

EXPERIMENT-10

- **AIM:** - To implement an object-oriented program in which a Person class is defined as a base class, and Student and Teacher classes are derived from it, demonstrating the concept of inheritance and polymorphism.
- **THEORY:** - Object-Oriented Programming (OOP):
 - It models real-world entities as classes and objects.
 - Provides features like inheritance, polymorphism, encapsulation, and abstraction.
 - Inheritance:
 - Mechanism of creating a new class using the properties and behaviors of an existing class.
 - Promotes code reusability.
 - Class Hierarchy:
 - A structured representation where Person is the parent class.
 - Student and Teacher are child classes that inherit from Person.
 - Application:
 - Such a hierarchy is used in university/school management systems, where persons may have different roles (student, teacher, staff).
- **CODE:-**

Base Class

```
class Person:
```

```
    def __init__(self, name, age):
```

```
        self.name = name
```

```
        self.age = age
```

```
    def display_info(self):
```

```
        print(f"Name: {self.name}, Age: {self.age}")
```

Derived Class: Student

```
class Student(Person):
```

```
    def __init__(self, name, age, student_id, course):
```

```
        super().__init__(name, age)
```

```
        self.student_id = student_id
```

```
        self.course = course
```

```
    def display_info(self):
```

```
        super().display_info()
```

```
        print(f'Student ID: {self.student_id}, Course: {self.course}')
```

Derived Class: Teacher

```
class Teacher(Person):
```

```
    def __init__(self, name, age, employee_id, subject):
```

```
        super().__init__(name, age)
```

```
        self.employee_id = employee_id
```

```
        self.subject = subject
```

```
    def display_info(self):
```

```
        super().display_info()
```

```
        print(f'Employee ID: {self.employee_id}, Subject: {self.subject}')
```

Driver Code

```
print("----Student Details----")
```

```
s1 = Student("Sara Kumari", 20, "S101", "Computer Science")
```

```
s1.display_info()
```

```
print("\n----Teacher Details----")
```

```
t1 = Teacher("Dr. Sharma", 45, "T501", "Mathematics")
```

```
t1.display_info()
```

➤ **OUTPUT**→

```
----Student Details----  
Name: Sara Kumari, Age: 20  
Student ID: S101, Course: Computer Science  
  
----Teacher Details----  
Name: Dr. Sharma, Age: 45  
Employee ID: T501, Subject: Mathematics
```

➤ **LEARNING OUTCOMES**-→

- ✓ Understand and implement class inheritance in OOP.
- ✓ Create a hierarchical relationship between classes.
- ✓ Demonstrate method overriding using polymorphism.
- ✓ Apply OOP concepts to real-world modeling (students, teachers, staff, etc.).
- ✓ Enhance code reusability and modularity in software design.