**👨🏽‍💻 C++ 101 – Session 11 Notes**

**🧵 Topic: Object-Oriented Programming (OOP) Basics**

**What is Object-Oriented Programming (OOP)?**

**OOP** is a programming approach based on the concept of **"objects"** — reusable components that contain both **data** (variables) and **functions** (methods).

C++ is an **object-oriented language**, which means it supports OOP principles like:

* **Encapsulation**
* **Abstraction**
* **Inheritance**
* **Polymorphism**

In today’s session, we focused on **encapsulation**, which is about grouping related variables and functions into a class.

**Classes and Objects**

**🔹 What is a Class?**

A **class** is a **blueprint** for creating objects.  
It defines **what data** an object will have and **what it can do**.

**🔹 What is an Object?**

An **object** is an **instance of a class**.  
When you create an object, you're creating a **real version** of the blueprint.

**Class Definition**

To define a class, use the class keyword:

class Car {

   // class body

};

This creates a blueprint called Car.

**Class Data (Variables)**

These are the **attributes** of the class — also called **member variables**.

In our example:

string brand;

string model;

string color;

int year;

float weight;

float price;

These represent **each car’s information**.

**Class Methods (Functions)**

These are **functions inside the class** that describe the behavior of the object.

Example:

void start() {

    cout << "Car started." << endl;

}

**Access Specifiers**

C++ classes use **access specifiers** to control how members (variables/functions) are accessed.

| **Specifier** | **Meaning** |
| --- | --- |
| public | Members are accessible from **outside** the class |
| private | Members are accessible **only within** the class |
| protected | Used in inheritance (we’ll cover later) |

In this session, we used public to make variables and methods accessible.

**Object Creation**

We create objects by using the class name:

Car car1;  // car1 is an object of class Car

Car car2; // car2 is another object of class Car

Each object has its own **copy** of the class data.

**Assigning Values to Object Variables**

You can assign values to object attributes using the dot operator (.):

car1.brand = "Toyota";

car1.model = "Corolla";

car1.year = 2020;

Each object can have **different values** for the same attributes.

**Accessing Object Methods**

You also use the dot operator to call methods:

car1.start();       // Outputs "Car started."

car2.turnLeft();    // Outputs "Car turned left."

These methods can **print information**, **return values**, or **modify data**.

**Sample Code (From Class)**

We used a Car class to demonstrate everything:

class Car {

public:

    string brand;

    string model;

    string color;

    int year;

    float weight;

    float price;

    void start() { cout << "Car started." << endl; }

    void turnLeft() { cout << "Car turned left." << endl; }

    void turnRight() { cout << "Car turned right." << endl; }

    void brake() { cout << "Car stopped." << endl; }

    void printDetails() {

        cout << "Brand: " << brand << endl;

        cout << "Model: " << model << endl;

        cout << "Color: " << color << endl;

        cout << "Year: " << year << endl;

        cout << "Weight: " << weight << " kg" << endl;

        cout << "Price: $" << price << endl;

    }

};

Then in main() we created two cars and printed their details:

int main() {

    Car car1;

    car1.brand = "Toyota";

    car1.model = "Corolla";

    car1.color = "Red";

    car1.year = 2020;

    car1.weight = 1300.5;

    car1.price = 20000.0;

    car1.printDetails();  // prints details of car1

    Car car2;

    car2.brand = "Benz";

    car2.model = "C-Class";

    car2.color = "Black";

    car2.year = 2021;

    car2.weight = 1500.0;

    car2.price = 40000.0;

    car2.printDetails();  // prints details of car2

    return 0;

}

**📌 Assignment**

🎯 **Task:** Look into **constructors** in C++

**What to find out:**

* What is a constructor?
* How to define and use one
* Types of constructors (default, parameterized)
* Why constructors are useful in OOP

You’ll use this knowledge in the **next session** to make object creation easier and cleaner.

**🧠 Summary Table**

| **Concept** | **Description** |
| --- | --- |
| **Class** | A blueprint for creating objects (defines variables and methods) |
| **Object** | An instance of a class |
| **Member Variables** | Data stored in an object |
| **Member Methods** | Actions/behaviors of the object |
| **Access Specifiers** | Control how members are accessed (public, private) |
| **Object Creation** | Instantiating a class using ClassName objectName; |
| **Dot Operator (.)** | Used to access object members |
| **Assignment** | Learn about constructors for next class |