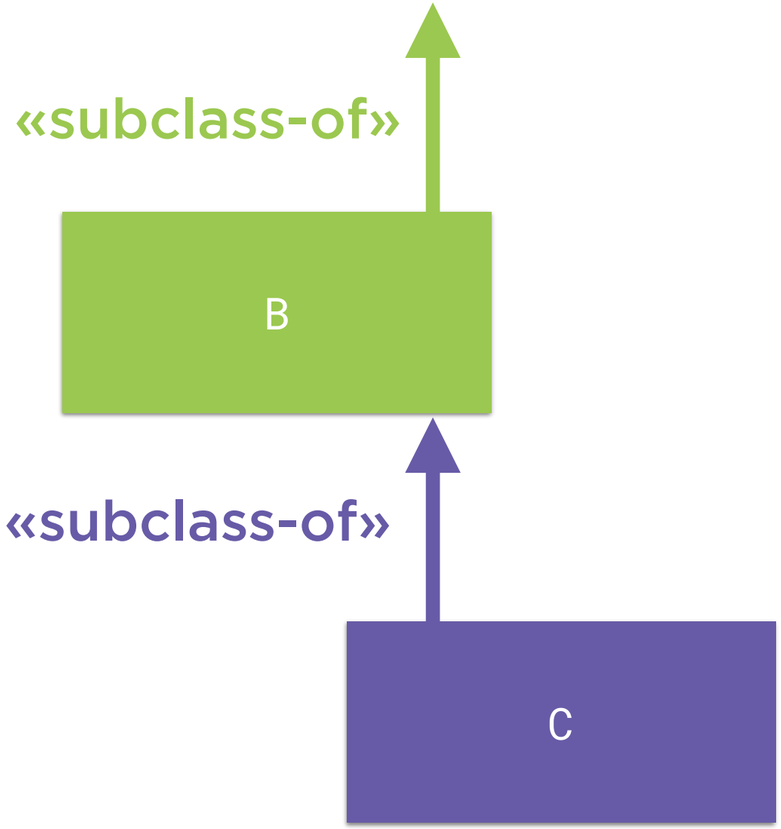
Non-transitive Subclass Relationships



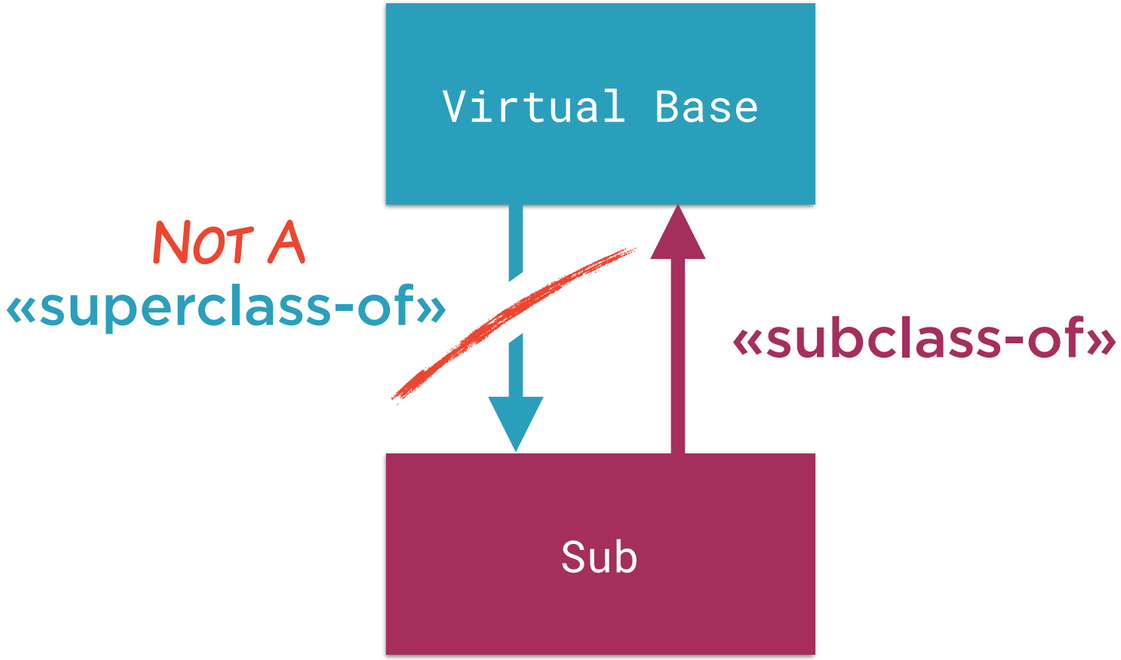


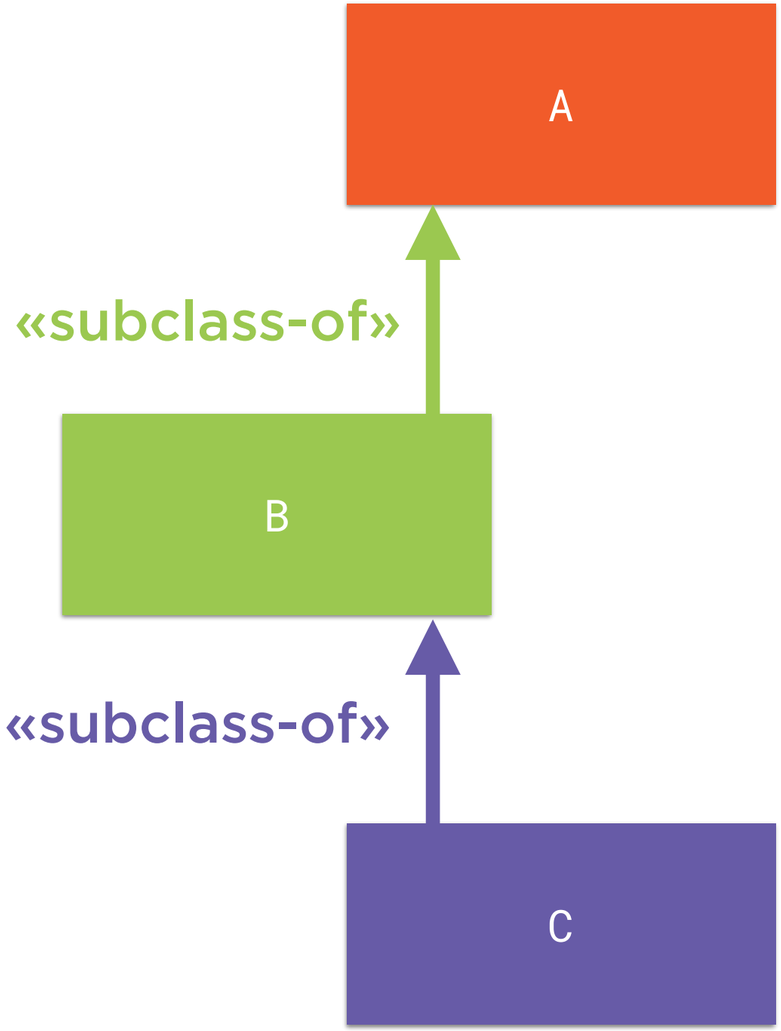
Non-transitive Subclass Relationships



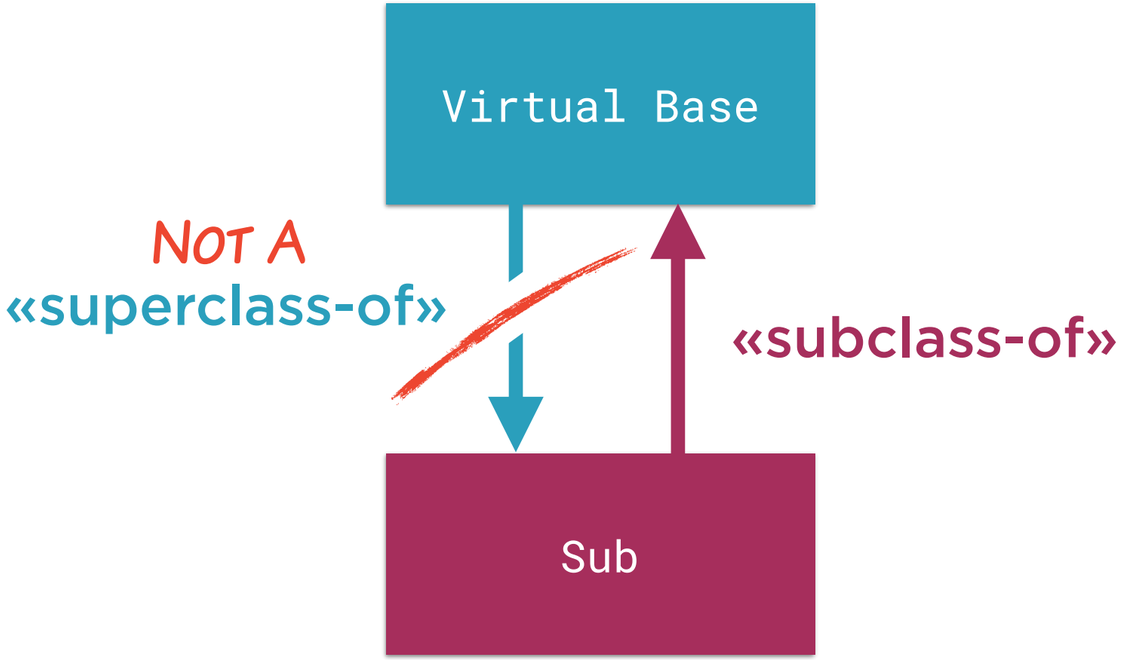


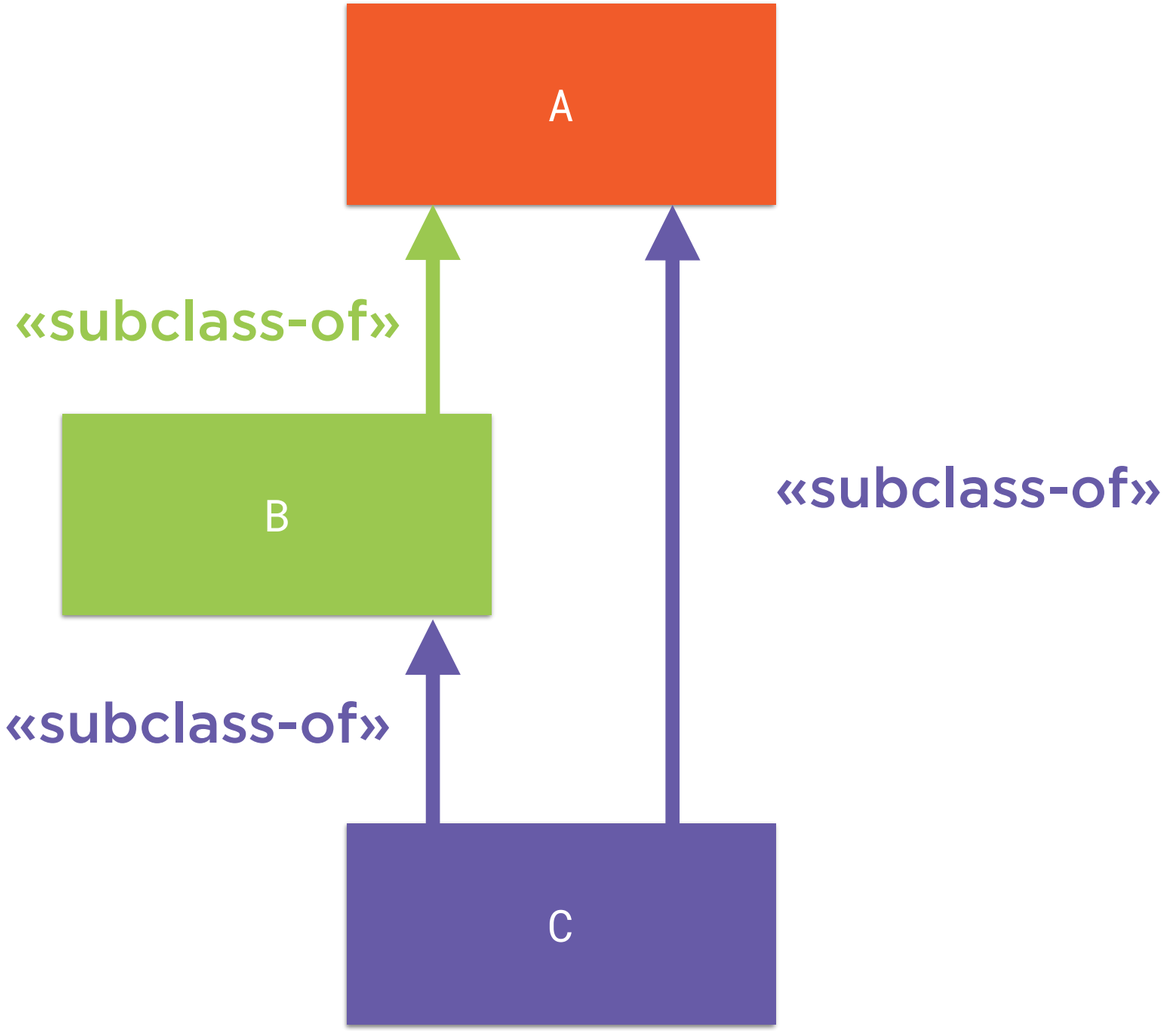
