```python
# Base Class: Component
class Component:
    def __init__(self, name, material):
        self.name = name
        self.material = material

    def calculate_volume(self):
        raise NotImplementedError("Subclasses must implement this
method.")

    def calculate_weight(self, density):
        """Calculates weight based on density and volume."""
        return self.calculate_volume() * density

# Subclass: Beam
class Beam(Component):
    def __init__(self, name, material, length, width, height):
        super().__init__(name, material)
        self.length = length
        self.width = width
        self.height = height

    def calculate_volume(self):
        """Calculates volume of the beam."""
        return self.length * self.width * self.height

# Subclass: Cylinder
class Cylinder(Component):
    def __init__(self, name, material, radius, height):
        super().__init__(name, material)
        self.radius = radius
        self.height = height

    def calculate_volume(self):
        """Calculates volume of the cylinder."""
        return 3.14159 * (self.radius ** 2) * self.height
```

Base Class: `Component`

The `Component` class provides a generic blueprint for components:

- **Attributes**
  - `name` (public): Name of the component.
  - `material` (public): Material used in the component.
- **Abstract Method**
  - `calculate_volume()`: Must be implemented by subclasses.
- **General Method**

    - `calculate_weight(density)`: Calculates weight based on the component's volume and material density.

Example

```
component = Component("Generic Component", "Material")  # Cannot directly instantiate because of the abstract method.
```

Subclass: `Beam`

The `Beam` class specializes `Component` for rectangular beams:

- **Attributes**
    - `length`, `width`, `height`: Dimensions of the beam.
- **Methods**
    - Implements `calculate_volume()` to compute the volume of the beam.

Example

```
beam = Beam("Concrete Beam", "Concrete", 6.0, 0.3, 0.5)
print(beam.calculate_volume())  # Output: 0.9 cubic meters
print(beam.calculate_weight(2400))  # Output: 2160 kilograms
```

Subclass: `Cylinder`

The `Cylinder` class specializes `Component` for cylindrical components:

- **Attributes**
    - `radius`, `height`: Dimensions of the cylinder.
- **Methods**
    - Implements `calculate_volume()` to compute the volume of the cylinder.

Example

```
cylinder = Cylinder("Steel Cylinder", "Steel", 0.5, 2.0)
print(cylinder.calculate_volume())  # Output: 1.570795 cubic meters
print(cylinder.calculate_weight(7850))  # Output: 12312.73825 kilograms
```

Summary

This civil engineering example demonstrates:

1. **Abstract Classes**: Using `Component` as a generic base class.
2. **Encapsulation and Inheritance**: Specialized subclasses (`Beam`, `Cylinder`) inherit from the base class and implement specific volume calculations.
3. **Polymorphism**: Both `Beam` and `Cylinder` can be treated as `Component` while preserving their unique behaviors.
4. **Practical Application**: Includes methods for calculating material volume and weight, essential for civil engineering design and analysis.

Example Usage

```python
# Beam Example
beam = Beam("Concrete Beam", "Concrete", 6.0, 0.3, 0.5)
print(f"{beam.name} Volume: {beam.calculate_volume()} cubic meters")
print(f"{beam.name} Weight: {beam.calculate_weight(2400)} kilograms")  # Assuming density of 2400 kg/m^3

# Cylinder Example
cylinder = Cylinder("Steel Cylinder", "Steel", 0.5, 2.0)
print(f"{cylinder.name} Volume: {cylinder.calculate_volume()} cubic meters")
print(f"{cylinder.name} Weight: {cylinder.calculate_weight(7850)} kilograms")  # Assuming density of 7850 kg/m^3
```