03/11/23. OOPS - C++ :

Object Oriented Programming.

→ To able to represent real life entity by using class - building block of OOP.

→ class is a more complex data type.

→ create a class :

```
#include <iostream>
using std::string;
class Employee {
public:
    std::string Name;
    std::string company;
    int age;
};
int main() {
    int number;
    Employee employee1;
    employee1.Name = "Veena";
    employee1.company = "Myntra";
    employee1.age = 21;
```

(private by default)

(include in global space)

```
void Intro() {
    std::cout << "Name -" <<
    Name << company << age <<
    std::endl;
}
```

Object

Access modifiers :

→ Private - can't be accessed outside the class.

→ Public - can be accessed outside the class.

→ protected - in b/w private & public

→ Behaviours - class methods.

→ constructor - special type of method.

→ Default constructor is automatically created by the compiler.

3 rules of constructor :

1. Doesn't have a return type.
2. It has the same name of a class
3. It is public (but there can be private constructors also).

Employee (string name, string company,
int age) {
name = name;
company = company;
age = age;
}

Employee employee1 = Employee ("Veena", "Delloitte");
Employee employee2 = Employee ("Bhuvan", "Google", 18);
employee1.Intro();
employee2.Intro();

* 4 pillars of OOPS:-

1. Encapsulation - bundling data and methods. others can't access and modify. We can access encapsulated classes through the methods (public methods).

2. Abstract : Hiding complex things behind a procedure. to make things simpler. Interface concepts are implemented using abstract class.

class AbstractEmployee {
private: string
virtual void AskForPromotion() = 0;
};

Provide implementation in the class.

3. Inheritance - Base class (super/parent) is inherited by child (derived/sub) class and child class can have extra attributes and methods as well.

class Developer : Employee {
}
child parent

members.

→ protected, access modifier can be accessed by all its derived class.

4. Polymorphism:
Polymorphism - Greek origin - multiple forms.
Ability of an object/method to have many forms.
Parent class reference is used to refer to an object of child class.

employee * el = &d; → to derived class object.
↓
base class pointer