بسم الله الرحمن الرحيم

Object Oriented Programming

البرمجة الشيئية

عدد الساعات: ٢ نظري + ٢ عملي

المتطلبات السابقة: ٢١١حسب

أستاذة/المادة: م. نجلاء حسن

مميزات البرمجة الشيئية

١. التجريد abstraction
٢. الكبسلة (التغليف) encapsulation
٣. اخفاء البيانات Data Hiding
٤. الوراثة inheritance
٥. التعددية الشكلية polymorphism

١. التجريد abstraction:

وهو عملية تحديد البيانات(الصفات) والعمليات (الوظائف)التي تنتمي لصنف معين وهي نوعان :
أ- تجريد البيانات : Data abstractionوهي عملية التعرف على الخصائص المرتبطة بكائن معين .
ب- تجريد العمليات :Methods abstractionوهو عملية تحديد العمليات والإجراءات دون ذكر شيء عن كيفية أدائها .

٢-الكبسلة (التغليف) encapsulation

- هي عملية تجميع كل الخصائص (الصفات) propertiesوالطرق (الوظائف) Methodsفي وحدة واحدة (داخل غلاف واحد) حيث لا يمكن الوصول إليها إلا عن طريق الكائن .
- تسهل عملية الصيانة لوجود البيانات و العمليات فى مكان واحد (الصنف).
- يتيح للمستخدم استعمال الصنف دون الاهتمام بتفاصيله ،المهم معرفة ما يفعله.

٣- إخفاء البيانات Data Hiding:

هي ميزة ناتجة عن كبسلة البيانات . وتعني إضافة **مستوى حماية** معين إلى البيانات حتى نمنع وصول الخطأ إلـها .

٤- الوراثة inheritance

- هي أن يرث صنف ما الخصائص والعمليات الموجودة في الصنف الآخر مما يساعد على إعادة الاستخدام الأصناف التي تم إنشاؤها من قبل المستخدم .
- توفر الكثير من الوقت للمبرمج وتقطع عنه أشواطاً في تطوير برنامجه.

٥-التعددية الشكلية polymorphism

تسمح ميزة تعدد الأشكال لنفس الدالة أن تتعرف بصورة مختلفة في أصناف مختلفة ويمكن عمل ذلك بـالوراثة مع تعدد الأشكال .
(الاستعمال المتعدد الاغراض للدوال و الادوات مثل +،-،*)

اللغات التي تدعم أسلوب البرمجة الشيئية كثيرة نذكر منها ..

C++و java

مستويات الحماية (محددات الوصول داخل الأصناف):

- **مستوى الحماية العام public :**

يستخدم لتعريف الأعضاء التي يمكن الوصول اليها من خارج الصنف ويمكن توريثها الى صنف آخر.

