

القسم: الحاسبات
تقنية المعلومات
المستوى: ثاني
المقرر: OOP - عملي



الجمهورية اليمنية
وزارة التعليم العالي والبحث العلمي
جامعة الجزيرة
كلية العلوم والهندسة

The Classes

Lect 2

Eng. Abeer Mohammed

الخصائص (**Attributes**) هي الأشياء (المتغيرات, المصفوفات و الكائنات) التي يتم تعريفها بداخل Class و التي سيملك نسخة خاصة منها أي كائن ننشئه منه. أي شيء تنوي تعريفه في Class لا بد لك من تحديد كيفية الوصول إليه كما قلنا سابقاً. لتحديد كيفية الوصول للأشياء التي تضعها في Class يمكنك استخدام الكلمات المخصصة لذلك والتي يقال لها **Access Specifiers**.

الكلمة	استخدامها
public	عند استخدام public، يمكن الوصول إلى العنصر من أي مكان في البرنامج، سواء كان داخل نفس class أو خارجه.
private	عند استخدام private، يمكن الوصول إلى العنصر فقط من داخل نفس class. بمعنى آخر، العناصر التي تحمل private لا يمكن الوصول إليها من خارج class.
protected	عند استخدام protected، يمكن الوصول إلى العنصر من class نفسه ومن أي class يرث منه، ولكن لا يمكن الوصول إليه من خارج class. بمعنى آخر protected تستخدم لتحديد أن الأشياء الموضوعة في Class يمكن الوصول لها عند وراثتها.

Example:

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

class Person {
public:
    string name; // public variable

private:
    int age; // private variable

public:
    void PersonInfo( int a)
    {
        age = a;
    }

    // Declaration of the DisplayInfo function
    void DisplayInfo();
};

// تنفيذ الدالة خارج الكلاس
```

```

void Person::DisplayInfo()
{
    cout << "The name is: " << name << endl;
    cout << "The age is: " << age << endl; // الوصول إلى العمر من داخل الكلاس مسموح به
}

int main()
{
    Person person1;

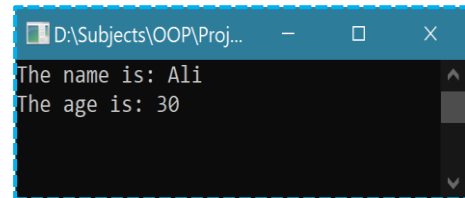
    person1.name = "Ali";

    // person1.age = 25; // سيؤدي إلى حدوث خطأ
    // لا يمكن الوصول إلى العمر من خارج الكلاس لأنه خاص

    person1.PersonInfo(30);
    person1.DisplayInfo();

    cin.get(); // إيقاف الشاشة
    return 0;
}

```



تعريف دالة تأخذ object من class كبارامتر

تمرير object إلى دالة هو أسلوب يسمح للدالة باستقبال (object) من كلاس معين كوسيط، بهدف تنفيذ عمليات عليه مثل القراءة أو التعديل أو الحساب.

متى نستخدم تمرير object ؟

- عندما نريد أن تعمل دالة واحدة مع أكثر من كائن مختلف.
- لتقليل التكرار في الكود.
- لتنظيم البرامج الكبيرة باستخدام البرمجة الكائنية OOP

في المثال التالي، تم تعريف كلاس اسمه Book يمثل المعلومات التي يمكن أن يتضمنها أي كتاب كعنوانه، إسم المؤلف، سعره و عدد صفحاته. بعدها تم تعريف دالة إسمها printInfo عند استدعائها تمرر لها كائن نوعه Book فتقوم بطباعة قيمه بشكل مرتب. في الأخير تم إنشاء كائن من Book و إعطاؤه قيم، و من ثم تمريره للدالة printInfo() حتى تقوم بطباعة قيمه.

