A class that is derived from an abstract class cannot be instantiated unless all of its abstract methods are overridden.

In [1]:

```
from abc import ABC, abstractmethod

class AbstractClassExample(ABC):

    @abstractmethod
    def do_something(self):
        print("Some implementation!")

class AnotherSubclass(AbstractClassExample):

    def do_something(self):
        super().do_something()
        print("The enrichment from AnotherSubclass")

x = AnotherSubclass()
x.do_something()
```

```
Some implementation!
The enrichment from AnotherSubclass
```

In [2]:

```
from abc import ABC, abstractmethod

class AbstractClassExample(ABC):

    def __init__(self, value):
        self.value = value
        super().__init__()

    @abstractmethod
    def do_something(self):
        pass
```

In [3]:

```
class DoAdd42(AbstractClassExample):
    pass

x = DoAdd42(4)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-3-2bcc42ab0b46> in <module>
      2     pass
      3
----> 4 x = DoAdd42(4)

TypeError: Can't instantiate abstract class DoAdd42 with abstract methods do
_something
```

In [4]:

```
class DoAdd42(AbstractClassExample):

    def do_something(self):
        return self.value + 42

class DoMul42(AbstractClassExample):

    def do_something(self):
        return self.value * 42

x = DoAdd42(10)
y = DoMul42(10)

print(x.do_something())
print(y.do_something())
```

```
52
420
```

In [ ]: