

**Assignment Of Object-Oriented Programming Course**

**Technology Institute of Computer**

**Object Oriented Programming: IPG-203**

**Dr. Mohammed khabaz**

**Term: F24**

|  |  |  |
| --- | --- | --- |
| **Prepared by:** | | |
| **الاسم:** | **المُعرف الجامعي:** | **الصف:** |
| **مايا غسان اسحق** | **Maya\_247412** | **C7** |
| **عبادة محمد نور الدين سوقية** | **Obada\_soukiah\_292577** | **C7** |
| **رغد سمير الشيخ** | **Raghad\_300342** | **C7** |
| **مريم عماد دركوش** | **Mariam\_253679** | **C7** |
| **رغد وسام ابوليلى** | **Raghad\_192187** | **C7** |

**المقدمة:**

**يهدف هذا المشروع إلى تطوير نظام مكتبة يطبق مبادئ البرمجة الغرضية التوجه (OOP) بشكل متكامل.  
تم اختيار فكرة المكتبة لأنها:**

1. **تغطي جميع مبادئ OOP بشكل عملي:** التجريد، الوراثة، تعدد الأشكال، التغليف، التفويض والأحداث، والفئات الساكنة.
2. **مجال عملي وواقعي:** يمكن عرض خصائص مختلفة لكل نوع عنصر (كتب، أقراص، مجلات) بطريقة سهلة الفهم.
3. **قابلية التوسع:** يمكن لاحقًا إضافة ميزات جديدة مثل البحث، تقارير الاستعارة، الرسائل التلقائية للمستعيرين، وغيرها.

**شرح مبادئ OOP المستخدمة:**

**1 ـ التجريد (Abstraction):**

**التجريد يظهر في هذا المشروع من خلال الواجهة (Interface) ILibraryItem التي تعرّف العمليات الأساسية لأي عنصر مكتبة مثل Borrow(), Return(), و GetInfo()هذه الواجهة تحدد ما يجب على أي عنصر مكتبة أن يوفّره من وظائف دون الخوض في تفاصيل كيفية تنفيذها.**

**بالإضافة لذلك، يوجد فئة مجردة (Abstract Class) باسم LibraryItem تقوم بتنفيذ أجزاء مشتركة بين كل عناصر المكتبة (كالحفاظ على العنوان والسنة والمعرّف الفريدID ) وتفرض وجود دالة مجردة CalculateLateFee() يجب على الفئات الفرعية إعادة تعريفها.**

**وبهذا يتحقق الفصل بين ما هو مطلوب وبين كيفية التنفيذ الفعلي.**

**2 ـ الوراثة وتعدد الأشكال (Inheritance & Polymorphism):**

**الوراثة تظهر بوضوح في الفئات Book و DVD و Magazine التي ترث جميعها من الفئة المجردة LibraryItem هذه الفئات تعيد تعريف الدالة CalculateLateFee() بحيث يختلف حساب الرسوم المتأخرة من نوع لآخر: الكتب لها معدل معيّن، والأقراص DVD معدل آخر، والمجلات معدل مختلف.  
تعدد الأشكال (Polymorphism) يتحقق عندما نخزن كل العناصر في قائمة واحدة من نوع LibraryItem، ثم نستدعي نفس الدالة CalculateLateFee() أو GetInfo()، فيتم استدعاء التنفيذ المناسب للفئة الفعلية أثناء التشغيل (Run-time) هذا يسمح بالتعامل مع كل العناصر بشكل عام دون الحاجة لمعرفة نوعها الحقيقي.**

**3 ـ التغليف (Encapsulation):**

**التغليف يظهر من خلال جعل الحقول داخل الفئة LibraryItem خاصة (private) مثل \_title و\_year وisBorrowed ، وعدم السماح بالوصول إليها مباشرة من الخارج.**

**بدلاً من ذلك يتم الوصول عبر خصائص (Properties) مثل Title و Year و IsBorrowed هذه الخصائص تحتوي على منطق تحقق (Validation) باستخدام الفئة الساكنة Validator، مما يضمن أن الكائن يبقى في حالة صحيحة دائمًا (مثلاً لا يمكن أن يكون العنوان فارغًا أو قصيرًا جدًا). كما أن خاصية ID للمعرّف الفريد معرفة للقراءة فقط (read-only) بحيث لا يمكن تعديلها بعد الإنشاء. هذا يعكس مبدأ التغليف الذي يحمي البيانات الداخلية ويحدد كيفية الوصول إليها.**

**4 ـ التفويض والأحداث (Delegates & Events):**

**المشروع يحتوي على تفويض (Delegate) باسم AvailabilityChangedHandler وحدث (Event) باسم AvailabilityChanged داخل الفئة LibraryItem عند استعارة عنصر (Borrow) أو إرجاعه (Return)، يتم إطلاق الحدث بواسطة الدالة OnAvailabilityChanged مع تمرير كائن AvailabilityChangedEventArgs يحتوي رسالة ووقت التغيير.  
عند الاشتراك في الحدث من قبل كائن خارجي مثل LibraryCatalog أو من خلال البرنامج الرئيسي، يتم إشعار المشترك فورًا بتغير حالة العنصر (مستعار/متاح).**