```

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

class Book {
public:
    string title;
    string author;
    double price;
    int numberOfPages;
}

```

```

// دالة الإدخال داخل الكلاس
void ReadInfo()
{
    cout << "Enter book title: ";
    getline(cin, title);

    cout << "Enter author name: ";
    getline(cin, author);

    cout << "Enter book price: ";
    cin >> price;

    cout << "Enter number of pages: ";
    cin >> numberOfPages;

    cin.ignore();
}

};

// دالة الطباعة خارج الكلاس
// كبراميتير class من object تأخذ
void PrintInfo(Book text)
{
    cout << "\n--- Book Information ---" << endl;
    cout << "Title: " << text.title << endl;
    cout << "Author: " << text.author << endl;
    cout << "Price: " << text.price << "$" << endl;
    cout << "Number of pages: " << text.numberOfPages << endl;
}

int main()
{
    Book book1, book2;

    cout << "--- Enter information for Book 1 ---" << endl;
    book1.ReadInfo();

    cout << "\n--- Enter information for Book 2 ---" << endl;
    book2.ReadInfo();

    cout << "\n===== \n";
    PrintInfo(book1);

    cout << "\n===== \n";
    PrintInfo(book2);

    cin.get();
    return 0;
}

```

```

D:\Subjects\OOP\Projects\Lect-2\Debug\Lect-2.exe
--- Enter information for Book 1 ---
Enter book title: Cyber Security
Enter author name: Majed Ahmed
Enter book price: 100
Enter number of pages: 750

--- Enter information for Book 2 ---
Enter book title: Internet of Things (IoT)
Enter author name: Hosam Mohammed
Enter book price: 85
Enter number of pages: 600

=====

--- Book Information ---
Title: Cyber Security
Author: Majed Ahmed
Price: 100$
Number of pages: 750

=====

--- Book Information ---
Title: Internet of Things (IoT)
Author: Hosam Mohammed
Price: 85$
Number of pages: 600

```

مثال: برنامج باستخدام classes يتم ادخل بيانات طلاب ودرجاتهم في 3 مقررات ثم حساب المجموع والمعدل لكل طالب (عدد الطلاب المدخل بياناتهم يحدده المستخدم)

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

class Student {
public:
    string name;
    int age;
    string major;
    int grades[3]; // مصفوفة لتخزين درجات الطالب
    int totalGrade;
    double averageGrade;

    void CalculateTotalGrade()
    {
        totalGrade = grades[0] + grades[1] + grades[2];
    }

    void CalculateAverageGrade()
    {
        averageGrade = double(totalGrade) / 3;
    }
};

int main()
{
    int numberOfStudents;

    cout << "Enter the number of students: ";
    cin >> numberOfStudents;

    // التحقق من أن العدد المسموح به هو عدد موجب
    while (numberOfStudents <= 0)
    {
        cout << "Error: Please enter a valid positive number of students: ";
        cout << "Enter number again: ";
        cin >> numberOfStudents;
    }

    Student* students = new Student[numberOfStudents]; // مصفوفة ديناميكية لتخزين معلومات الطلاب

    for (int i = 0; i < numberOfStudents; i++)
    {
        cout << "\nEnter student " << (i + 1) << " details:\n";

        cout << "Name: ";
```

```

cin >> students[i].name;

cout << "Age: ";
cin >> students[i].age;

cout << "Major: ";
cin >> students[i].major;

// إدخال درجات الطالب
cout << "Enter 3 grades:\n";
for (int j = 0; j < 3; j++)
{
    cout << "Grade " << (j + 1) << ": ";
    cin >> students[i].grades[j];
}

students[i].CalculateTotalGrade(); // حساب مجموع الدرجات
students[i].CalculateAverageGrade(); // حساب المعدل
cin.ignore();
}

cout << "\n===== Student Details =====\n";
for (int i = 0; i < numberOfStudents; i++)
{
    cout << "Name: " << students[i].name << endl;
    cout << "Age: " << students[i].age << endl;
    cout << "Major: " << students[i].major << endl;
    cout << "Total Grade: " << students[i].totalGrade << endl;
    cout << "Average Grade: " << students[i].averageGrade << endl;
    cout << "-----\n";
}

delete[] students; // تحرير الذاكرة بعد الإنتهاء

cin.get();
return 0;
}

