Ref: Object oriented programming with C++, Balaguruswamy

01

CSE 261
10.12.2023

CT → Best will be counted

Program: Set of instructions which will perform a task.

Programming: To solve a program

Programming Language [HLL → High Level Language
LLL

Characteristics of a successful Language

- Simplicity
- Reliability
- Support
- Abstraction
- Efficient implimentation

LLL → Machine Friendly Language

HLL → Human " "

Assembly Language → mid level language

- Opcode

Structured Programming / Procedural Language

C program

- .c
- .exe
- .o

Object Oriented Program

c++

array
struct
.
.
.
sequence
selection
iteration

Task

To revise C

c21

CSE

17.12.2023

Object Oriented Paradigm → C++

Structured Programming → C + OOP

Structure → Class / Object

Array → Similar type data

Structure → Different type data

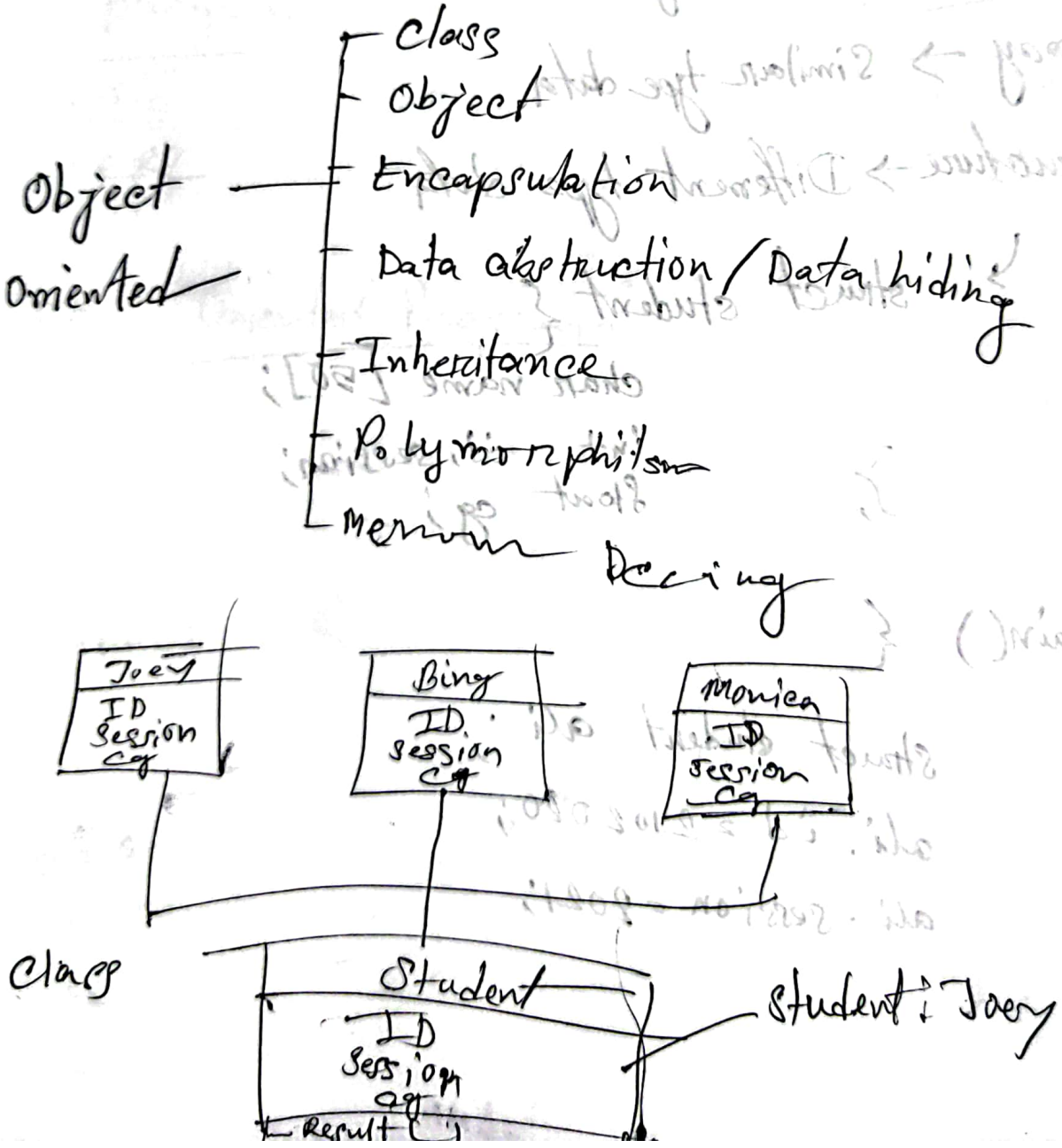
```
struct student {
        char name[50];
        int id, session;
        float cg;
};

main() {
        struct student ali;
        ali.id = 2102020;
        ali.session = 2021;
```