Non-transitive Subclass Relationships



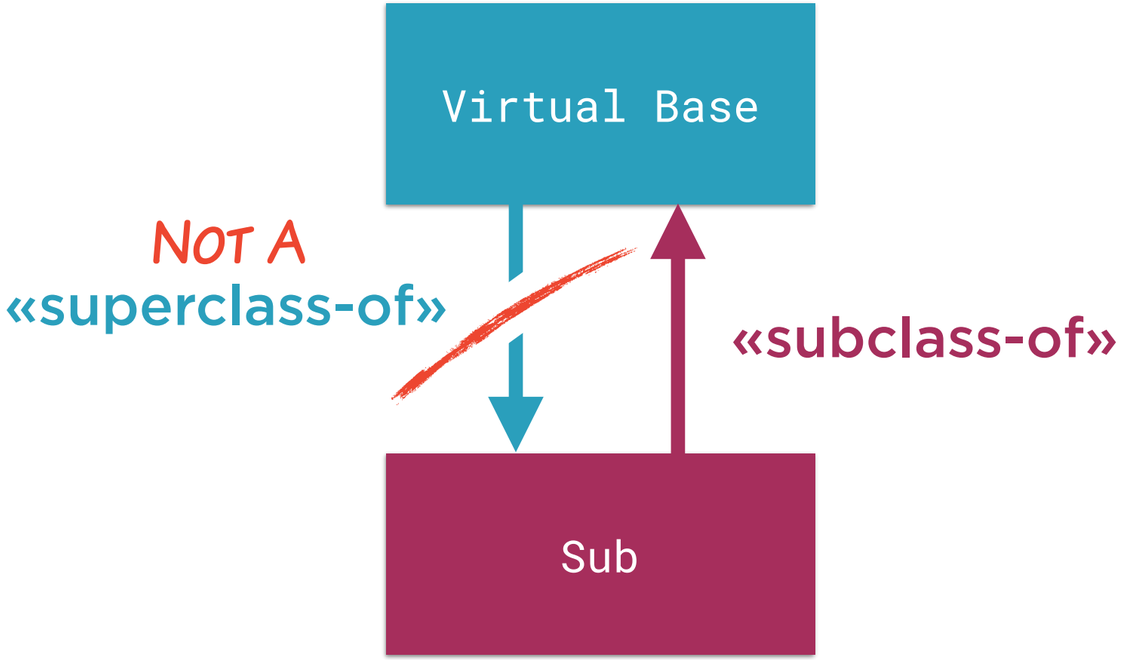


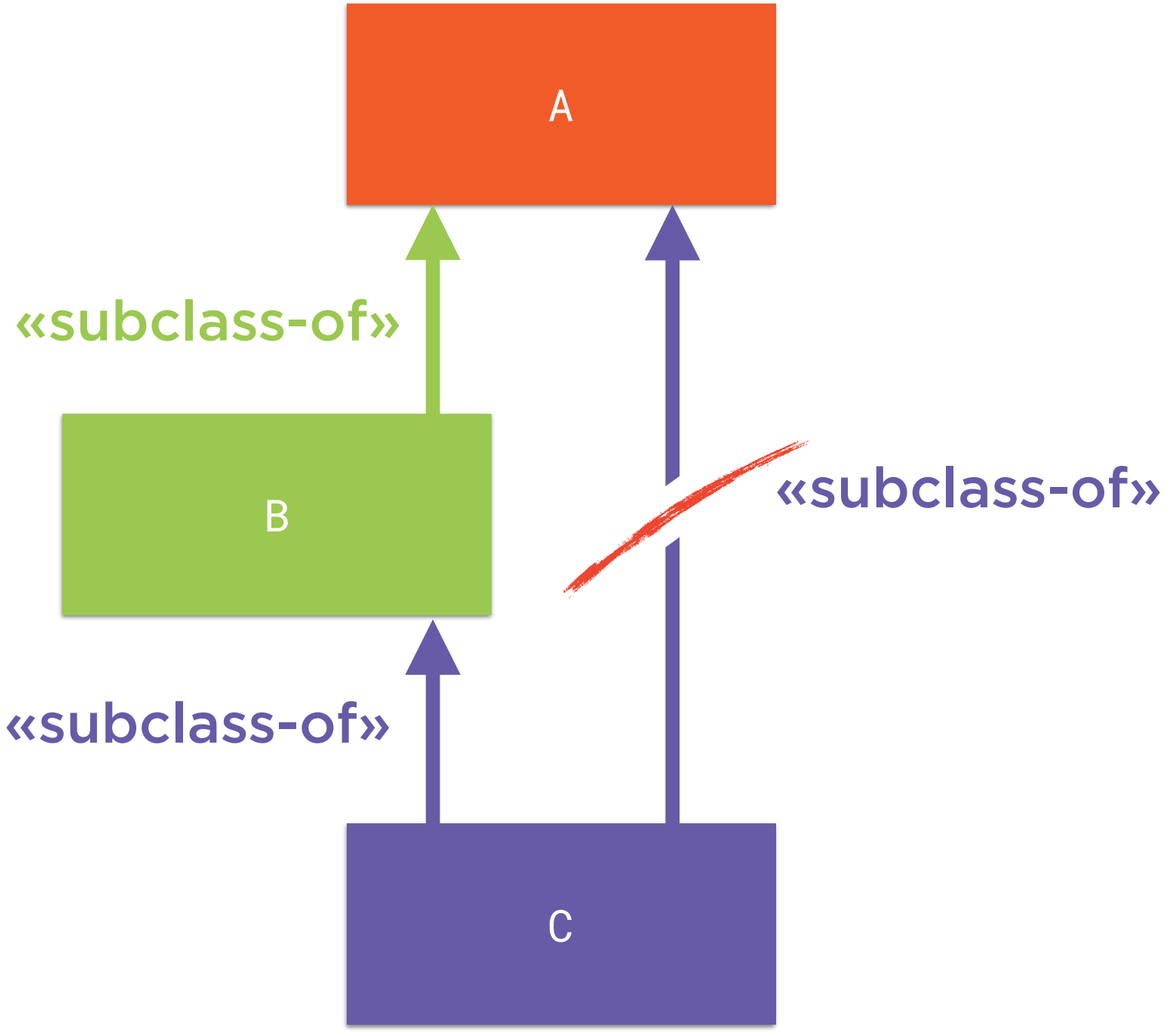
Non-transitive Subclass Relationships



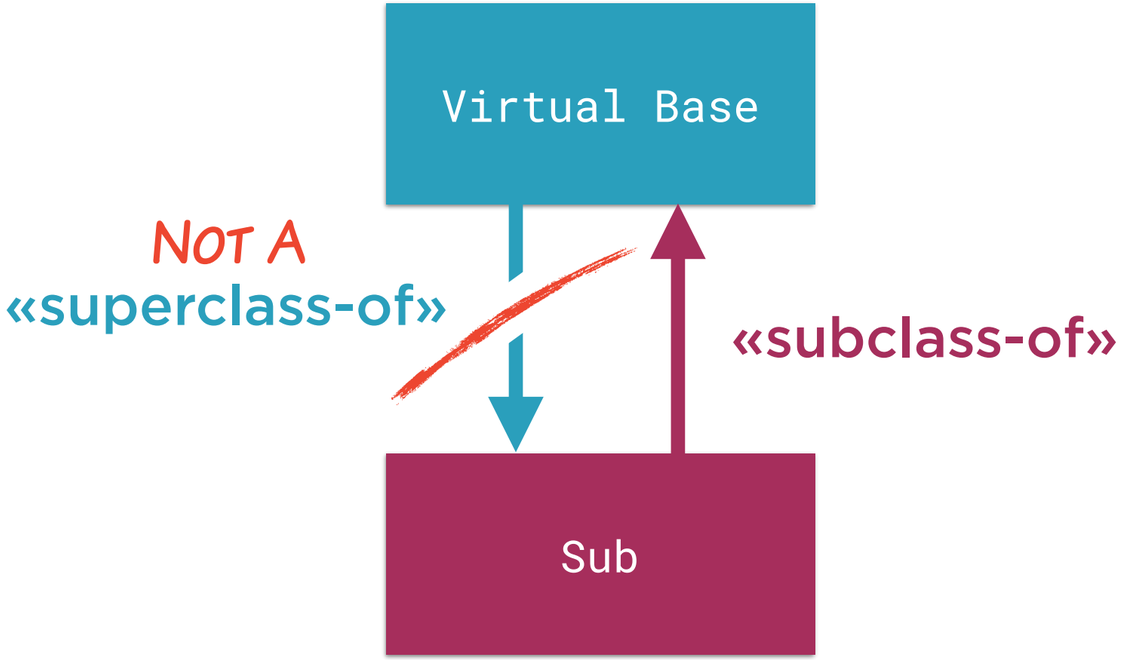


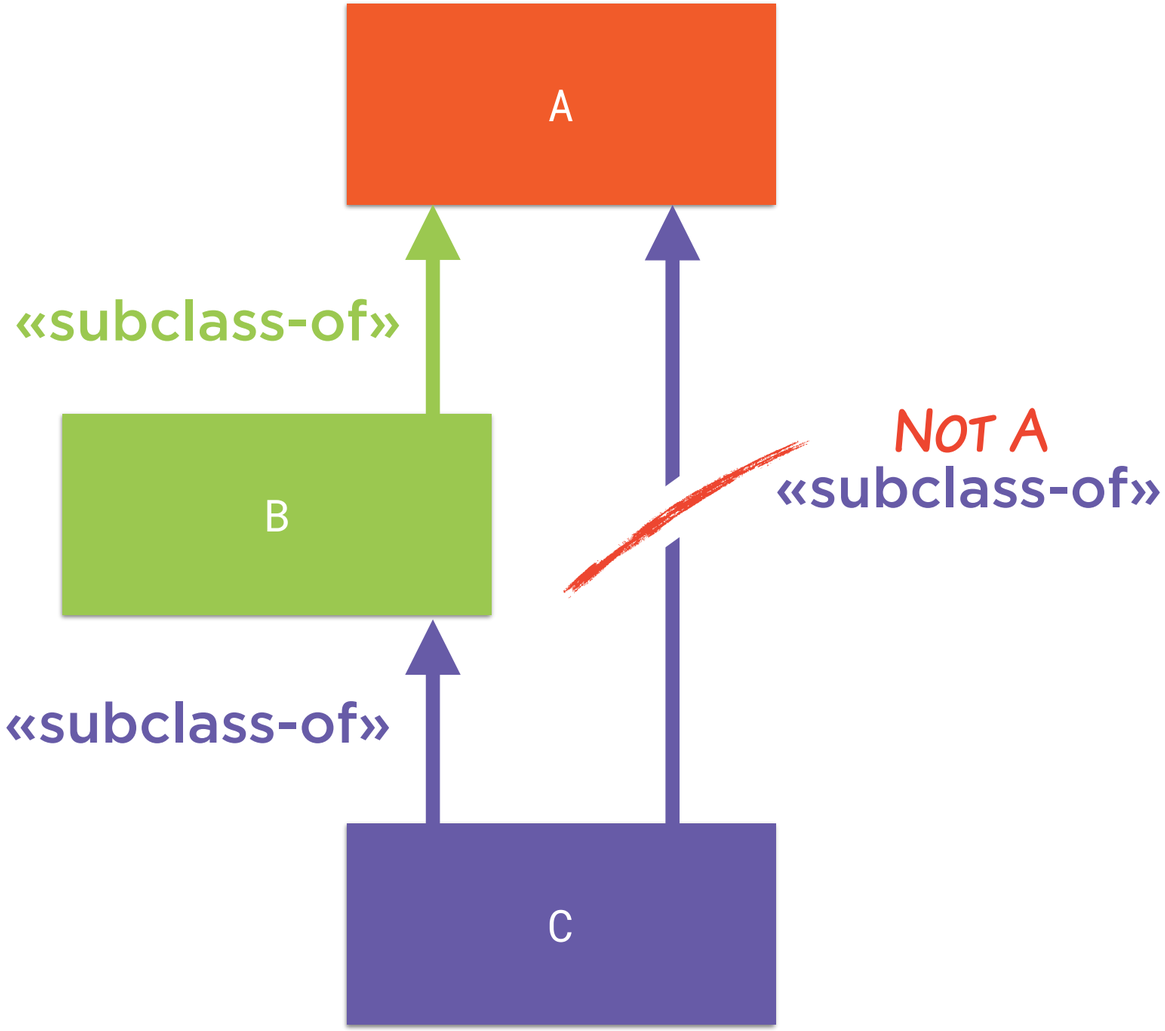
