PAGE No.
DATE 04/10/2023

Assignment -13

Q] What is use of copy constructor?

Ans Copy constructor is used to ~~initialise members of one object~~ copy the members of one object into another object's members

eg:

```
Class Demo
{
    Pubic:
        int i, j;

    Demo (int a, int b)
    {
        cout<<" Inside Parameterized constructor";
        i=a;
        j=b;
    }
    Demo ( Demo & ref)
    {
        cout<<"\n Inside copy constructor";
        i = ref.i;
        j = ref.j;
    }
};
int main ()
{
    Demo obj1 (11, 12);
    Demo obj2 ( obj1);
}
```

PAGE No.
DATE / /

Q2] When the constructor & destructor get called?

Ans.] Constructors & destructors are considered special functions.

:] Constructor is a function which gets automatically called when we create the object of class.

:] Destructor gets automatically called before deallocating the memory of an object.

Q3] What are the rules which has to be followed while writing constructors & destructors?

Ans Rules are:

1] Name of constructor should be same as class name.

2] The constructors & destructors should be inside public access specifier.

3] There should not be return value from the constructor & destructor

4] There is no need of explicit call to constructor or destructor.

5] If we create an object without passing any parameter, then the default constructor gets called.

6] If we create an object by passing parameters, the parameterized constructor gets called.

7] If we create an object by passing another object as a parameter, there the copy constructor gets called.

Q4] What is meant by this pointer?

Ans

1] In case of C++ & Java to call any non-static member function we need object of that class

2] By using the object & the direct accessing operator ('.'), we can call any non-static method of class

3] When are use that caller object, internally the compiler will send address of an object as the first hidden implicit parameter

4] That address gets store inside first ~~hi~~ implicit argument of the function & that argument is called as 'this pointer'

5] 'this' is considered as keyword in C++ & Java

Important points about this pointer

~~Imp~~ ~~thi~~

1] this pointer is an implicit pointer added by the compiler directly

2] For every non-static member function of a class, the concept of this pointer gets added implicitly

3] this pointer is always considered as the first argument of non-static function

4] ~~According~~ To call an non-static function we need an object & the compiler will pass address of that object as the first parameter to every function

* That address gets accepted inside a special ~~function~~ pointer i.e. this pointer

5] The concept of this pointer is applicable inside constructor, destructor, inside any non-static member function.

6] If there are N arguments to the non-static member function, then after compilation, the arguments become N+1.

7] In Java as there no concept of pointer it is called this reference.

Q5] What is the prototype of this pointer?

Ans

~~Eg:~~

~~void fun (in~~

Eg:

```
Class Demo
{
    public:
    int i,j,k;

    void fun(int no1, int no2)
    {
        cout<<"Inside fun\n"<<endl;
    }
// void fun (Demo * this, int no1, int no2)
};
int main(){
Demo Obj;
    obj.fun(5,6); // fun(&obj,5,6)
                  // fun(Address of obj, 5, 6)
}
```

Q6] What is meant by inheritance? Explain types of inheritance.