Disadvantage / Drawback

- No data hiding
- Functions cannot be used

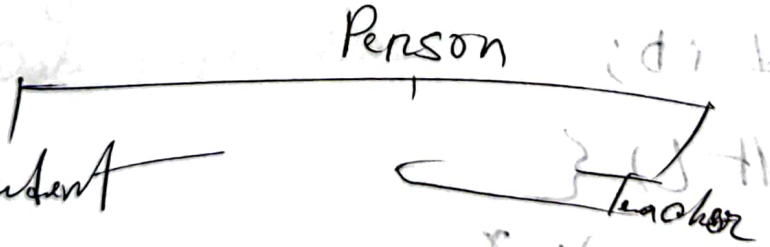
Data vs Function



```
Class student {
private:
    int cg; — Data member
Public:
    string name;
    int session;
    void result();
};
```

→ Member function / method

Data | Method → Encapsulation



Inheritance → Reuse

Polymorphism

Poly - many

morphism - form

```
int add (int a, int b) { }
                  sz a+b
int add (int a, int b, int c, sz a+b+c)}
```

Message Passing

```
Class student {
    float cg;
    Public int ID;
    void result () {
     printf ("Pass"); }

main() {
        student joey;
        joey.result();
        return 0;
        }
```

Benefits of OOP
Applications of OOPs

Chapter 1.

CS| CSE 261
15.01.24

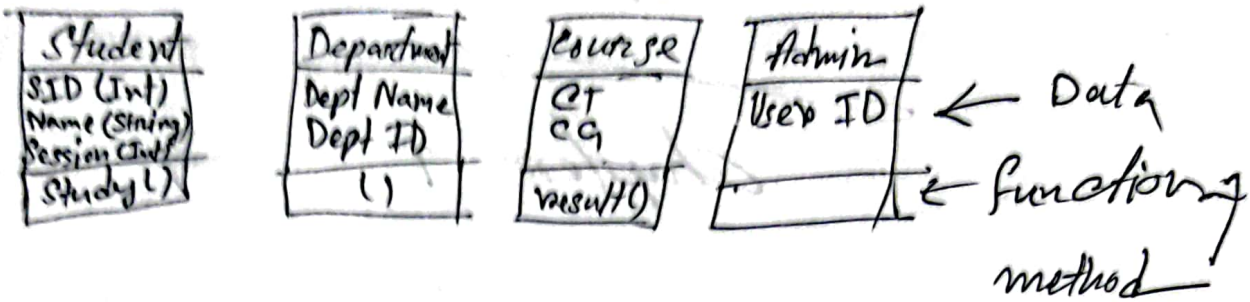
OOP model / Class diagram

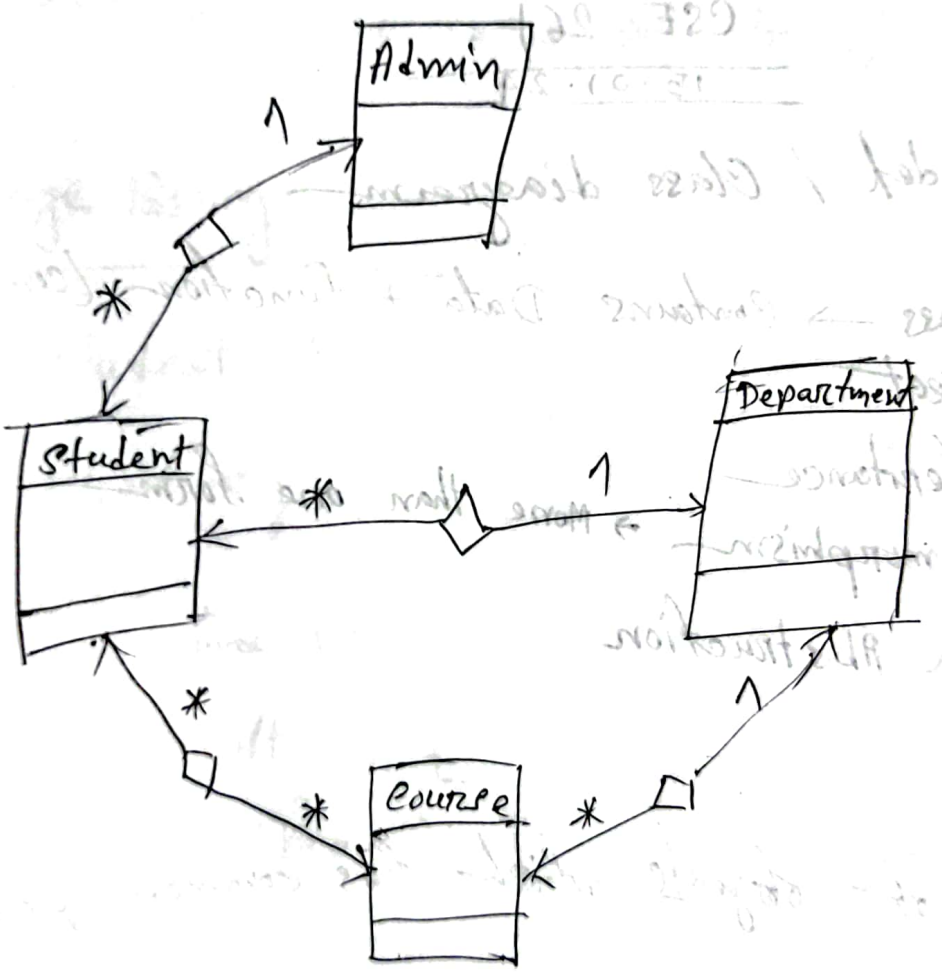
- Class → Contains Data + function (common property)
- Object
- Inheritance
- Polymorphism → More than one form
- Data Abstraction

Class
Group of objects which has common properties

Object
Entities which have some attributes

Student Information Management System





OOP Model

1 question in final, mid

Chapter 2

C ⟷ C++

(Ancestor) — C++ is a superset of C

C++ supports the functions of C, but C cannot support the functions of C++

To Read: Structure vs C vs C++ vs Java

Code

.c

```
#include <stdio.h>
int main() {
    printf("Hello world");
    return 0;
}
```

.cpp

```
#include <iostream>
int main() {
    cout << "Hello world";
    return 0;
}
```

iostream → input output stream

↳ key board ↳ monitor

using namespace std;

Identifier
↓
Variable name
function name

<bits/stdc++.h>
↳ To call all functions at once.

Bitwise Operation << → Left shift

Cout << → insertion / put to

scanf("%d", &a); | cin >> a;
↳ exertion operator

IDE → Integrated Developmet Environment

.o ↳ object → Intermediate Object file

.exe ↳ executable → Linked up with .o file

To read [C tokens → Identifier Declaration

Chapter 3

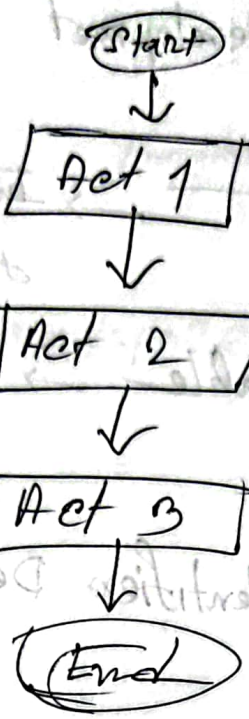
3.25

To read [Control Structures
Control Statements

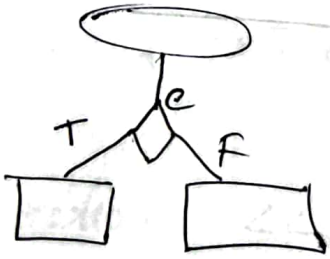
i) Sequence
ii) Selection
iii) Iteration / Loop

To read [Fig 3.4, 3.5

Sequence /



Selection / If else



Iteration /

Loop → 1. Exit control / Post test

2. Entry control / Pre test

for (Initialization; condition; increment / decrement) {

}

→ Entry control

while (condition) {

}

→ Entry control

DO

{

statement

}

while (condition)

→ Exit control

Jump statement

break; → exit from loop

continue; → skip

Switch statement

Next Class → Chapter 4

Functions and C++

C++ CSE 261
23.01.24

*Function / Method

*STL
swap()
pow()

```
int main() {

    cout << "Hello" << endl;
    return 0;
}
```

*Prototype → dummy
Function prototyping → no definition, only declaration

```
int add (int a, int b)
```

int → return type
add → Full name
(int a, int b) → parameter list / arguement

4.4 → Call by reference

- Bubble sort

C++ → referencing C → pointer

* what is inline function? (4.6)
  - Link up
  - 'inline' keyword
  - saves time

Recursion

- factorial n(n-1)
- Tower of Hanoi
- Fibonacci sequence
- Power of Number

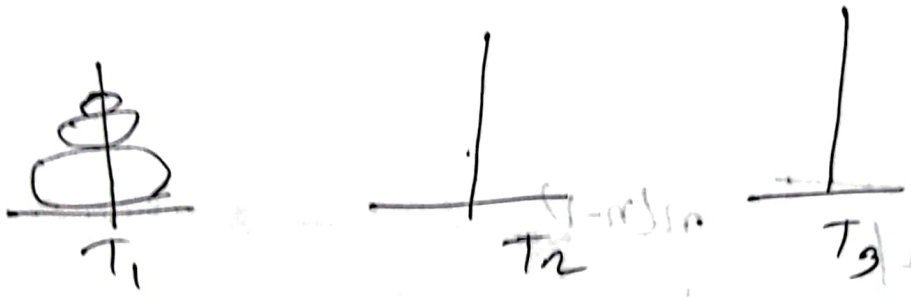
```
int fac(int a) {
    if (a == 1) {   //base case
        return 1;
else
    return a * fac(a-1)
```

[Write the recursion function of factorial]

* Manual Tracing
  - Work as a compiler

[How to solve it!]

Tower of Hanoi



Move the discs from T_1 to T_3

Conditions

- One disc at a time
- Bigger disc can't be kept on smaller disc

Fibonacci Sequence

$$F_n = f_{n-2} + f_{n-1}$$

Power of Numbers

9^3

$3 \times 3 \times 3$

Function Overloading

```
add (int a, int b);
add (int a, int b , int c );
add (int a , int b; int c, int d);

{ int s1 = a+b ; cout << s1 ; }
{ int s2 = a+b+c ; cout << s2 ; }
{ int s3 = a+b+c+d ; cout << s3 };
```

int add (int a , float b);

Quiz (1-4)
01.02.23

After 2 chapters → Mid (1-6)

CSE 261
01.02.24

CSI

Classes and Structures | chapter-5

C Structure

funcX
Data hiding X

C++ has data hiding in structure.

Class has two types of member.

1. Data
2. Function

```
Student {
        int id;
        string name;

    void result (void);
```

- Access specifier

Keyword that expresses if the data is public or private or protected.

Fig : 5.1

~~Compiler~~

Compiler ⟶ By default private

Object Creation

```
int main() {
Student roy;
        roy.id = 2102100;
        roy.cg = 3.98
        roy.result();
```

Getter & Setter Method

Getter → Retrive / Return

Setter → Set

=

```
Class Student {
        int id;
Public:
    void setdata (int i); //Setter
    int getdata ( ){ //Getter
        return id };

main( ) {
    student s;
    s.setdata (2102200);
    s. getdata ( );
    return 0;}

}
```

```
void student :: setdata(int i){
    id=i;
}
```

Value Pass

Reference pass

Inline function → Linkup
Time saving

Characteristics of member function

Syllabus

Pdf (6) + Pdf + 3 → Quiz

Quiz + Tut 6 → Mid

CSE 261

05.02.21

```
class add {
        int x;
        int y;
Public:
    void setdata (int a, int b){
        x = a;
        y = b; }
    void display (void);
};
void :: add :: display (void) {
        cout << x+y;
    }
```

```
main() {
    add sum;
    sum.setdata(10,20);
    sum.display();
    sum.x
```

add (int , int) → only class name

Program 5.7

Friend function

Friend fun

↓

can access private data, not a member. object is not necessary to call it

```
class add {
    int x;
  Public:
friend void sumx (void) {
    cout << x+x;
}
```

Imp. ← Characteristics of friend function

. Program 5.8

Constructors and Distructors

```
class sum {
    int a, b;
  Public:
    sum (int x, int y) {
        a = x;
        b = y; }
void display ( ) {
    cout << a+b; }
};
```

```
main() {
sum add (10,20);
add display();
}
```

```
class {
    int a, b, c;
public:
    sum(int x, int y) {
        a=x;
        b=y; cout<<a+b; }

    sum(int x, int y, int z) {
        a=x
        b=y;
        c=t;
        cout<<a+b+c; }

main() {
implicit sum add(10,20,30)
sum(add)(10,30)
}
```

~ → tilde operator

```
class sum {
    int a, b, c;
public
    sum(int x, int y) {
        a = x;
        b = y;  cout << a+b; {
Sum(int x, float y) {
        a = x;
        b = y;
~sum() { cout << a+b;
cout << "  .  " ; >

        main() {
implicit sum add1(10, 20);
        sum add2(10, 30.5);
} return 0;
```

Mid