- **مستوى الحماية الخاص private:**

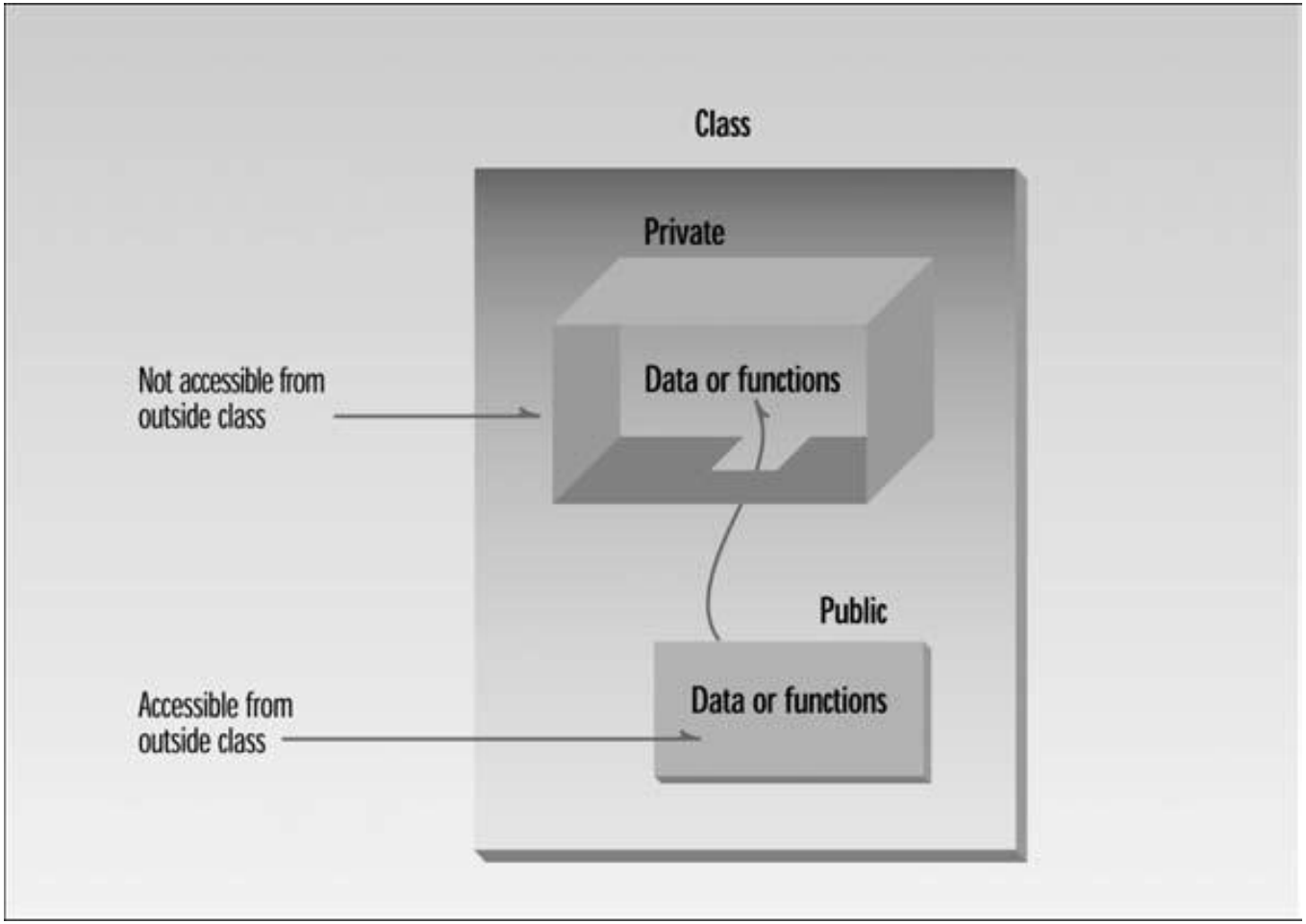
-يستخدم لتعريف الأعضاء التي لا يمكن الوصول اليها من خارج الصنف (هذة من أهم ميزات البرمجة الشيئية ويطلق عليها data hiding).

- **مستوى الحماية محمي Protected :**

يشبه مستوى الحماية الخاص غير أنه يمكن توريثه الى صنف آخر

* ما هي فائدة تحديد مستوى الحماية؟؟؟؟؟

تابع:مستويات الحماية (محددات الوصول داخل الأصناف):



البرمجة الشيئية بلغة C++

مثال لطبقة Student (طالب)

```
class Student                    //اسم الطبقة
{
private:                         //النوع خصوصي
  float grade; //معدل الطالب
protected:                       // النوع محمي
  int X1,X2;
Public:                          // نوع الدوال والبيانات عام
  Student() { }                  //دالة البناء الأولية الفارغة
  Student(int i, char *n, float g); //دالة البناء
  ~Student ();                   //دالة الهدم
  int id; //الرقم الجامعي
  char Name[50]; //اسم الطالب
  void Setgrade(float g) {grade=g;}//ادخال قيمة في متغير خاص
  float Getgrade() {return grade;}//استرجاع قيمة من متغير خاص
};                               //نهاية الطبقة
Student:: Student(int i, char *n, float g); //تعريف دالة البناء
 {
    id=I;
    strcpy(Name,n);
    grade=g;
 }
```