Non-transitive Subclass Relationships



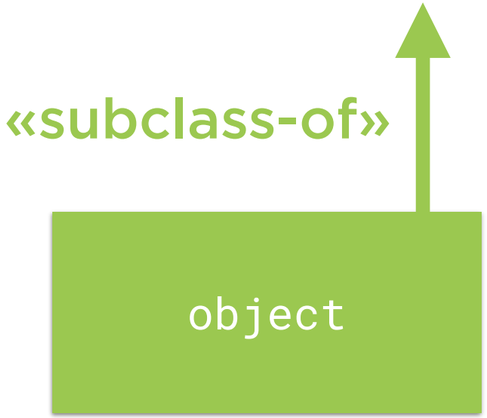


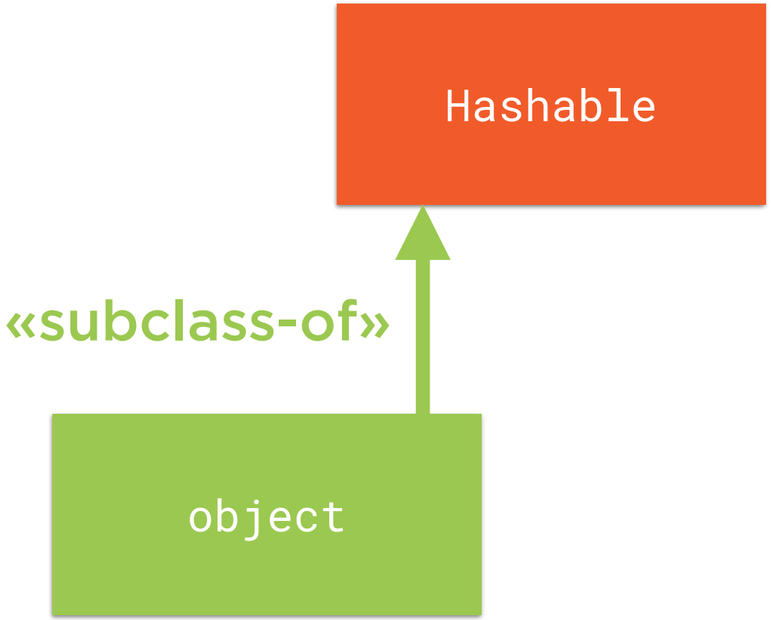
Non-transitive Subclass Relationships

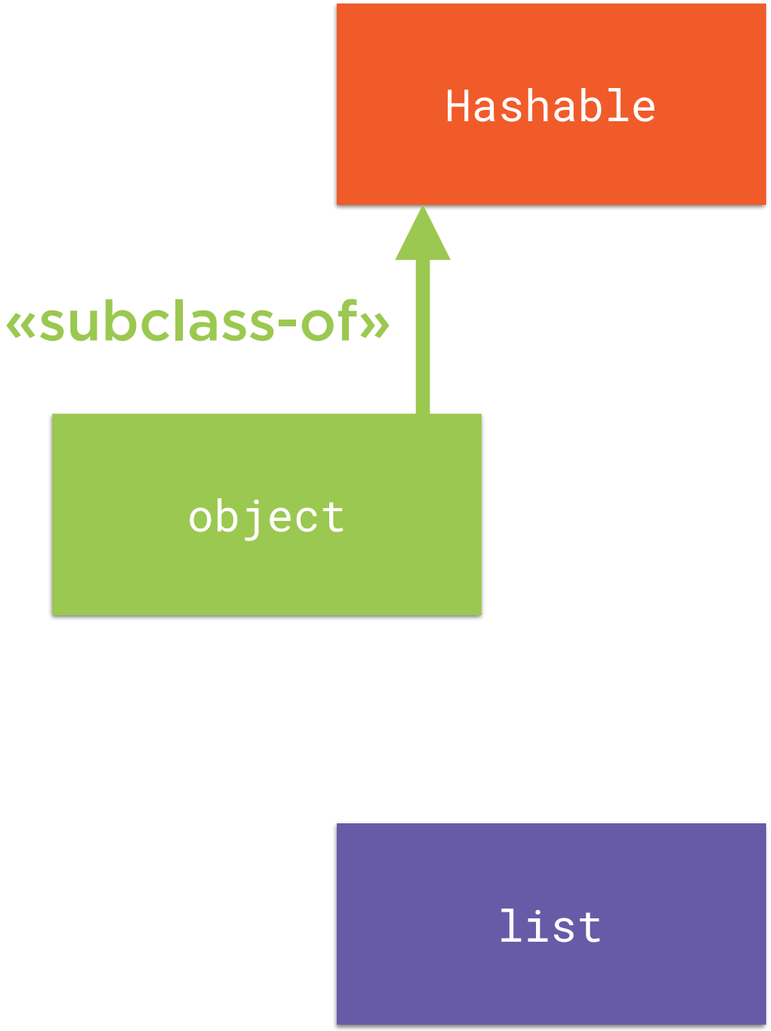


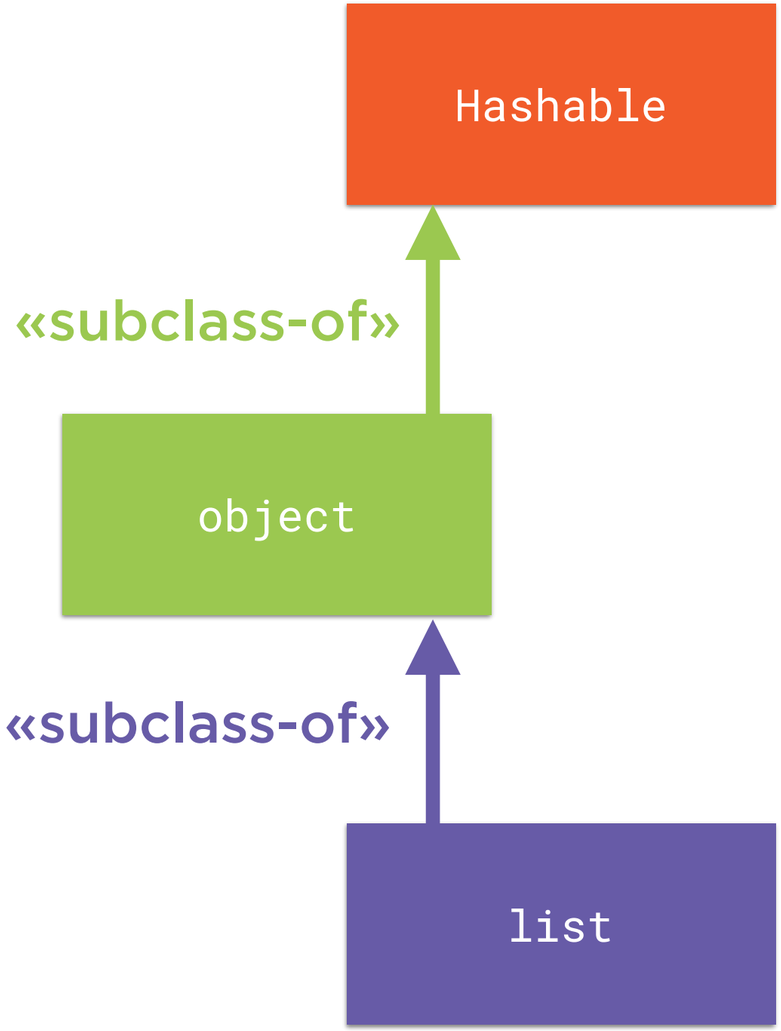


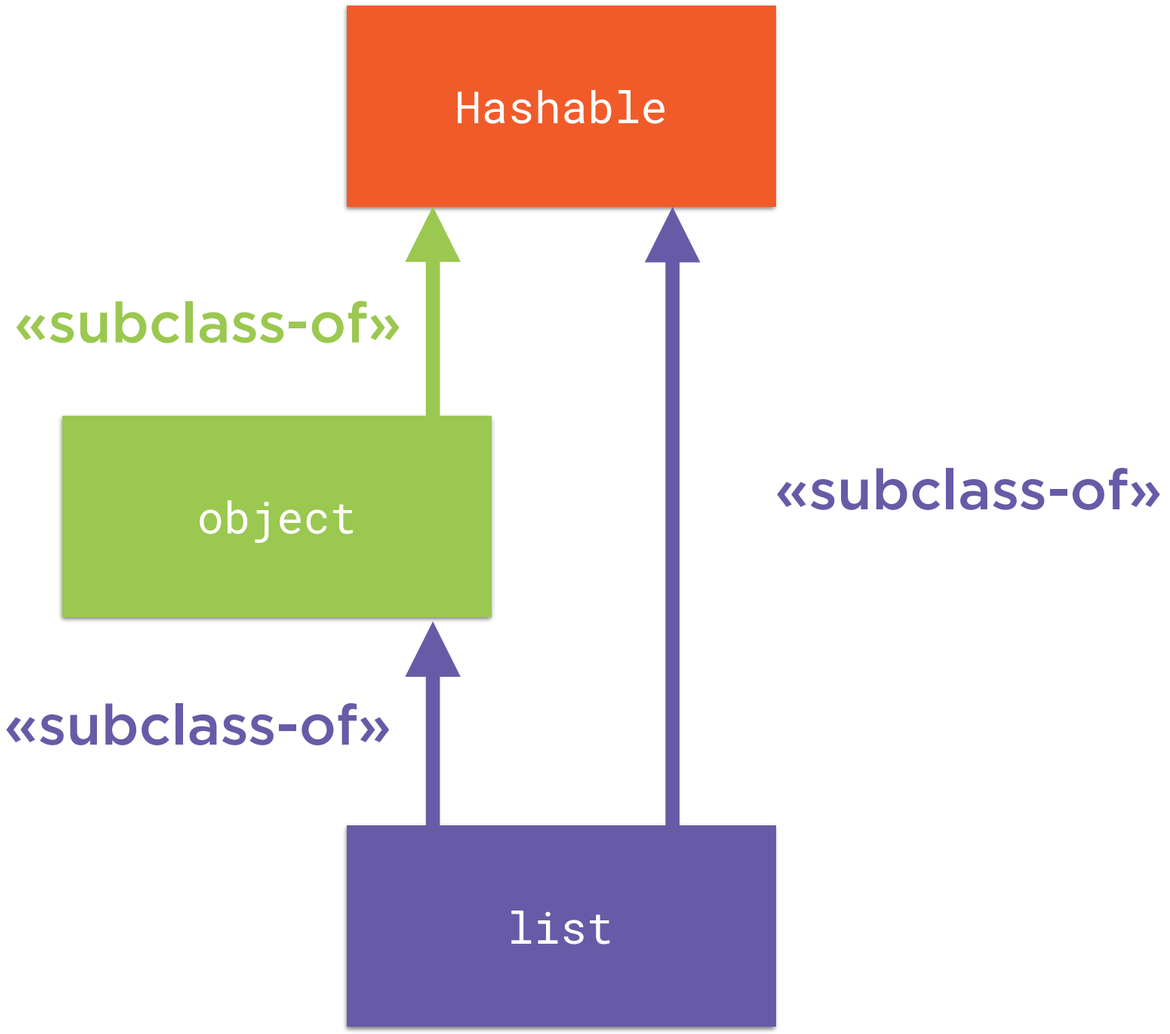