```

```

D:\Subjects\OOP\Projects\Lect-2\Debug\Lect-2.exe
Enter student 1 details:
Name: Mohammed
Age: 22
Major: IT
Enter 3 grades:
Grade 1: 80
Grade 2: 95
Grade 3: 77

Enter student 2 details:
Name: Ahmed
Age: 21
Major: Cyber
Enter 3 grades:
Grade 1: 90
Grade 2: 74
Grade 3: 88

===== Student Details =====
Name: Mohammed
Age: 22
Major: IT
Total Grade: 252
Average Grade: 84
-----
Name: Ahmed
Age: 21
Major: Cyber
Total Grade: 252
Average Grade: 84
-----

```

نفس البرنامج مع اختلاف جعل ادخال البيانات وطباعتها باستخدام دوال تم انشائها داخل class

```

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

class Student {
public:
    string name;
    int age;

```

```

string major;
int grades[3]; // مصفوفة لتخزين درجات الطالب
int totalGrade;
double averageGrade;

// دالة لإدخال بيانات الطالب
void InputInfo() {
    cout << "\nEnter student details:\n";
    cout << "Name: ";
    cin >> name;

    cout << "Age: ";
    cin >> age;

    cout << "Major: ";
    cin >> major;

    cout << "Enter 3 grades:\n";
    for (int i = 0; i < 3; i++) {
        cout << "Grade " << (i + 1) << ": ";
        cin >> grades[i];
    }
    CalculateTotalGrade(); // حساب مجموع الدرجات
    CalculateAverageGrade(); // حساب المعدل
}

// دالة لحساب المجموع
void CalculateTotalGrade() {
    totalGrade = grades[0] + grades[1] + grades[2];
}

// دالة لحساب المتوسط
void CalculateAverageGrade() {
    averageGrade = static_cast<double>(totalGrade) / 3;
}

// دالة لطباعة معلومات الطالب
void DisplayInfo() {
    cout << "\n--- Student Info ---\n";
    cout << "Name: " << name << endl;
    cout << "Age: " << age << endl;
    cout << "Major: " << major << endl;
    cout << "Total Grade: " << totalGrade << endl;
    cout << "Average Grade: " << averageGrade << endl;
}

};

int main() {

```

```

int numberOfStudents;

cout << "Enter the number of students: ";
cin >> numberOfStudents;

// التحقق من أن العدد المسموح به هو عدد موجب
while (numberOfStudents <= 0)
{
    cout << "Error: Please enter a valid positive number of students: ";
    cout << "Enter number again: ";
    cin >> numberOfStudents;
}

Student* students = new Student[numberOfStudents]; // مصفوفة ديناميكية لتخزين معلومات الطلاب

// استدعاء دالة الإدخال من الكلاس
for (int i = 0; i < numberOfStudents; i++) {
    cout << "\n--- Student " << (i + 1) << " ---";
    students[i].InputInfo();
}

// استدعاء دالة الطباعة من الكلاس
cout << "\n===== All Students =====\n";
for (int i = 0; i < numberOfStudents; i++) {
    students[i].DisplayInfo();
}
delete[] students; // تحرير الذاكرة بعد الإنتهاء
cin.ignore();

cin.get();
return 0;
}

```

المصفوفة الديناميكية: هي نوع من بنية البيانات في البرمجة تستخدم لتخزين مجموعة من العناصر، حيث يكون حجم المصفوفة غير ثابت ويمكن تغييره في وقت التشغيل.

مميزات المصفوفة الديناميكية:

1. **حجم غير ثابت:** - على عكس المصفوفات الثابتة حيث يتم تحديد الحجم عند وقت الترجمة، فإن المصفوفات الديناميكية تتيح لك تحديد الحجم أثناء وقت التنفيذ. هذا يجعلها أكثر مرونة لتناسب احتياجاتك الفعلية، حيث يمكنك طلب حجم أكبر أو أصغر حسب الحاجة.