برنامج مبسط: باستخدام الأصناف class اكتبي برنامج لحساب مساحة المستطيل

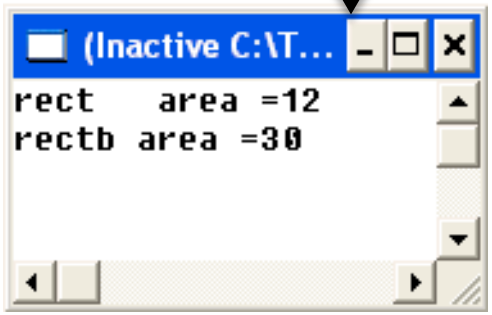
```
#include<iostream.h >
class rectangle
{
private:
int x,y;
public:
void set_values(int a,int b)
{
x=a;
y=b;
}
int area(  ) { return x*y; }
};
void main(  )
{
rectangle rect,rectb;
rect.set_values(3,4);
rectb.set_values(5,6);
cout<<"rect   area ="<<rect.area(  )<<endl;
cout<<"rectb area ="<<rectb.area(  )<<endl;

}
```

اشتقاق كائنات

calling

ناتج التنفيذ



الأصناف (الطبقات) Classes

يعد الصنف في لغة C++ من أهم مميزات اللغة والتي تجعل منها لغة OOP

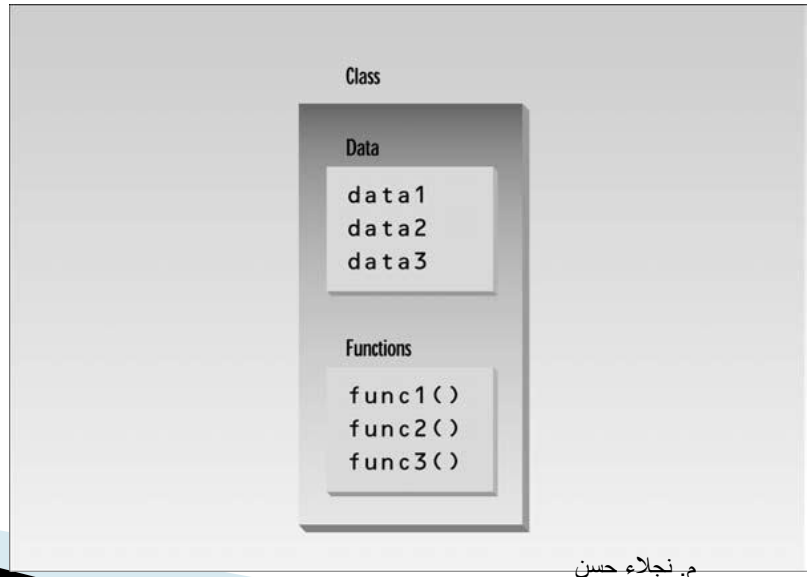
يحتوي الصنف علي:

