**Al- Mustaqbal University**

**College of Sciences**

**Department of Cybersecurity**

 **Objects and Member Functions in Object-Oriented Programming**

**(OOP)**

Second stage

Lecture 7/ Practical Part

2025-2024

 **Lectuer : Muntather AL-Mousawi**

**الكائنات Objects**

ومن خلالها يمكننا الوصول إلى أعضاء الفئة (البيانات والدوال) (Classes ) هي مثيلات للفئات

(OOP) في البرمجة الكائنية التوجه

فيما يلي شرح للمفاهيم الأساسية المتعلقة بالوصول إلى أعضاء الفئة باستخدام الكائنات :

 **(Public Members) أعضاء الفئة العامة -1**

هي الأعضاء التي يمكن الوصول إليها من خارج الفئة باستخدام الكائنات.

**(Private Members) أعضاء الفئة الخاصة 2-**

الأعضاء الخاصة لا يمكن الوصول إليها مباشرة من خارج الفئة، وتستخدم غالبًا لحماية البيانات من التعديل غير المقصود.

(public membersللوصول إلى هذه الأعضاء أو تعديلها، يتم استخدام دوال الأعضاء العامة (

والتي تقوم بالتفاعل مع البيانات الخاصة بطريقة منظمة.

**Example**

**class MyClass {**

**public:**

 **int x; // عضو عام يمكن الوصول إليه مباشرة**

 **void display() {**

 **std::cout << "Value of x: " << x << std::endl;**

 **}**

**};**

**(1)**

**int main() {**

 **MyClass obj;**

 **obj.x = 10; // الوصول إلى العضو العام مباشرة**

 **obj.display(); // استدعاء دالة العضو العامة**

 **return 0;**

**}**

**Example**

**class MyClass {**

**private:**

 **int x; // عضو خاص**

**public:**

 **void setX(int val) { // دالة عامة لتعيين قيمة العضو الخاص**

 **x = val; // يمكن الوصول إلى 'x' من داخل الفئة**

 **}**

 **int getX() { // دالة عامة لاسترجاع قيمة العضو الخاص**

 **return x;**

 **}**

**};**

**int main() {**

 **MyClass obj; // obj.x = 10; // خطأ! لا يمكن الوصول إلى العضو الخاص مباشرة**

 **obj.setX(10); // يتم الوصول إلى 'x' باستخدام دالة عامة**

 **std::cout << "Value of x: " << obj.getX() << std::endl;**

 **return 0;**

**} (2)**

**3-الوصول إلى الأعضاء باستخدام الكائنات**

**(Dot Operator) ).)** مشغل النقطة

يُستخدم للوصول إلى أعضاء الكائن إذا كان الكائن متغيرًا عاديًا.

**الخلاصة:**

الأعضاء العامة يمكن الوصول إليها مباشرة باستخدام مشغل النقطة (.) أو السهم (-> إذا كان الكائن مؤشرًا).

الأعضاء الخاصة لا يمكن الوصول إليها مباشرة من خارج الفئة، ولكن يمكن التفاعل معها عبر دوال الأعضاء العامة التي توفر واجهة آمنة للتعامل مع البيانات الخاصة.

**Defining Member Functions**

 يمكن تعريف الدوال الأعضاء بطريقتين :C++في لغة

**) Inline Functions) داخل الفئة الدوال المضمنة-1**

يعني أن المترجم قد يستبدل استدعاء الدالة بشفرة الدالة نفسها لتحسين الأداء عن طريق تجنب تكلفة استدعاء الدالة.

**Example**

**#include <iostream>**

**using namespace std;**

**class Rectangle {**

**public:**

 **int length, width;**

 **void setDimensions(int l, int w) {**

 **length = l;**

 **width = w;**

**(3)**

 **}**

 **int calculateArea() {**

 **return length \* width;**

 **}**

**};**

**int main() {**

 **Rectangle rect;**

 **rect.setDimensions(5, 10);**

 **cout << "Area: " << rect.calculateArea() << endl;**

 **return 0;**

**}**

**خارج الفئة (باستخدام معامل النطاق) (::) 2-**

هذه الطريقة مفيدة عندما ترغب في فصل إعلان الدالة في الفئة عن تعريفها، وهو أمر شائع في البرامج الكبيرة.

