Object-Oriented Programming Concepts

1. Encapsulation

Encapsulation is the concept of bundling data and methods that operate on that data within a single unit (class). Private attributes are used to protect data from direct access.

Diagram:

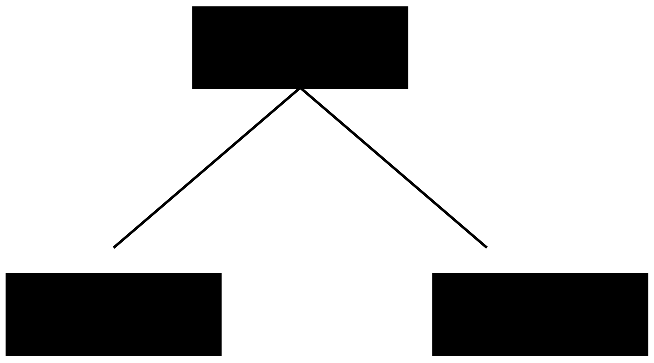
Accessing Private Attributes Using Getter and Setter

```
class Student:
    def __init__(self, name, grade):
        self.__name = name
        self.__grade = grade

    def get_name(self):
        return self.__name

    def set_grade(self, grade):
        self.__grade = grade
```

2. Inheritance

Inheritance allows a class to acquire properties and methods from another class, enabling code reuse.

Diagram:



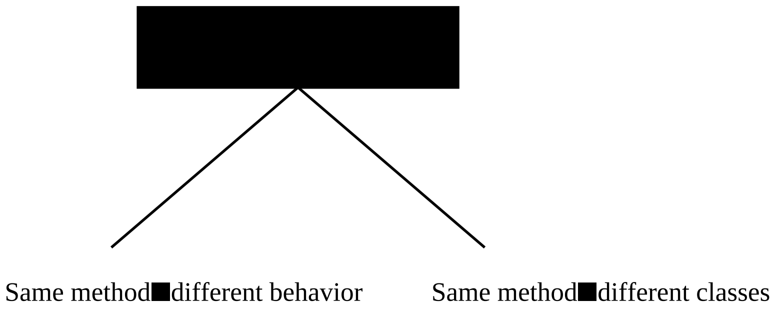
Types of Inheritance with Examples

```
# Single Inheritance
class Animal:
    def sound(self):
        return "Some sound"

class Dog(Animal):
    def sound(self):
        return "Bark"

# Multiple Inheritance
class A:
    def method_a(self):
        return "A"

class B:
    def method_b(self):
        return "B"

class C(A, B):
    pass
```

3. Polymorphism

Polymorphism allows different classes to implement the same method in different ways.

Diagram:



Overloading vs Overriding

- **Overloading**: Same method name, different parameters (not fully supported in Python, simulated using default parameters).

- **Overriding**: A child class redefines a method of the parent class.

```
class Shape:
    def area(self):
        return 0

class Circle(Shape):
    def area(self):
        return 3.14 * 5 * 5

# Overloading simulation
def add(a, b=0):
    return a + b
```