1) **البيانات الأعضاء :**

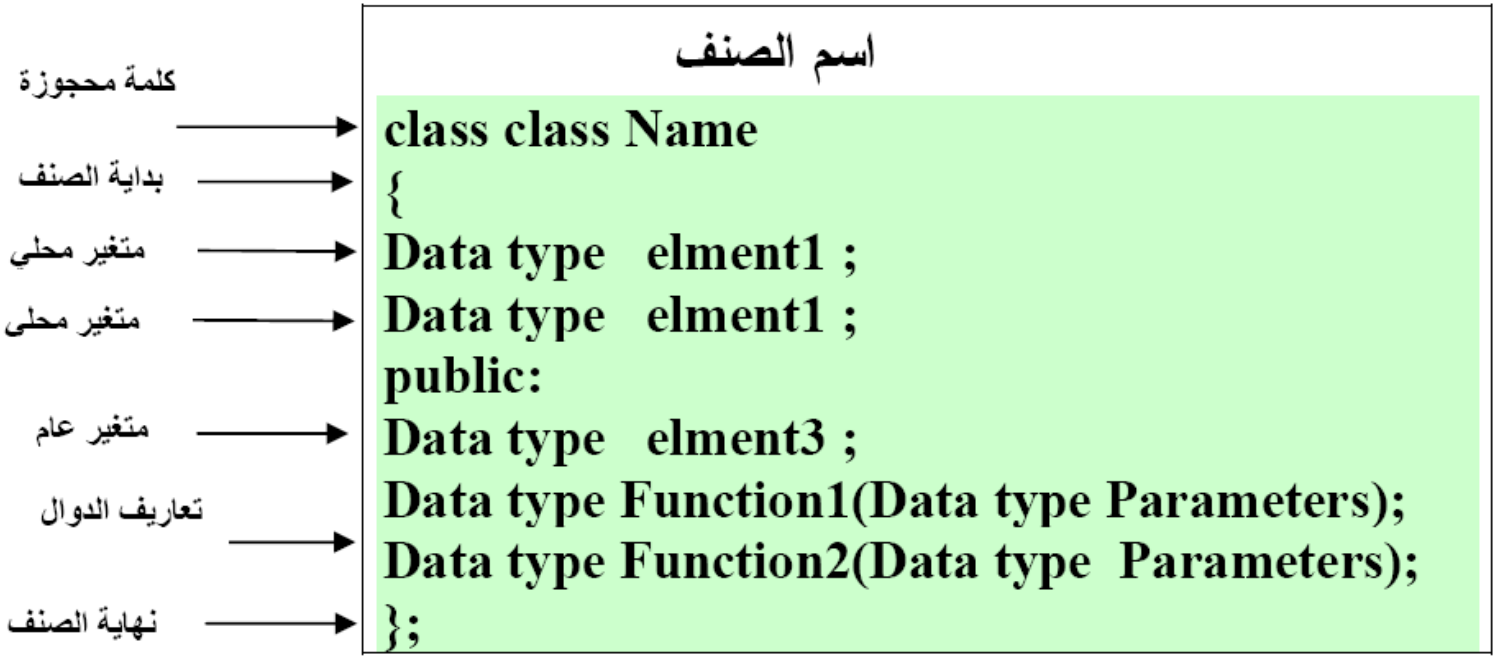
وهي خصائص الصنف .

2) **الدوال الأعضاء :**

هي الوظائف المعرفة على الصنف و تقوم بمعالجة خصائص الصنف.



الصيغة العامة لإنشاء صنف :



مثال

```
Class   rectangle
{
  int   x , y ;
Public:
void  set_values(int , int);
int area ( );
};
```

عبارة عن إعلان عن صنف Class باسم rectangle هذا الصنف يحتوي على أربعة أعضاء

الكائن Object

- نستطيع اشتقاق كائن أو مصفوفة من الكائنات بعد الاعلان عن صنف معينه بما تحتويه من متغيرات ودوال.
- كل كائن من الكائنات التي تنتمي الى صنف معين نستطيع انشائه فارغاً ثم نملأه بالقيم المختلفة أو انشائه محتوياً على قيم أولية أو ملئه مباشرة بالقيم الحقيقة .
- لا نتعامل مع الطبقة /الصنف أو متغيرات أو دوال الطبقة مباشرة ولكن نتعامل مع الكائنات المشتقة من الطبقة والتي تتصف بنفس صفات الطبقة.

اشتقاق الكائنات objects:

يتم تعريف الكائنات في البرنامج الرئيسي وفقاً للصيغة التالية:-

Class name Object name ;

اسم الصنف اسم الكائن

مثلاً إذا أردنا تعريف الكائن rect من نوع الصنف rectangle نكتب الآتي:-

rectangle rect ;

أما إذا أردنا تعريف مجموعة كائنات من نوع صنف معين فيجب أن يفصل بين كل كائن وآخر بواسطة فاصلة(,)

مثلاً إذا أردنا تعريف الكائنات rect1 و rect2 و rect3 من نوع الصنف rectangle نكتب الآتي:-

rectangle rect 1,rect2,rect3 ;

ملاحظة:-

يتم الوصول إلي الأعضاء خارج الصنف باستخدام الكائنات مستخدمين في ذلك أداة النقطة (.)

