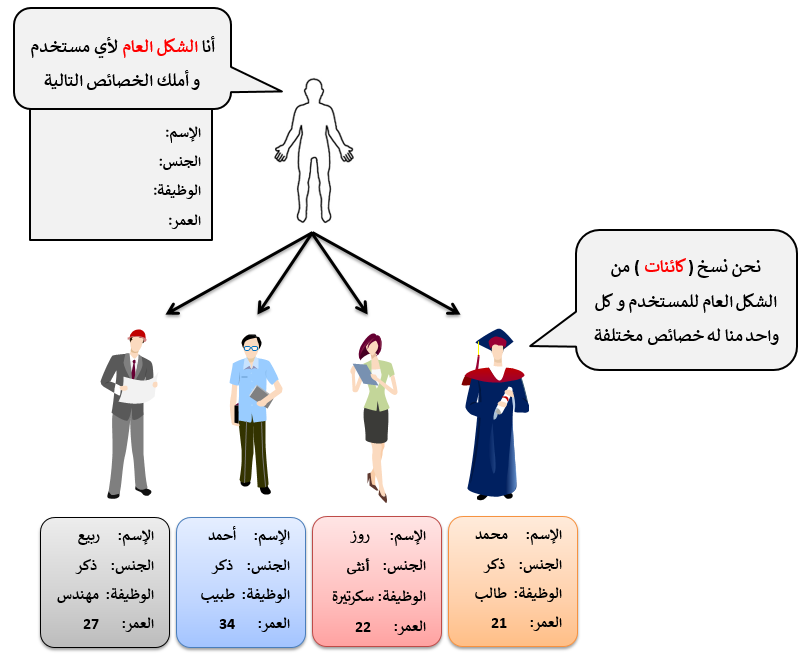
**مفهوم البرمجة الكائنية (OOP)**

**البرمجة الكائنية ( Object Oriented Programming ) تختصر بكلمة OOP و هي عبارة عن أسلوب نتبعه في كتابة الكود لجعل كتابة الكود أكثر سهولة.  
إذاً البرمجة الكائنية هي مجرد أسلوب في العمل لا أكثر و هي ليست خاصة بلغة C++ حيث أنها تطبق في باقي لغات البرمجة.**

**فكرة البرمجة الكائنية بشكل عام هي تجهيز الشكل الذي سيتم فيه حفظ المعلومات مما يجعل الوصول إليها و التعديل عليها سهل للغاية.**

**كمثال بسيط, إذا كنت تنوي بناء برنامج لحفظ معلومات المستخدمين, ستقوم بتجهيز الشكل العام للمعلومات التي تنوي حفظها لكل مستخدم.  
بعدها, أي مستخدم جديد تنوي إنشاؤه تجعله نسخة من الشكل العام لأي مستخدم و تجعله يدخل القيم الخاصة به كما في الصورة التالية.**

**الآن, عليك معرفة أن النوع الجديد أو الشكل الذي تقوم بتجهيزه بهدف إنشاء نسخ منه لاحقاً يقال له النسخة الخام ( Blue Print ).  
كلمة النسخة الخام يقصد بها النسخة الأصلية التي يتم تجهيزها بهدف إنشاء نسخ منها.**

**في C++ لديك خيارين لإنشاء نوع جديد و هما:**

* **أن تنشئه بواسطة الكلمة struct**
* **أن تنشئه بواسطة الكلمة class**

**ملاحظة: ستعرف الفرق بين هاتين الكلمتين لاحقاً في الدورة.**

**ما هو الكائن؟**

**بشكل عام, الكائن ( Object ) عبارة عن نسخة من نوع محدد تم تعريفه بالأساس بواسطة الكلمة struct أو الكلمة class.**

**أمثلة واقعية عن دور الكائنات**

**من أكثر أنواع المشاريع التي قد تجد أنك تتعامل فيها مع عدد هائل من الكائنات هو برمجة الألعاب.**

**كمثال بسيط, في أي لعبة تلعبها تجد أنه يوجد شرير و لكن هذا الشرير قد يظهر لك نفسه مئات المرات في اللعبة.  
أيضاً في بعض الألعاب تجد مطر يتساقط, و للدقة أكثر فإنك تجد نفس قطرة المطر تنزل في أماكن مختلفة من الشاشة.**

**إذاً في الألعاب, يتم تجهيز الشرير في الأصل مرة واحدة و كلما أرادوا أن يظهروا لك نفس الشرير, يقوموا بإنشاء نسخة منه فقط.  
و حتى بالنسبة للمطر فإنه أيضاً يمكن تجهيز قطرة مطر واحدة و كلما أرادوا أن يظهروا لك مطر, يقوموا بإنشاء عشرات النسخ منها و إظهارها في أماكن مختلفة.**