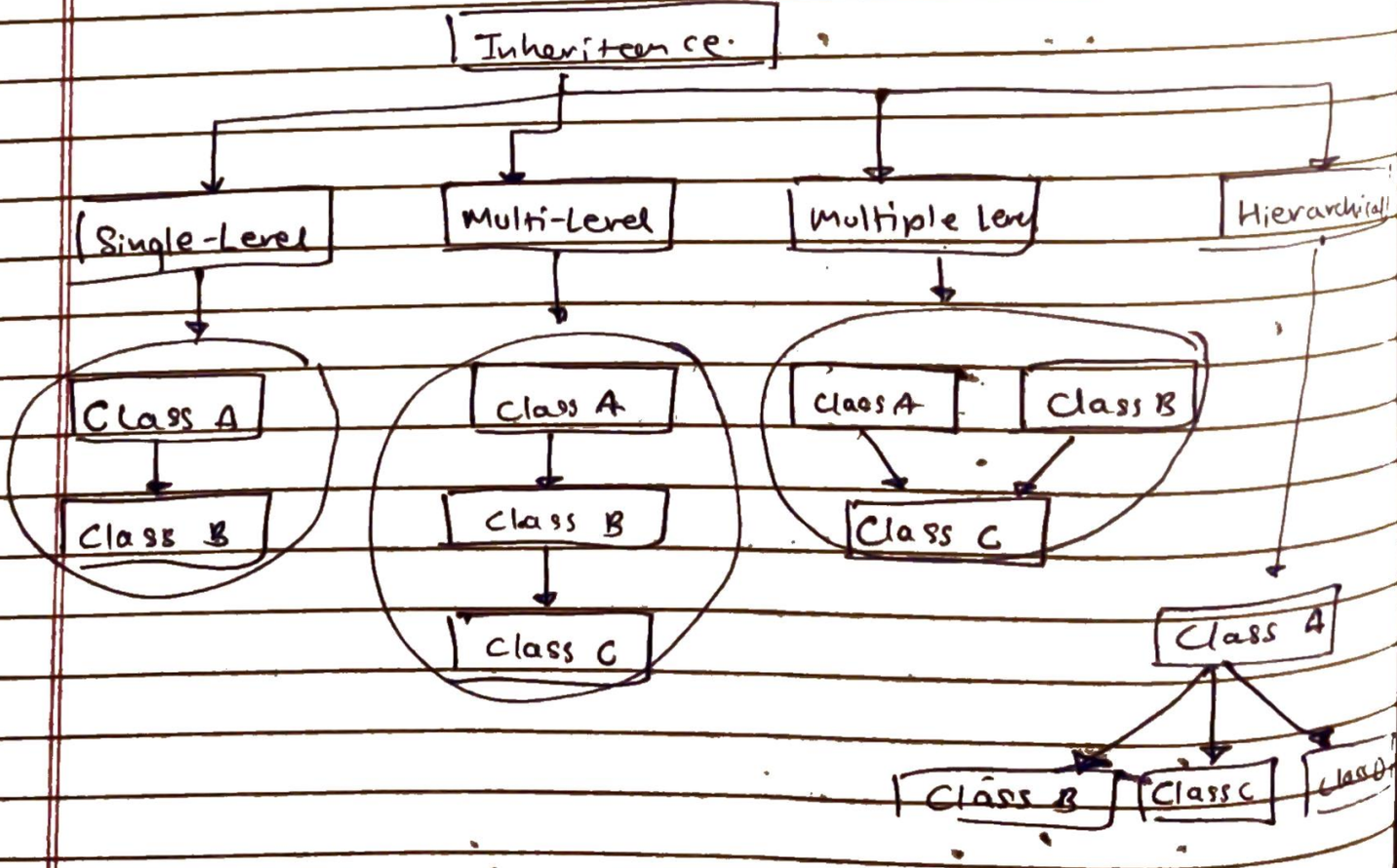
Ans

1] Inheritance is considered as one of the object oriented paradigm in C++ & Java

2] In simple terms, inheritance is defined as reusability.

3] By using inheritance one class can acquire characteristics & behaviours of other class.

According to structural layout there are 4 different types of inheritances



PAGE No.
DATE / /

1] Single-Level-Inheritance:

In Single level there is one parent class & one derived class from it only.
Only two classes exist.

2] Multi-Level-Inheritance:

In Multi level there, it is considered as extension of single level inheritance
Incase of multi-level Inheritance, there should be at-least 3 levels

3] Mutiple-Inheritance:

In multiple inheritance, one class can inherit more that none classes at a time.

Q] What is meant by polymorphism? Explain its types.
Ans The term polymorphism is defined as single name & multiple behaviours.

Polymorphism is of two types
1] Compile-time polymorphism
2] Run-time polymorphism.

1] The concept of compile-time polymorphism is considered as early binding because the decision taken by the compiler is in early phase.

PAGE No.
DATE

1] To achieve compile-time polymorphism in C++ & Java we use term called overloading.

2] The Run-time polymorphism is called Considered Late binding because the decision taken by compiler is in late phase.

To achieve run-time polymorphism in C++ & Java we use the term called overriding

Q8] What is the use of & operator in case of copy constructor?

Ans The & operator used inside copy constructor is considered as a reference operator & if that operator is not written, it may lead to recursive calls.
It refers to the object in parameter.

Q9] Why the name of constructor & destructor is same as class name?

Ans

@10] What is meant by function overloading?

Ans The concept of compile time polymorphism is considered as early binding because the decision taken by the compiler is in the early phase.

1] To achieve compile time polymorphism in C++ & Java we have to use the term called as overloading.

2] There are two types of overloading
- 1] Function overloading
- 2] Operator overloading

3] In case of function overloading we can define multiple functions in a class with the same name & with different prototypes

Example

```
class Demo
{   public:
    int addition (int a, int b)  //Addition@2ii
    {
        int ans = 0;
        ans = a+b;
        return ans;
    }
    int addition (int a, int b, int c) //Addition@3iii
    {
        int ans = 0;
        ans = a+b+c;
        return ans;
    }
```

```
float addition (float a, float b) // Addition @ 2ff
{
    float ans = 0.0f
    ans = a+b;
    return ans;
}
};
```

2] Due to concept of function overloading as a user of Demo class, there is no need to remember names of different functions.

3] When we compile the code, compiler changes the name of every function with the mangled-name (edited name).

4] When we overload the function all the names of function are same but due to the concept of name-mangling, the compiler will edit, the name of every function with new name.

5] According to the above point we conclude that there is no such concept of function overloading after code gets executed

6] For every function there is a seperate address.

7] When we compile the code the CALL instruction gets written which mentions the address of that function.

8] Which function should be called gets decided at time of compilation, due to which it is called as ~~that~~ early-binding.