object

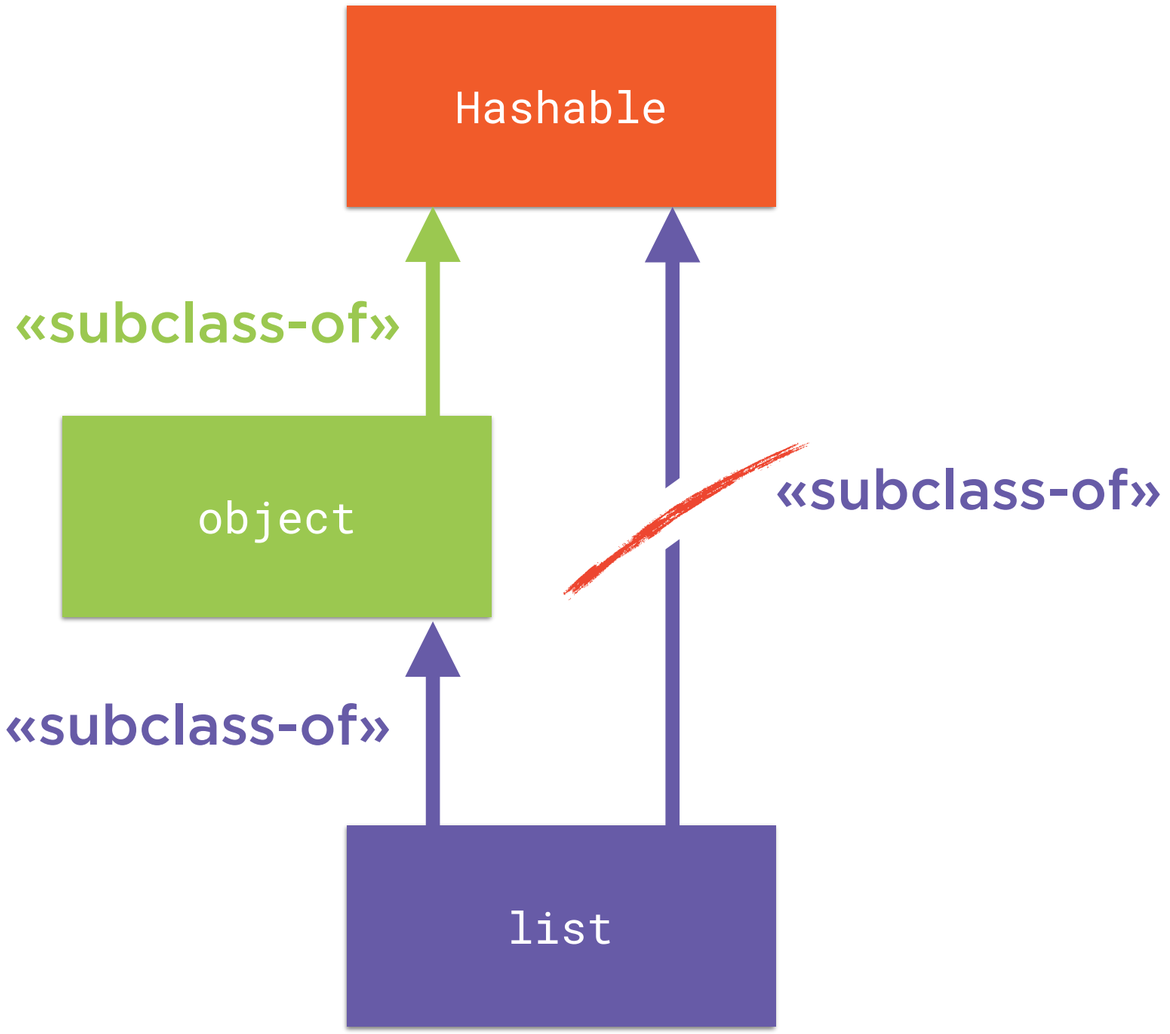

«subclass-of»
object

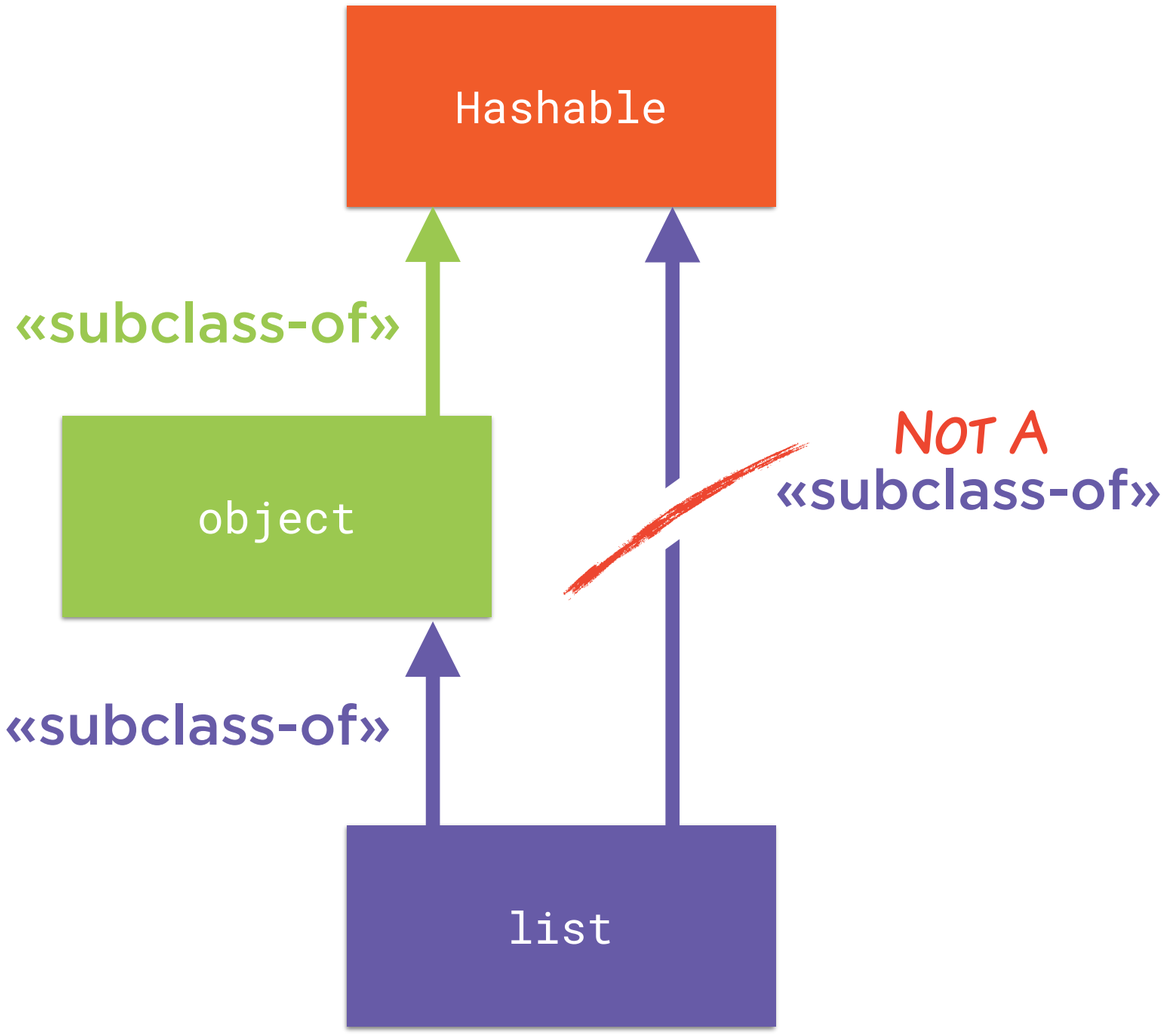

Hashable
«subclass-of»
object


Hashable
«subclass-of»
object
list


Hashable
«subclass-of»
object
«subclass-of»
list


Hashable
«subclass-of»
object
«subclass-of»
«subclass-of»
list


Hashable
«subclass-of»
object
«subclass-of»
«subclass-of»
list


Hashable
«subclass-of»
object
NOT A
«subclass-of»
«subclass-of»
list

Method Resolution With Virtual Base Classes

Library Support for Abstract Base Classes

Standard Library Support for ABCs

Custom Metaclass

`__subclasscheck__()`

`__instancecheck__()`

Tricky to get right

Onerous to implement a metaclass

Standard Library

`abc` **module**

- `ABCMeta` **metaclass**
- `ABC` **base class**
- `@abstractmethod` **decorator**