2. **تخصيص الذاكرة:** - يتم استخدام عمليات تخصيص الذاكرة الديناميكية مثل new في ++C لإنشاء مصفوفة ديناميكية.

3. **تحرير الذاكرة:** - عند الانتهاء من استخدام المصفوفة، يجب تحرير الذاكرة المخصصة لإعادة استخدام تلك الذاكرة. في ++C، يتم ذلك باستخدام delete أو delete[]، يتم ذلك تلقائيًا عند انتهاء نطاق الكائن.

🚩 Constructor

Special Methods in the class

- The same name of the class
- No return type
- It's automatically called when an object is created
- can initialize the object attributes and perform other object initialization tasks.

Constructor عبارة عن دالة مميزة يتم استدعاؤها بشكل تلقائي عند إنشاء object من Class هذه الدالة تجعلك قادر على تمرير قيم أولية لل object بشكل مباشر له لحظة إنشائه.

• نقاط مهمة حول Constructor

- كل Class يتم إنشاؤه يحتوي على Constructor واحد على الأقل و حتى إن لم تقم بتعريف أي Constructor, سيقوم المترجم بإنشاء واحد افتراضي.
- في كل مرة يتم إنشاء object جديد من Class, يجب استدعاء Constructor من Class حتى يتم إنشاء هذا الكائن.
- القاعدة الأساسية عند تعريف Constructor هي أنه يجب أن يحمل نفس إسم Class و يكون نوعه **public**
- في حال قمت بتعريف Constructor, لن يقوم المترجم بإنشاء واحد افتراضي, أي لن يعود هناك Constructor افتراضي.
- يمكنك تعريف أكثر من Constructor. و يمكنك دائماً إنشاء Constructor فارغ, حتى تستخدمه إن كنت لا تريد إعطاء قيم أولية محددة للخصائص عند إنشاء object.

▪ Constructor Types

1. Default Constructor

Default constructors do not take any parameters. If a default constructor is not provided by the programmer explicitly, then the compiler provides a implicit default constructor. In that case, the default values of the variables are 0.

مثال: برنامج لتوضيح استخدام constructor الافتراضي

```
#include <iostream>
using namespace std;
class Wall {
private:
    double length;

public:
```

```

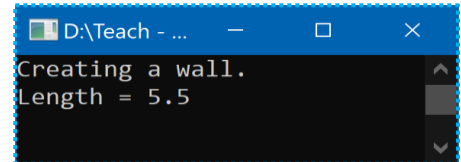
// default constructor to initialize variable
Wall()
{
    length = 5.5;
    cout << "Creating a wall." << endl;
    cout << "Length = " << length << endl;
}

};

int main()
{
    Wall w;

    cin.get();
    return 0;
}

```



2. Parameterized Constructor

A constructor with parameters is known as a parameterized constructor. This is the preferred method to initialize member data.

Example -1:

```

#include <iostream>
using namespace std;

class Wall {
private:
    double length;
    double height;

public:
    // parameterized constructor to initialize variables
    Wall(double len, double hgt)
    {
        length = len;
        height = hgt;
    }

    double CalculateArea()
    {
        return length * height;
    }
};

int main()
{
    // create object and initialize data members
    Wall wall1(10.5, 8.6);
    Wall wall2(8.5, 6.3);
}

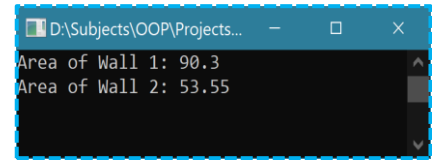
```

```

cout << "Area of Wall 1: " << wall1.CalculateArea() << endl;
cout << "Area of Wall 2: " << wall2.CalculateArea() << endl;

cin.get();
return 0;
}