**مفهوم التغليف**

**التغليف ( Encapsulation ) عبارة عن أسلوب يمكن اتباعه لإخفاء خصائص الكلاس ( Global Variables ) و جعل الكائنات التي تنشئها منه و الكلاسات الأخرى التي تقوم بتضمينه قادرة على التعامل مع هذه الخصائص فقط من خلال دوال يقوم بإنشائها المبرمج الأساسي للكلاس.**

**إذاً فائدة التغليف أنه يتيح لك وضع شروط لتخزين القيم في الخصائص, كما أنه يتيح لك الحصول على القيم بالطريقة التي تريدها.**

**الأسلوب المتبع في عملية التغليف :**

**بما أن فكرة التغليف الأساسية هي إخفاء البيانات من جهة و إتاحة التعامل معها من جهة أخرى.**

**أول ما يجب أن يخطر في بالك هو أنه يجب جعل نوع جميع الخصائص ( أي المتغيرات التي ستحفظ البيانات ) الموجودة في الكلاس private لأن تعريف جعلها private يعني أنه يمكن الوصول إليهم فقط من داخل الكلاس الموجودين فيه.**

**ثاني شيىء عليك التفكير فيه هو إيجاد طريقة للوصول إلى هذه الخصائص من الخارج.  
لفعل ذلك عليك تجهيز دوال نوعها public للتعامل مع هذه الخصائص, لأن الدوال التي نوعها public يمكن الوصول إليهم من أي مكان.**

**إذاً لتحقيق مبدأ التغليف, عليك جعل نوع الخصائص private و جعل نوع الدوال التي تستخدم للوصول إليهم public.**

**مفهوم دوال الـ Setter و الـ Getter :**

**عند التعامل مع أي متغير ( أو خاصية ) فعندك خيارين و هما إما إعطاءه قيمة جديدة و إما الحصول على القيمة الموجودة فيه.  
بما أنه يجب بناء دوال للتعامل مع كل خاصية من الخصائص الموجودة في الكلاس, ينصح بإعتماد أسماء متعارف عليها كالتالي:**

* **إبدأ إسم كل دالة الهدف منها إعطاء قيمة للخاصية بالكلمة set ثم إسم الخاصية.**
* **إبدأ إسم كل دالة الهدف منها الحصول على قيمة الخاصية بالكلمة get ثم إسم الخاصية**.

**أمثلة شاملة حول التغليف**

**الآن سنقوم بإنشاء كلاس إسمه Employee و فكرته تخزين معلومات الموظفين مثل الإسم name, الراتب salary, العمر age.  
بعدها سنقوم بتجربة إنشاء كائن من الكلاس Employee في الدالة main() و التعامل معه.**

