LECTURE 2

LEARN BY SOLVING PROBLEMS, NOT MEMORIZING DEFINITIONS

TODAY'S MOTTO

- ACCESS MODIFIERS
- ENCAPSULATION
- GETTERS AND SETTERS

ACCESS SPECIFIERS IN C++

Access Specifiers in C++

- public → Accessible everywhere
- private → Accessible only inside the class
- protected → Accessible in derived classes

Access Modifiers \ Scope	Global	Derived class	Friend class	Within class
Private	✗	✗	✓	✓
Public	✓	✓	✓	✓
Protected	✗	✓	✓	✓

WARM-UP QUESTION (PREDICTION BASED)

```
class Test {
    int x;
public:
    int y;
};

int main() {
    Test t;
    t.y = 10;
    t.x = 5;
}
```

Predict:

- **Will it compile?**
- **If not, why?**

PROBLEMS

1.

Secure Bank Account

Design a class BankAccount such that:

- Balance cannot be modified directly
- Anyone can check balance
- Only deposit & withdraw methods can change it
- Withdraw should fail if balance is insufficient

Constraints

- balance must be hidden
- Invalid withdrawal must be prevented

```
class BankAccount {
private:
    int balance;

public:
    BankAccount() { balance = 0; }

    void deposit(int amount);
    bool withdraw(int amount);
    int getBalance();
};
```

ENCAPSULATION

- Encapsulation
- Hide data
- Access through controlled methods (get / set)

- Key Rule
- Data private + public methods = Encapsulation

Problem: Online Exam System

Design a class ExamResult such that:

- Marks are private
- Marks must be between 0–100
- Grade is calculated automatically
- User cannot directly change grade

Constraints:

- Encapsulation mandatory
- Invalid state must be impossible

SCENARIO BASED QUESTION

SCENARIO: DIGITAL WALLET SYSTEM

- YOU ARE DESIGNING A DIGITAL WALLET USED BY MILLIONS OF USERS.

RULES (MUST BE ENFORCED)

- WALLET BALANCE MUST NEVER BE DIRECTLY ACCESSIBLE OR MODIFIABLE
- BALANCE CANNOT GO NEGATIVE
- EVERY TRANSACTION MUST BE VALIDATED
- TRANSACTION HISTORY CANNOT BE MODIFIED FROM OUTSIDE
- USERS CAN ONLY INTERACT THROUGH ALLOWED OPERATIONS

SYSTEM REQUIREMENTS

- DESIGN A CLASS DIGITALWALLET WITH THE FOLLOWING BEHAVIOR:
- PRIVATE DATA (MUST BE HIDDEN)
- BALANCE
- TRANSACTIONCOUNT
- TRANSACTIONHISTORY[]

PUBLIC INTERFACE (CONTROLLED ACCESS)

- ADDMONEY(AMOUNT)
- PAY(AMOUNT)
- GETBALANCE()
- GETTRANSACTIONCOUNT()
- PRINTSTATEMENT()

CONSTRAINTS

- ADDMONEY(AMOUNT)
- AMOUNT MUST BE POSITIVE
- PAY(AMOUNT)
- AMOUNT MUST BE POSITIVE
- BALANCE MUST BE SUFFICIENT

N**O METHOD SHOULD ALLOW:**

- NEGATIVE BALANCE
- DIRECT ACCESS TO TRANSACTION LIST
- TRANSACTIONHISTORY CANNOT BE RETURNED DIRECTLY
- TRANSACTION COUNT MUST UPDATE AUTOMATICALLY

ANY QUESTION?

THANK YOU