Class and Object

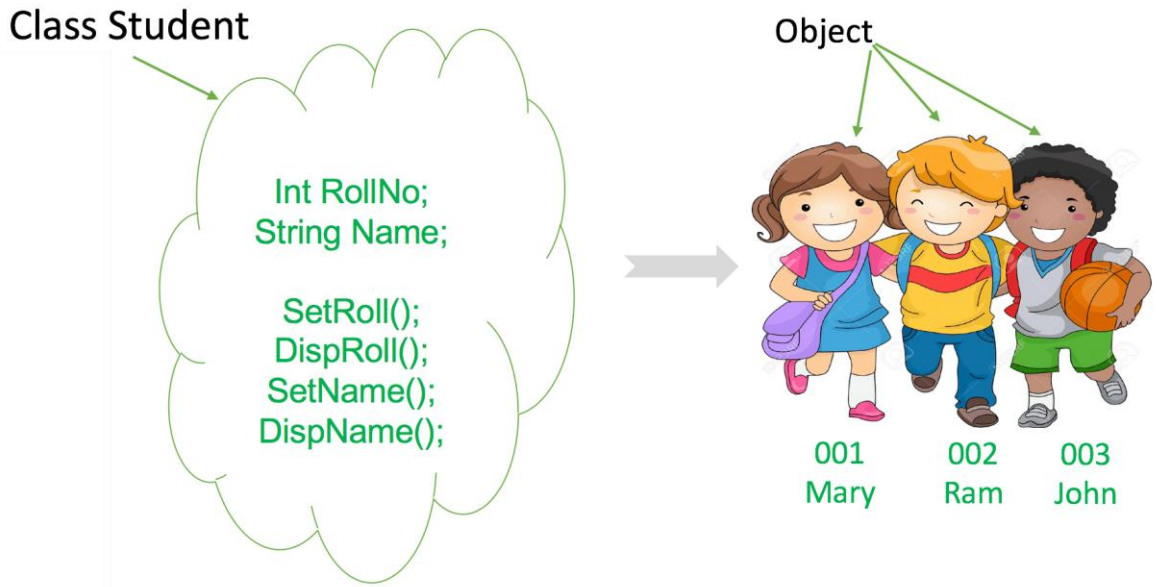
A *class* is a prototype (template) from which objects are created
An *object* is a software bundle of related state and behavior

Student
has
- Last name
- First name
- Age
- List of courses

can
- Pass an exam
- Enroll to course

student1
- Last name - Petrenko
- First name - Ostap
- Age - 19
- List of courses – Java, MQC

student2
- Last name - Romaniv
- First name - Maryna
- Age - 21
- List of courses – Java, ATQC



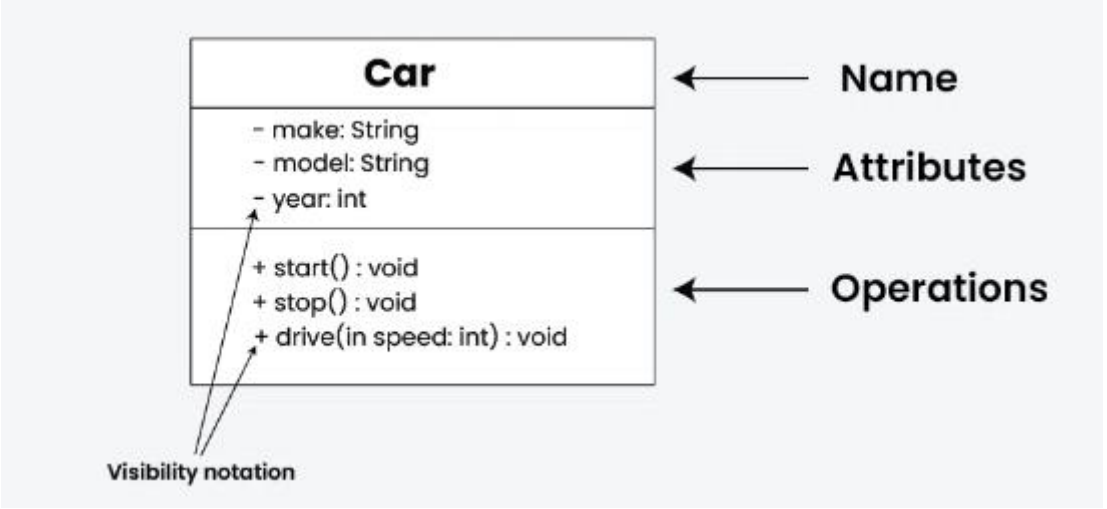

```
class Box
{
  private:
      // data members
  public:
      // member functions
};
```

⇦ Class

Box b1 | Box b2 | Box b3 ⇦ Objects

Unified Modeling Language (UML) – Class Diagram

Class diagrams are a type of UML diagram used in software engineering to visually represent the structure and relationships of classes in a system. UML is a standardized modeling language that helps in designing and documenting software systems. They are an integral part of the software development process, helping in both the design and documentation phases.



Class Definition

Circle
-radius:double=1.0 -color:String="red"
+Circle() +Circle(r:double) +Circle(r:double,c:String) +getRadius():double +getColor():String +getArea():double

Instances

c1:Circle	c2:Circle	c3:Circle
-radius=2.0 -color="blue"	-radius=2.0 -color="red"	-radius=1.0 -color="red"
+getRadius() +getColor() +getArea()	+getRadius() +getColor() +getArea()	+getRadius() +getColor() +getArea()

Access Modifiers

- Access modifiers are used to control access to class members
- **public**, **private** & **protected** are three of the access modifiers available in C++
- In C++, class members are considered **private** when no access modifier is used

Getter/Setter Functions

- Getter functions (or accessor functions) are used to read value of a private member of some class

- Setter functions (or mutator functions) are used to modify the value of a private member of some class

Example – Getter Function

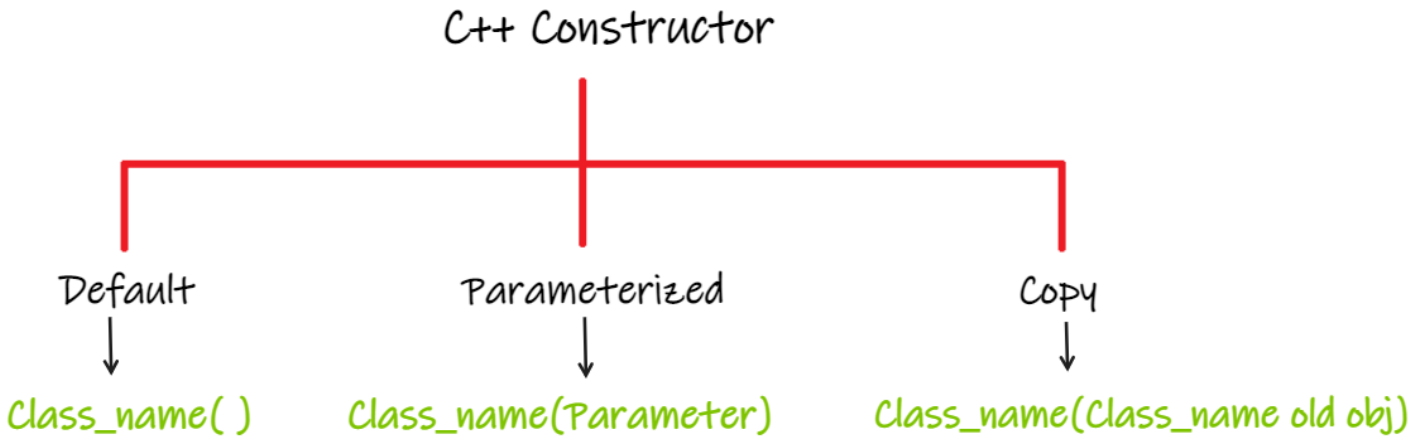
```
class BankAccount
{
    int PIN;            //private variable

    int get_PIN() const
    {
        return PIN;
    }
}
```

Example – Setter Function

```
class BankAccount
{
   int accountNo;        //private variable

   void set_accountNo(int num)
   {
        accountNo = num;
   }
}
```



Constructor & Destructor in C++

```
class Cube
{
  int side;
  public:
    Cube()  // constructor
    {
      cout<<"Constructor Called";
    }
    ~Cube()  // destructor
    {
      cout<<"Destructor Called";
    }
};
```

week_01.cpp

```cpp
#include<iostream>
using namespace std;
class circle{
    float radius;
    public:
        circle(){
            radius=0;
            cout<<"The default constructor is invoked"<<endl;
        }
        circle(float r=0):radius(r){
            cout<<"The parametrized constructor is invoked"<<endl;
        }
        ~circle(){
            cout<<"Destructor is invoked"<<endl;
        }
    float getRadius(){
        return radius;
    }
    float Area(){
        return 3.14*radius*radius;
    }
    float circumference(){
        return 2*3.14*radius;
    }
};

int main(){
    circle c1;
    cout<<"Radius is : "<<c1.getRadius()<<endl;
    cout<<"The area of the circle is : "<<c1.Area()<<endl;
    circle c2(10);
}
```

Data Members

Access Specifier

Default Constructor

Class - Blueprint or template

Parametrized Constructor

Destructor

Getter function

object created by invoking default constructor

Accessing public methods of the object

object created using parametrized constructor