**ملاحظة: هنا قمنا بتعريف كل شيء بشكل عادي بدون تطبيق أي مبدأ من مبادئ التغليف.**

**المثال الأول**

**المثال الأول**

**#include <iostream>**

**using namespace std;**

**// يحتوي على 3 خصائص Employee هنا قمنا بتعريف كلاس إسمه**

**class Employee {**

**public:**

**string name; // لأنه عبارة عن نص string الإسم نوعه**

**int age; // لأنه عبارة عن رقم int العمر نوعه**

**double salary; // لأنه عبارة عن رقم كبير يمكن أن يحتوي على فاصلة double الراتب نوعه**

**};**

**// main() هنا قمنا بتعريف الدالة**

**int main()**

**{**

**// e إسمه Employee هنا قمنا بإنشاء كائن من الكلاس**

**Employee e;**

**// e هنا قمنا بإعطاء قيم لخصائص الكائن**

**e.name = "Walid";**

**e.age = 21;**

**e.salary = 950;**

**// e هنا قمنا بعرض قيم خصائص الكائن**

**cout << "Name: " << e.name << "\n";**

**cout << "Age: " << e.age << "\n";**

**cout << "Salary: " << e.salary;**

**}**

**و النتسجة ستكون كالتالي :**

**Name: Walid**

**Age: 21**

**Salary: 950**

**الآن سنقوم بإعادة المثال السابق مع جعل نوع الخصائص private و سنقوم بتعريف دوال نوعها public للتعامل مع هذه الخصائص.**

**ملاحظة: في هذا المثال قمنا بتطبيق مبدأ التغليف**

**#include <iostream>**

**using namespace std;**

**// Employee هنا قمنا بتعريف الكلاس**

**class Employee {**

**// و بالتالي لم يعد ممكناً الوصول لهم بشكل مباشر من خارج الكلاس private لاحظ أننا جعلنا نوع الخصائص**

**private:**

**string name;**

**int age;**

**double salary;**

**// حتى نستطيع من الوصول من خارج الكلاس public هنا قمنا بتعريف جميع الدوال التي سنتعامل من خلالها مع الخصائص كـ**

**public:**

**// name هذه الدالة ترجع قيمة المخزنة الخاصية**

**string getName() {**

**return name;**

**}**

**// age هذه الدالة ترجع قيمة المخزنة الخاصية**

**int getAge() {**

**return age;**

**}**

**// salary هذه الدالة ترجع قيمة المخزنة الخاصية**

**double getSalary() {**

**return salary;**

**}**

**// name هذه الدالة نعطيها إسم فتقوم بوضعه للخاصية**

**void setName(string n) {**

**name = n;**

**}**

**المثال الثاني**

**// age هذه الدالة نعطيها رقم فتقوم بوضعه للخاصية**

**void setAge(int a) {**

**age = a;**

**}**

**// salary هذه الدالة نعطيها رقم فتقوم بوضعه للخاصية**

**void setSalary(double s) {**

**salary = s;**

**}**

**};**

**// main() هنا قمنا بتعريف الدالة**

**int main()**

**{**

**// e إسمه Employee هنا قمنا بإنشاء كائن من الكلاس**

**Employee e;**

**// Setter من خلال دوال الـ e هنا قمنا بوضع قيم لخصائص الكائن**

**e.setName("Walid");**

**e.setAge(21);**

**e.setSalary(950);**

**// Getter من خلال دوال الـ e هنا قمنا بعرض قيم خصائص الكائن**

**cout << "Name: " << e.getName() << "\n";**

**cout << "Age: " << e.getAge() << "\n";**

**cout << "Salary: " << e.getSalary();**

**}**

**سنحصل على النتيجة التالية عند التشغيل.**

**Name: Walid**

**Age: 21**

**Salary: 950**

**الآن سنعيد المثال السابق مع إضافة شرط على الدالة setName() يحدد أنه عند إدخال الإسم يجب أن يتألف من أكثر من حرفين.  
و إضافة شرط على الدالة getName() لجعلها ترجع Not defined إذا كان لا يوجد إسم مدخل في الخاصية name.**

**ملاحظة: قمنا بتعليم الشروط و التعديلات الجديدة باللون الأصفر.**

**#include <iostream>**

**using namespace std;**

**// Employee هنا قمنا بتعريف الكلاس**

**class Employee {**

**// private لاحظ أننا أبقينا نوع الخصائص**

**private:**

**string name;**

**int age;**

**double salary;**

**public:**

**// name في حال لم يتم يكن هناك إسم مدخل في الخاصية "Not defined" لجعلها ترجع العبارة getName() هنا وضعنا شرط في الدالة**

**string getName() {**

**if (name == "")**

**{**

**return "Not defined";**

**}**

**else**

**{**

**return "Mr." + name;**

**}**

**}**

**// لم نجري أي تعديل على هذه الدالة**

**int getAge() {**

**return age;**

**}**

**// لم نجري أي تعديل على هذه الدالة**

**double getSalary() {**

**return salary;**

**}**

**المثال الثالث**

**// فقط في حال كان الإسم الذي نمرره لها يتألف من أكثر من حرفين name لجعلها تقبل تخزين الإسم في الخاصية setName() هنا وضعنا شرط في الدالة**

**void setName(string n) {**

**if (n.length() < 3)**

**{**

**cout << "Name is too short, name can't be less then 3 characters!\n";**

**}**

**else**

**{**

**name = n;**

**}**

**}**

**// لم نجري أي تعديل على هذه الدالة**

**void setAge(int a) {**

**age = a;**

**}**

**// age هذه الدالة نعطيها رقم فتقوم بوضعه للخاصية**

**void setSalary(double s) {**

**salary = s;**

**}**

**};**

**// main() هنا قمنا بتعريف الدالة**

**int main()**

**{**

**// e إسمه Employee هنا قمنا بإنشاء كائن من الكلاس**

**Employee e;**

**e.setName("dj"); // كإسم لأنه يتألف من حرفين فقط "dj" هنا لن يقبل أن ندخل**

**e.setAge(21);**

**e.setSalary(950);**

**cout << "Name: " << e.getName() << "\n"; // لأن الإسم الذي أدخلناه سابقاً لم يتم قبلوه "Not defined" هنا يفترضي أن ترجع الدالة عبارة**

**cout << "Age: " << e.getAge() << "\n";**

**cout << "Salary: " << e.getSalary();**

**}**

**سنحصل على النتيجة التالية عند التشغيل.**

**Name is too short, name can't be less then 3 characters!**

**Name: Not defined**

**Age: 21**

**Salary: 950**

**لاحظ أنه لم يقبل الإسم الذي أدخلناه عن طريق الدالة setName() لأنه أصغر من ثلاثة أحرف, لذلك طبع الرسالة التي قمنا بتجهيزها في حال تم إدخال إسم أصغر من ثلاثة أحرف.**

**و بما أنه لم يضع الإسم الذي قمنا بإدخاله في الخاصية name لاحظ أن الدالة getName() قامت بإرجاع القيمة Not defined  التي قمنا بتجهيزها في حال لم يتم إدخال أي إسم فيه.**