lesson 30 CPP Inheritance

نريد فى هذا الدرس عمل برنامج يقوم بإضافة هدية لمستخدمين موقع **Gammal Tech**
لكن ليس يوم واحد كما كان فى البرنامج السابق، لكن هنا سيكون الهدية على حسب اشتراكه
المشترك السنوى يضيف له سنة ، والمشترك الشهرى يكون له شهر.
يرتبط المشترك السنوي والشهري بالعديد من الأشياء:
كل منهم مشترك فى الموقع ، ولهم تاريخ انتهاء للأشتراك
لكن يختلفوا في تاريخ الأنتهاء ونوع الاشتراك
فى البرنامج السابق كنا نقوم بزيادة يوم واحد باستخدام **operator overloading**،
لكن هنا نريد إذا قمنا بعمل **++** للمشترك الشهرى يقوم بزيادة شهر
وإذا قمنا بعمل ++ للمشترك السنوى يقوم بزيادة سنة
وهنا سنتعرف على **Inheritance** وتعنى الميراث
لدينا **class Gammal Tech member** قاعدة عامة تسري على كل المشتركين،
نريد أضافة **class** للمشترك السنوى و **class** أخر للمشترك الشهري
لكن هنا بدلاً من عمل **Class** من جديد ، سنقوم بعمل **copy** من **class** الاساسى مع اضافة بعد المميزات للمشترك السنوي وكذلك الحال بالنسبة للمشترك الشهرى وذلك باستخدام **Inheritance**
وسيكون البرنامج كالتالي :

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


class Gammal_Tech_member {
private:
  //لا يستطيع أي مبرمج أخر استخدام هذه المتغيرات
  int day, month, year;
```

```
public:
  //التي يستطيع أي 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
  }
  int getDay() {
    return day;
  }
  int getMonth(){
    return month;
  }
  int getYear() {
    return year;
  }
  void print() {
```

```
    //للطباعة  method
    cout << day << "/" << month << "/" << year << endl;
  }
};

class yearly_member :public Gammal_Tech_member {
  //  class Gammal_Tech_member من Inheritance قمنا بعمل
public:
  void operator ++() {
    setDate(getDay(), getMonth(), getYear()+1);
    //  هنا نقوم بعمل + 1 للسنة للمشترك السنوى
  }
};

class monthly_member :public Gammal_Tech_member {
  //  class Gammal_Tech_member من Inheritance قمنا بعمل
public:
  void operator ++() {
    if (getMonth() == 12) {
      setDate(getDay(), 1, getYear() + 1);
      // لو كان الشهر = 12 نقوم بزيادة العام والشهر
    }
    else
    setDate(getDay(), getMonth()+1, getYear() );
    //  هنا نقوم بعمل + 1 للسنة للمشترك الشهرى
  }
};
```

```
int main() {
  monthly_member Aly;
  yearly_member Nader;
  //calss monthly_member قمنا بعمل نسخة من
  //calss yearly_member قمنا بعمل نسخة من
  Aly.setDate(1, 12, 2020);
  Nader.setDate(1, 11, 2021);
  //قمنا بوضع التواريخ
  Aly.print();
  Nader.print();
  // طباعتهم قبل الزيادة
  ++Aly;
  ++Nader;
  Aly.print();
  Nader.print();
  // طباعتهم بعد الزيادة


}
output:
1/12/2020
1/11/2021
1/1/2021
1/11/2022
```

إذن نستفيد من **object oriented programming** أنها تنظم الكود من المبرمجين وبعضهم وتوفر اننا نقوم بعمل **Inheritance** للقواعد (**classes**)التى قام بعملها بعض المبرمجين الآخرين