وفقاً للصيغة التالية:-

Object name. member ;

مثلاً إذا كان لدينا الكائن rect من نوع الصنف rectangle فيمكن استخدامه للوصول للدالة area( )

كالآتي :-

rect. area( );

- الكائن حالة للصنف instance of class لأنه حالة حقيقية لمواصفات الصنف، بمعنى اذا كان لصنف عدة كائنات متشابه فان لكل كائن بياناته المنفصله.
- الصنف تجريد منطقي لا يمثل جزء من الذاكرة الا بعد انشاء كائن من الصنف
- كل كائن من نوع الصنف rectangle يحتوي على نسخة الخاصة من المتغيرات المعرفة في الصنف.

- **يجب التفريق بين تناول متغيرات الصنف (الطبقة)من النو ع prviateوpublic.**
  - **فالمتغيرات من النوع public يمكن تناولها كالتالي (بدون أدنى مشكلة)**

ادخال بيانات كائن:

```
cin >>rec1.x>> rec1.y
```

او اسناد قيمة :

```
rec1.x=2;
rec1.y=3;
```

استخراج بيانات كائن:

```
cout<<rec1.x<< rec1.y;
```

- أما المتغيرات من النو ع prviate لا يمكن تناولها الا من خلال دوال أعدت خصيصاً لذلك:

ادخال بيانات كائن:

```
cin >>x>> y;
rec1.SetSvalue(x,y);
```

أو

```
rec1.SetSvalue(2,3);
```

استخراج بيانات كائن:

```
cout<< rec1.area();
```

أعيدي كتابة البرنامج السابق باستخدام متغيرات من النوع public

```
#include<iostream.h >
class rectangle
{
public:
int x,y;

int area(  ) { return x*y; }
};
void main(  )
{
rectangle rect;
rect.x=2;
rect.y=3;
cout<<"rect   area ="<<rect.area(  )<<endl;


}
```

ما هو ناتج التنفيذ؟؟؟؟

مثال لاستخدام cin لإدخال قيم المتغيرات من النوع public :

```
#include<iostream.h >
class rectangle
{
public:
int x,y;

int area(  ) { return x*y; }
};
void main(  )
{
rectangle rect;
cin>>rect.x>>rect.y;
cout<<"rect   area ="<<rect.area(  )<<endl;

}
```

تصريح وتعريف معاً

كتابة تفاصيل الأعضاء الدالية (نهج) خارج الصنف:

يمكن كتابة تفاصيل دوال الصنف داخلة أو خارجه:

الصيغة العامة لكتابة تفاصيل الدوال الأعضاء خارج الصنف

النوع	اسم الصنف	اسم الدالة

```
Data type  Class_Name::Function_name(Function arguments)
```

مؤثر محدد دقة الوصول (::) scope Resolution Operator

هو عبارة عن مؤثر يستخدم عندما تكتب تفاصيل الدالة خارج نطاق الصنف

مثال int rectangle::area()

مثال:كتابة تفاصيل الأعضاء الدالية (نهج) خارج الصنف بالاضافة الى استخدام cin لإدخال قيم المتغيرات من النوع private :

```
#include<iostream.h>
class rectangle
{
private:
int x,y;
public:
void set_values(int ,int );
int area(  ) ;
};
void rectangle::set_values(int a,int b)
{
x=a;
y=b;
}
int  rectangle::area() {return x*y;}
void main(  )
{
int m,n;
cin>>m>>n;
rectangle rect;
rect.set_values(m,n);
cout<<"rect   area ="<<rect.area(  )<<endl;

}
```

تصريح فقط

تعريف الدالة خارج الصنف