# lesson 29  Operator overloading (OOP)

فى الدرس السابق أخذنا مثال بسيط عن **Object-oriented programming**،
لكن فى هذا الدرس نريد تكبير هذا المثال و سنتعرف على بعض خصائص
**Object-oriented programming**
سنتعرف اليوم على شئ إذا تم كتابته فى الكود سيتم بطريقة، وإذا تم وضعه داخل **class**
الخاص بك يعمل بصورة أخرى، مثل **++** هى تقوم بزيادة الرقم **1** وتسجيله فى المتغير
لكن سنقوم فى هذا الدرس بعمل **++** لكن سيعمل بطريقة أخرى داخل **class**
وهو **operator overloading**
وسيكون البرنامج كالتالي :

```
#include <iostream>
using namespace std;


class Gammal_Tech_member {
private:
  //لا يستطيع اى مبرمج أخر استخدام هذه المتغيرات
  int day, month, year;
public:
  //التى يستطيع اى مبرمج أخر استخدامها method
  bool setDate(int d, int m, int y) {
    if (d >= 1 && d <= 31)
      day = d;
    else
      return false;
    if (m >= 1 && m <= 12)
      month = m;
```

```
  else
    return false;
  if (y >= 2020)
    year = y;
  else
    return false;
  return true;
  //class هنا وضعنا القانون الذي يسير عليه إى نسخة من
}

void operator ++() {
  // ++  هنا وضعنا قاعدة لتغيير
  if (day < 31)
    day++;
  else {
    day = 1;
    if (month < 12)
      month++;
    else {
      month = 1;
      year++;
    }
  }
}
void print() {
  //للطباعة  method
  cout << day << "/" << month << "/" << year << endl;
}
```

```
};


int main() {
  Gammal_Tech_member omar;
  //class نسخة من  object
  //  class تسرى عليه القاعدة الموجودة داخل


  if (omar.setDate(31, 12, 2022) == false)
    cout << "Date is incorrect\n";
  else
    cout << "Date is correct\n";
  // لتحديد هل تم وضع التاريخ بصورة صحيحة أم لا

  omar.print();
  // لطباعة التاريخ قبل الزيادة
  ++omar;
  //تزيد object  جعل
  omar.print();
  // لطباعة التاريخ بعد الزيادة
}
```

**output:**

Date is correct
31/12/2022
1/1/2023

تعلمنا فى هذا الدرس **overloading** وبعض المميزات، عند عمل **class**، فنحن نضع القاعدة التي يسير عليها أى نسخة (**object**) منها، مهما كانت قواعد البرمجة، فأى نسخة من **class** الذى تقوم بعملها، القاعدة التي تسير عليها باقى **objects**