ABCMeta

ABCMeta

Reliable implementations of

ABCMeta

Reliable implementations of

`__subclasscheck__()`

ABCMeta

Reliable implementations of

`__subclasscheck__()`

`__instancecheck__()`

ABCMeta

Reliable implementations of

`__subclasscheck__()`

`__instancecheck__()`

in terms of

ABCMeta

Reliable implementations of

`__subclasscheck__()`

`__instancecheck__()`

in terms of

`__subclasshook__()`

ABCMeta

Reliable implementations of

`__subclasscheck__()`

`__instancecheck__()`

in terms of

`__subclasshook__()`

ABCMeta

```
ABCMeta.__instancecheck__(sub)
```

and

```
ABCMeta.__subclasscheck__(sub)
```

delegate to

```
Base.__subclasshook__(sub)
```

ABCMeta

```
Base.__subclasshook__(sub)
```

ABCMeta

```
Base.__subclasshook__(sub)
```

True

ABCMeta

```
Base.__subclasshook__(sub)
```

True

False

ABCMeta

```
Base.__subclasshook__(sub)
```

True

False

NotImplemented

ABCMeta

```
Base.__subclasshook__(sub)
```

True

False

NotImplemented : to lookup via MRO

ABCMeta

```
Base.__subclasshook__(sub)
```

True : when sub is a subclass of Base

False : when sub is not a subclass of Base

NotImplemented : to lookup via MRO

Virtual Subclass Registration

Virtual Subclass Registration

Virtual Subclass Registration

Register a class as a *virtual* subclass

Virtual Subclass Registration

Register a class as a *virtual* subclass

Base metaclass must be `ABCMeta`

Virtual Subclass Registration

Register a class as a *virtual* subclass

Base metaclass must be `ABCMeta`

Call `register(sub)` **metamethod**

Virtual Subclass Registration

Register a class as a *virtual* subclass

Base metaclass must be `ABCMeta`

Call `register(sub)` metamethod

Call to `register()` returns its argument

Virtual Subclass Registration

Register a class as a *virtual* subclass

Base metaclass must be `ABCMeta`

Call `register(sub)` metamethod

Call to `register()` returns its argument

Can use `@register` as a class-decorator

Combining Subclass Detection and Registration

Combining Subclass Detection and Registration

Combining Subclass Detection and Registration

You can use both subclass `register()` **and** `__subclasshook__()`

Combining Subclass Detection and Registration

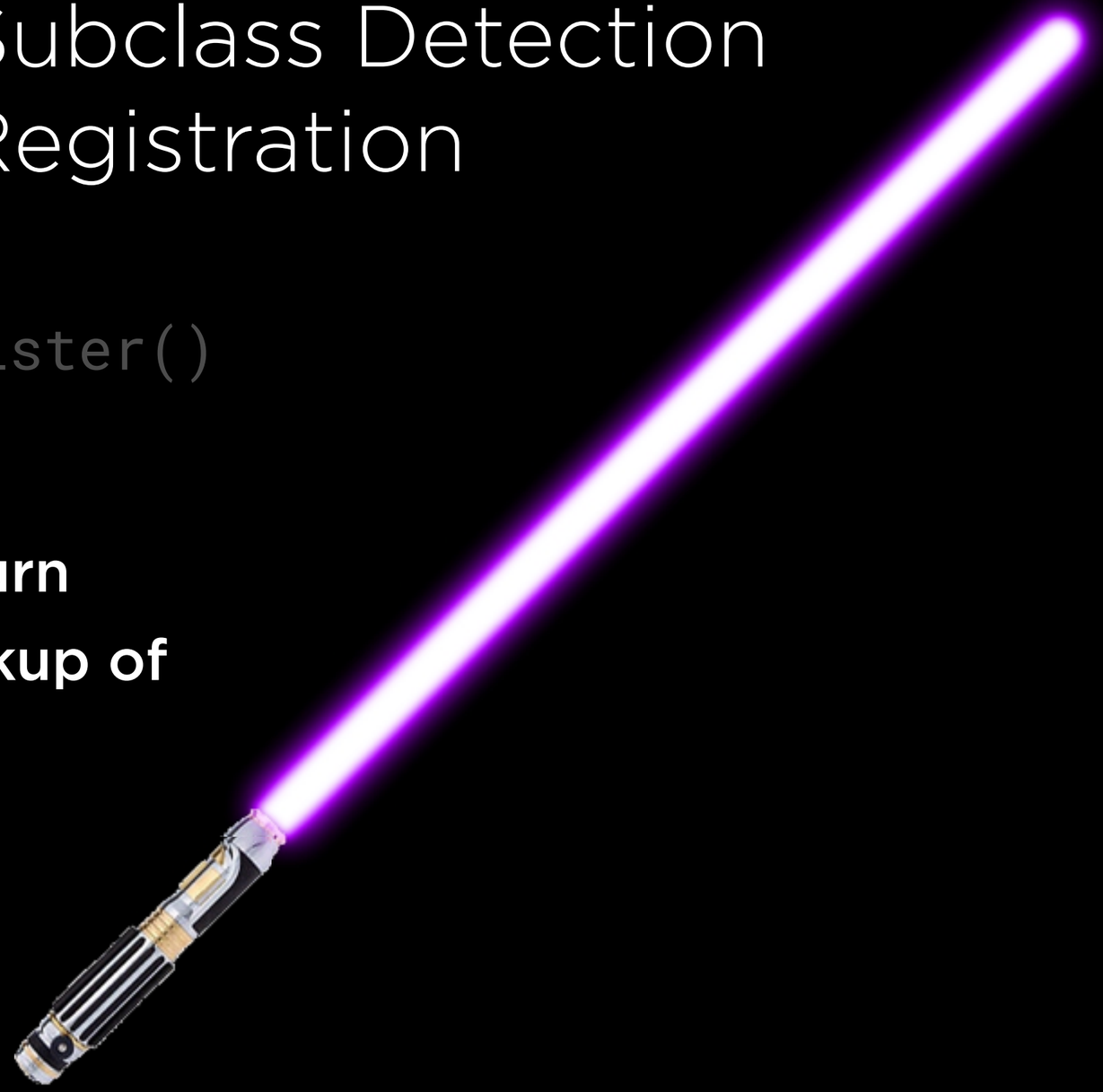
You can use both subclass `register()` and `__subclasshook__()`

`__subclasshook__()` must return `NotImplemented` to trigger lookup of registered subclasses.

Combining Subclass Detection and Registration

You can use both subclass `register()` and `__subclasshook__()`

`__subclasshook__()` must return `NotImplemented` to trigger lookup of registered subclasses.