**Example**

**#include <iostream>**

**using namespace std;**

**class Circle {**

**private:**

 **double radius;**

**public:**

 **void setRadius(double r);**

 **double calculateArea();**

**}; (4)**

**void Circle::setRadius(double r) {**

 **radius = r;**

**}**

**double Circle::calculateArea() {**

 **return 3.14 \* radius \* radius;**

**}**

**int main() {**

 **Circle circle;**

 **circle.setRadius(7);**

 **cout << "Area of the circle: " << circle.calculateArea() << endl;**

 **return 0;**

**}**

**(5)**

**Object as Function Arguments**

يمكن تمرير الكائنات كوسائط إلى الدوال بطريقتين:

 **(By Value)بالقيمة 1-**

يتم نسخ الكائن عندما يتم تمريره إلى الدالة. أي تغييرات يتم إجراؤها على الكائن داخل الدالة لن تؤثر على الكائن الأصلي، حيث يتم التعامل مع نسخة منفصلة منه .

**(By Reference)بالإشارة -2**

(reference)يتم تمرير الكائن الأصلي إلى الدالة، مما يسمح للدالة بتعديل محتوياته. باستخدام الإشارة

 فإن أي تغييرات يتم إجراؤها داخل الدالة تنعكس على الكائن الأصلي.

**Example**

**#include <iostream>**

**using namespace std;**

**class Book {**

**public:**

 **string title;**

 **int pages;**

 **void setDetails(string t, int p) {**

 **title = t;**

 **pages = p;**

 **}**

 **void display() {**

 **cout << "Title: " << title << ", Pages: " << pages << endl;**

 **} (6)**

**};**

**void printBook(Book b) { // Object passed by value**

 **b.title = "New Title";**

 **b.display(); // Changes do not affect the original object**

**}**

**int main() {**

 **Book book1;**

 **book1.setDetails("C++ Programming", 350);**

 **printBook(book1); // Passing by value**

 **book1.display(); // Original object remains unchanged**

 **return 0;**

**}**

**Example**

**#include <iostream>**

**using namespace std;**

**class Book {**

**public:**

 **string title;**

 **int pages;**

 **void setDetails(string t, int p) {**

 **title = t;**

 **pages = p;**

**(7)**

 **}**

 **void display() {**

 **cout << "Title: " << title << ", Pages: " << pages << endl;**

 **}**

**};**

**void modifyBook(Book &b) { // Object passed by reference**

 **b.title = "Advanced C++";**

**}**

**int main() {**

 **Book book1;**

 **book1.setDetails("C++ Programming", 350);**

 **modifyBook(book1); // Passing by reference**

 **book1.display(); // The original object is modified**

 **return 0;**

**(8)**

**Object as Return Type**

في البرمجة، يمكن إرجاع كائن كنوع إرجاع لدالة أو طريقة إرجاع كائن من دالة.

**Example**

**#include <iostream>**

**using namespace std;**

**class Person { // تعريف فئة بسيطة**

**private:**

 **string name;**

 **int age;**

**public:**

 **Person(string n, int a) { // مُنشئ (Constructor)**

 **name = n;**

 **age = a;**

 **}**

 **void display() { // طريقة لعرض تفاصيل الشخص**

 **cout << "الاسم: " << name << ", العمر: " << age << endl;**

 **}**

**};**

**Person createPerson(string name, int age) { // Person دالة تقوم بإرجاع كائن من نوع**

 **return Person(name, age); // Person إنشاء وإرجاع كائن من الفئة**

**}**

**(9)**

**int main() {**

 **Person person = createPerson("John", 25); // استدعاء الدالة وتخزين الكائن المُرجع**

 **person.display(); // استخدام الكائن المُرجع**

 **return 0;**

**} // العمر 25 John الناتج: الاسم**

**Example**

**#include <iostream>**

**using namespace std;**

**class Complex {**

**private:**

 **int real, imag;**

**public:**

 **void setValues(int r, int i) {**

 **real = r;**

 **imag = i;**

 **}**

 **void display() {**

 **cout << "Complex number: " << real << " + " << imag << "i" << endl;**

 **}**

**(10)**

 **Complex add(Complex c) {**

 **Complex result;**

 **result.real = real + c.real;**

 **result.imag = imag + c.imag;**

 **return result;**

 **}**

**};**

**int main() {**

 **Complex c1, c2, sum;**

 **c1.setValues(3, 4);**

 **c2.setValues(5, 6);**

 **sum = c1.add(c2); // Returning an object from the function**

 **sum.display(); // Displaying the result**

 **return 0;**

**}**

**(11)**