

Introduction to Object-Oriented Programming (OOP)

Language: C++

Prepared by: Nabeela Ashraf
Lecturer @CS Department
UAF

What is Object-Oriented Programming?

- Object-Oriented Programming (OOP) is a programming approach that uses 'objects' to design and organize software.
- Objects contain data (attributes) and behavior (methods).
- OOP helps make code more organized, reusable, and easier to maintain.

Basic Building Blocks of OOP

- 1. Class – Blueprint that defines data and functions.
- 2. Object – Instance of a class.
- 3. Encapsulation – Hiding data inside classes.
- 4. Inheritance – Reusing code from parent classes.
- 5. Polymorphism – One interface, multiple behaviors.
- 6. Abstraction – Showing only essential details

Class and Object

- Class is a blueprint for creating objects.
- Object is an instance of a class.

Example C++ Code:

```
class Car {  
public:  
    string color;  
    void start() { cout << "Car Started"; }  
};
```

```
Car myCar;
```

Class: Car
- color
- start()

Object:
myCar

Object:
yourCar

Encapsulation

Encapsulation hides data from direct access and allows control through methods.

Example:

```
class Student {  
    private:  
        int marks;  
    public:  
        void setMarks(int m) { marks = m; }  
        int getMarks() { return marks; }  
};
```



Inheritance

Inheritance allows one class to use the properties of another.

Example:

```
class Vehicle {  
public:  
    void move() { cout << "Moving"; }  
};
```

```
class Car : public Vehicle {  
public:  
    void honk() { cout << "Beep!"; }  
};
```

Parent: Vehicle

Child: Car

Polymorphism

Polymorphism means 'many forms' – a function behaves differently based on the object.

Example:

```
class Shape {  
    public:  
        virtual void draw() { cout << "Drawing Shape"; }  
};  
class Circle : public Shape {  
    public:  
        void draw() { cout << "Drawing Circle"; }  
};
```

Abstraction

Abstraction hides complex details and shows only essential features.

Example:

```
abstract class Machine {  
public:  
    virtual void start() = 0; // Pure virtual function  
};
```

```
class Fan : public Machine {  
public:  
    void start() { cout << "Fan Started"; }  
};
```

Summary

- OOP provides a modular, reusable, and maintainable approach to programming.
- Main Concepts:
 - Class & Object – Real-world modeling
 - Encapsulation – Data protection
 - Inheritance – Code reuse
 - Polymorphism – Flexible behavior
 - Abstraction – Simplified complexity

Class Activity

- Task 1:

Create a class 'Student' with data members: name, rollNo, marks.
Add methods to input and display data.
- Task 2:

Create a class 'Teacher' inherited from 'Person'. Add method 'showDetails()'.
- Task 3:

Demonstrate polymorphism using a 'Shape' base class with 'draw()' function.