```



مثال على استخدام (Classes) و (Constructor) لطباعة بيانات موظفين

```

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

class Employees {
public:
    string Name;
    string Department;
    double Salary;
    int YearsOfExperience;

    // لتعيين القيم الابتدائية (constructor)
    Employees(string name, string department, double salary, int yearsOfExperience)
    {
        Name = name;
        Department = department;
        Salary = salary;
        YearsOfExperience = yearsOfExperience;
    }

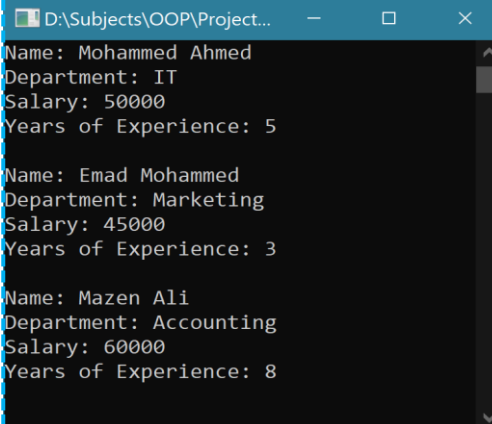
    void PrintInfo()
    {
        cout << "Name: " << Name << endl;
        cout << "Department: " << Department << endl;
        cout << "Salary: " << Salary << endl;
        cout << "Years of Experience: " << YearsOfExperience << endl;
    }
};

int main() {
    Employees employee1("Mohammed Ahmed", "IT", 50000.0, 5);
    Employees employee2("Emad Mohammed", "Marketing", 45000.0, 3);
    Employees employee3("Mazen Ali", "Accounting", 60000.0, 8);
}

```

```
// طباعة معلومات الموظفين
employee1.PrintInfo();
cout << endl;
employee2.PrintInfo();
cout << endl;
employee3.PrintInfo();

cin.get();
return 0;
}
```



```
D:\Subjects\OOP\Project...
Name: Mohammed Ahmed
Department: IT
Salary: 50000
Years of Experience: 5

Name: Emad Mohammed
Department: Marketing
Salary: 45000
Years of Experience: 3

Name: Mazen Ali
Department: Accounting
Salary: 60000
Years of Experience: 8
```

Constructors outside the class

constructors can also be defined outside the class. First, declare the constructor inside the class, and then define it outside of the class by specifying the name of the class, followed by the scope resolution `::` operator, followed by the name of the constructor (which is the same as the class):

```
#include <iostream>
#include <string>
using namespace std;
class Car {
public:
    string brand;
    string model;
    int year;

    Car(string x, string y, int z); // Constructor declaration
};
// Constructor definition outside the class
Car::Car(string x, string y, int z)
{
    brand = x;
    model = y;
    year = z;
}
int main()
{
    // Create Car objects and call the constructor with different values
```

```

Car car1("BMW", "X5", 2010);
Car car2("Ford", "Mustang", 2022);

// Print values
cout << " The brand of car1: " << car1.brand << " " << " Model: " << car1.model << " " << "
Year: " << car1.year << "\n";
cout << " The brand of car2: " << car2.brand << " " << " Model: " << car2.model << " " << "
Year: " << car2.year << "\n";

cin.get();
return 0;
}

```

The screenshot shows a terminal window with the following output:

```

D:\Teach - 2\AL-Jazeera\OOP\C++_Projects\OOP-L...
The brand of car1: BMW Model: X5 Year: 2010
The brand of car2: Ford Model: Mustang Year: 2022

```

❖ برنامج لحساب فاتورة الكهرباء

```

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

// تعريف كلاس يمثل فاتورة الكهرباء
class ElectricityBill {
private:
    string customerName;    // اسم العميل
    double consumedKwh;     // كمية الاستهلاك بالكيلو وات
    double billAmount;      // قيمة الفاتورة المحسوبة

public:
    // (constructor) لتعيين القيم الابتدائية
    ElectricityBill(string name, double kwh) {
        customerName = name;
        consumedKwh = kwh;
        billAmount = 0; // يتم التعيين لاحقا عند حساب الفاتورة
    }

    // دالة لحساب قيمة الفاتورة حسب كمية الاستهلاك
    void CalculateBill() {
        if (consumedKwh <= 100)
            billAmount = consumedKwh * 0.5; // تعرفة ثابتة لأول 100 ك.و.س
        else
            billAmount = (100 * 0.5) + ((consumedKwh - 100) * 0.75); // بعد 100 ك.و.س ترتفع التعرفة
    }
}