The ABC Convenience Base Class

The ABC Base Class – In Its Entirety!

```
class ABC(metaclass=ABCMeta):
    """Helper class that provides a standard way to create an ABC using
    inheritance.
    """
    pass
```

Declaring Abstract Methods

Abstract Methods

Abstract Methods

Declared in Abstract Base Classes

Marked with the `@abstractmethod` decorator

Abstract Methods

Declared in Abstract Base Classes

Marked with the `@abstractmethod` decorator

Useful Definition Not Required

Python syntax requires a placeholder definition

Abstract Methods

Declared in Abstract Base Classes

Marked with the `@abstractmethod` decorator

Useful Definition Not Required

Python syntax requires a placeholder definition

Must Be Overridden in Concrete Classes

Abstract methods prevent instantiation of ABCs

Abstract Methods

```
from abc import (ABC, abstractmethod)

class AbstractBaseClass(ABC):  # metaclass is ABCMeta

    @abstractmethod
    def an_abstract_method(self):
        raise NotImplementedError  # Method body syntactically required.
```

Metaclass must be `ABCMeta` for abstractness to be enforced

Combining Method Decorators

Combining Method Decorators

`@abstractmethod`
must be innermost

`@property`
getters and setters can be independently abstract

```
class AbstractBaseClass(ABC):

    @staticmethod
    @abstractmethod
    def an_abstact_static_method():
        raise NotImplementedError

    @classmethod
    @abstractmethod
    def an_abstract_class_method(cls):
        raise NotImplementedError

    @property
    @abstractmethod
    def an_abstract_property(self):
        raise NotImplementedError

    @an_abstract_property.setter
    @abstractmethod
    def an_abstract_property(self, value):
        raise NotImplementedError
```

Abstract Descriptors

Abstract Descriptors

`__get__()`, `__set__()` and `__delete__()` can be independently abstract

`__isabstractmethod__` attribute (*e.g.* property) must evaluate to `True` if any are abstract

```
class MyDataDescriptor(ABC):

    @abstractmethod
    def __get__(self, instance, owner):
        # ...
        pass

    @abstractmethod
    def __set__(self, instance, value):
        # ...
        pass

    @abstractmethod
    def __delete__(self, instance):
        # ...
        pass

    @property
    def __isabstractmethod__(self):
        return True  # or False if not abstract
```

Improving `@invariant` with ABCs

Summary

Summary

Summary

Abstract Base Classes (ABCs)

Summary

Abstract Base Classes (ABCs)

`issubclass()` **delegates to metamethod** `__subclasscheck__()`

Summary

Abstract Base Classes (ABCs)

`issubclass()` **delegates to metamethod** `__subclasscheck__()`

`isinstance()` **delegates to metamethod** `__instancecheck__()`

Summary

Abstract Base Classes (ABCs)

`issubclass()` **delegates to metamethod** `__subclasscheck__()`

`isinstance()` **delegates to metamethod** `__instancecheck__()`

`abc` **module provides** `ABCMeta` **and** `ABC`

Summary

Abstract Base Classes (ABCs)

`issubclass()` **delegates to metamethod** `__subclasscheck__()`

`isinstance()` **delegates to metamethod** `__instancecheck__()`

`abc` **module provides** `ABCMeta` **and** `ABC`

`ABCMeta` **eases customisation with** `__subclasshook__()` **regular method**

Summary

Summary

Summary

Abstract Base Classes (ABCs)

Summary

Abstract Base Classes (ABCs)

`ABCMeta` **allows registration of any class – even built-in classes – as a virtual base class**

Summary

Abstract Base Classes (ABCs)

`ABCMeta` allows registration of any class – even built-in classes – as a virtual base class

`@abstractmethod` makes classes abstract and ensures overriding of methods

Summary

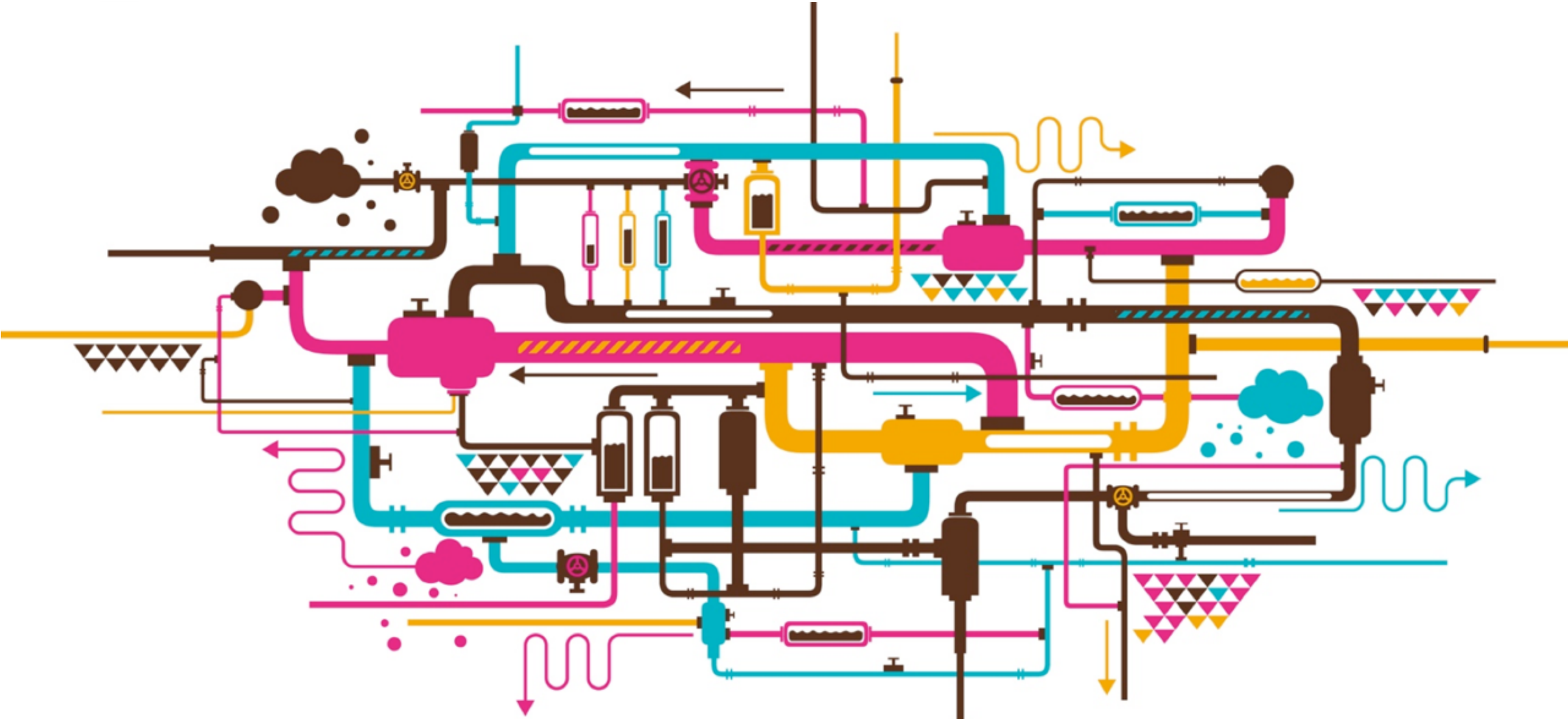
Abstract Base Classes (ABCs)

`ABCMeta` allows registration of any class – even built-in classes – as a virtual base class

`@abstractmethod` makes classes abstract and ensures overriding of methods

`@abstractmethod` **must be innermost when combined with** `@staticmethod`, `@classmethod` **or** `@property`

python™



Companion **Python Craftsman** Book Series

https://leanpub.com
/python-apprentice
/c/pluralsight

https://leanpub.com
/python-journeyman
/c/pluralsight

https://leanpub.com
/python-master
/c/pluralsight

Companion **Python Craftsman** Book Series

Pluralsight
Python Fundamentals

https://leanpub.com
/python-apprentice
/c/pluralsight

https://leanpub.com
/python-journeyman
/c/pluralsight

https://leanpub.com
/python-master
/c/pluralsight

Companion **Python Craftsman** Book Series

Pluralsight
Python Fundamentals

https://leanpub.com
/python-apprentice
/c/pluralsight

Pluralsight
Python – Beyond the Basics

https://leanpub.com
/python-journeyman
/c/pluralsight

https://leanpub.com
/python-master
/c/pluralsight

Companion **Python Craftsman** Book Series

Pluralsight
Python Fundamentals

https://leanpub.com
/python-apprentice
/c/pluralsight

Pluralsight
Python – Beyond the Basics

https://leanpub.com
/python-journeyman
/c/pluralsight

Pluralsight
Advanced Python

https://leanpub.com
/python-master
/c/pluralsight


Fun!