```
class DecoratorMeta(type):
    def __call__(cls, *args, **kwargs):
        # Add additional logic or modification before class creation
        print(f"Decorating class: {cls.__name__}")

        # Call the actual class creation process using the default type.__call__
        instance = super().__call__(*args, **kwargs)

        # Add post-creation logic if needed
        print(f"Class {cls.__name__} created successfully")

        return instance

# Example usage
class MyClass(metaclass=DecoratorMeta):
    def __init__(self, value):
        self.value = value

# Creating an instance of MyClass triggers the decorator-like behavior
my_instance = MyClass(value=42)
```

```
Decorating class: MyClass
Class MyClass created successfully
```

Start coding or generate with AI.