```

```

// دالة لطباعة تفاصيل الفاتورة
void DisplayBill()
{
    cout << "\n===== \n\n";

    cout << "Customer Name: " << customerName << endl;
    cout << "Consumed Energy : " << consumedKwh << " (kWh)" << endl;
    cout << "Total Bill Amount: " << billAmount << " $" << endl;

    cout << "\n===== \n\n";
}

};

int main() {
    string name;           // لتخزين اسم العميل
    double kwh;            // لتخزين الاستهلاك بالكيلو واط

    cout << "Enter customer name: ";
    getline(cin, name);

    cout << "Enter consumed energy in kWh (must be > 0): ";
    cin >> kwh;

    // التحقق من صحة الإدخال (الاستهلاك يجب أن يكون رقم موجب)
    while (kwh <= 0) {
        cout << "Invalid consumption value! Please enter a positive number." << endl;
        cout << "Enter consumed energy in kWh (must be > 0): ";
        cin >> kwh;
    }

    // إنشاء كائن من الكلاس وتخزين البيانات فيه
    ElectricityBill customer(name, kwh);
    // حساب الفاتورة وعرض النتائج
    customer.CalculateBill();
    customer.DisplayBill();

    system("pause");
    return 0;
}

```

```

D:\Subjects\OOP\Projects\Lect-2\Debug\Lect-2.exe
Enter customer name: Osama
Enter consumed energy in kWh (must be > 0): 150

=====

Customer Name: Osama
Consumed Energy : 150 (kWh)
Total Bill Amount: 87.5 $

=====

Press any key to continue . . .

```

Destructor

A destructor works opposite to constructor; it destructs the objects of classes. It can be defined only once in a class. Like constructors, it is invoked automatically. A destructor is defined like constructor. It must have same name as class. But it is prefixed with a tilde sign (~).

■ نقاط مهمة حول Destructor

- كل class يتم إنشاؤه يحتوي على destructor. أي حتى إن لم تقم بتعريف destructor بنفسك سيقوم المترجم بإنشاء واحد افتراضي عنك.
- destructor يتم استدعاؤه بشكل تلقائي عندما يتم مسح ال object من الذاكرة.
- القاعدة الأساسية عند تعريف destructor هي أنه يجب أن يحمل نفس إسم class و قبله رمز ~ و يكون نوعه **public**
- يمكنك تعريف destructor واحد فقط في class
- لا يمكن وضع باراميترات في destructor

متى يتم استدعاء destructor؟

- في حال تم إنشاء object بداخل دالة وتم تنفيذ كل محتوى الدالة لأنه من الطبيعي أن يتم حذف أي شيء تم تعريفه فيها بعد أن تنتفذ.
 - في حال تم تنفيذ جميع أوامر البرنامج الموضوعة بداخل الدالة **main()** لأنه من الطبيعي أن يتم حذف أي شيء تم تعريفه فيها بعد أن تنتفذ.
 - في حال تم حذف object من الذاكرة بأي طريقة أخرى.
- الغرض من destructor هو القيام بأي عمليات تنظيف أو إطلاق للموارد المخصصة لل object قبل إتلافه.

```
#include <iostream>
using namespace std;
class Employee {
public:
    Employee()
    {   cout<<"Constructor Invoked"<<endl;   }

    ~Employee()
    {   cout<<"Destructor Invoked"<<endl;   }
};

int main(){
{
    Employee e1, e2;

    cout << "End of block" << endl;
    cin.get(); }
```