**هذا يحقق مبدأ الأحداث والتفويضات حيث يتم فصل منطق العنصر نفسه عن منطق التفاعل مع التغييرات.**

**5 ـ الفئات والأعضاء الساكنة (Static Classes & Static Members):**

**تم استخدام فئة ساكنة (Static Class) باسم Validator تحتوي على دوال تحقق مثل IsValidTitle() و IsValidYear() وIsValidISBN() كونها ساكنة يعني أنه يمكن استدعاء هذه الدوال مباشرة دون الحاجة لإنشاء كائن من Validator.**

**كذلك الفئة LibraryItem تحتوي على خاصية ساكنة TotalItems تُستخدم لحساب عدد العناصر التي تم إنشاؤها في النظام. هذه الخاصية يتم زيادتها تلقائيًا في كل مرة يتم إنشاء عنصر جديد من أي فئة فرعية.**

**هذا يوضح كيفية استخدام الأعضاء الساكنة لتتبع حالة عامة تخص كل الكائنات مجتمعة.**

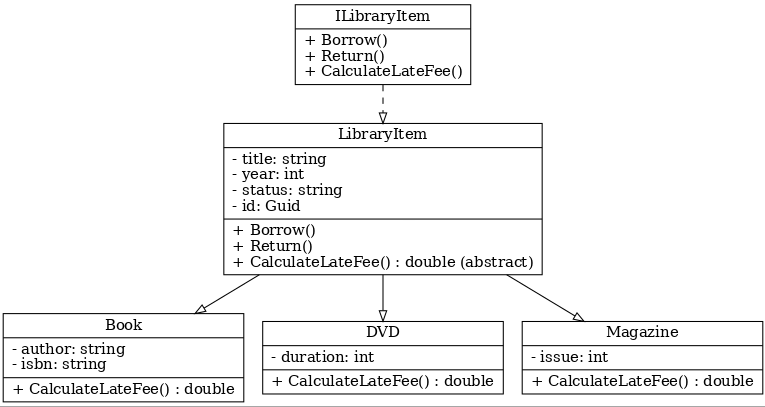
**الخُلاصة:**

**هذا المشروع يطبق المبادئ الأساسية للبرمجة الكائنية بوضوح:**

* **التجريد عبر الواجهة والفئة المجردة.**
* **الوراثة وتعدد الأشكال عبر الفئات Book, DVD, Magazine وإعادة تعريف الدوال.**
* **التغليف عبر الخصائص والحقول الخاصة مع التحقق.**
* **الأحداث والتفويضات لإشعار المستخدم بالتغييرات.**
* **الفئات والأعضاء الساكنة لأغراض التحقق وتتبع عدد العناصر.**

**وبذلك يغطي النظام بشكل عملي ومتكامل أهم مبادئ OOP ويوضح فائدتها المباشرة في بناء برنامج منظم، آمن، وقابل للتوسعة.**

**رسم UML موسع:**



* **يوضح الوراثة، الفئات الفرعية، الواجهة، الخصائص الأساسية والدوال المجردة.**

**شرح التنفيذ العملي (Code Walkthrough):**

1. **الفئة الأساسية: LibraryItem**
   * **تمثل حجر الأساس للنظام، حيث تحتوي على الخصائص المشتركة بين جميع عناصر المكتبة (العنوان، السنة، الحالة، المعرّف الفريد).**
   * **تتضمن دالة مجردة CalculateLateFee(int daysLate) التي يجب على كل فئة فرعية أن تعرّفها بطريقتها الخاصة.**
   * **بهذه الطريقة نحقق مبدأ التجريد (Abstraction).**
2. **الفئات Book / DVD / Magazine:**
   * **كل فئة ترث من LibraryItem وتعيد تعريف الدالة CalculateLateFee() بحيث تناسب طبيعة العنصر:**
     + **: Book رسوم متأخرة منخفضة (0.5 دولار لكل يوم).**
     + **:DVD رسوم أعلى (1.5 دولار لكل يوم) بسبب قيمة الـ DVD.**
     + **: Magazine رسوم متوسطة (0.2 دولار لكل يوم).**
   * **هنا يظهر مبدأ الوراثة وتعدد الأشكال (Inheritance & Polymorphism) بوضوح.**
3. **الفئة: LibraryCatalog**
   * **تحتوي على قائمة (List) من عناصر المكتبة (Library Item).**
   * **باستخدام تعدد الأشكال يمكن إضافة أي عنصر )كتاب، مجلة، (DVDوالتعامل معه عبر نفس القائمة.**
   * **توفر خصائص لحساب العدد الكلي للعناصر، وعدد العناصر المستعارة.**
   * **تعكس عمليًا فكرة إدارة مجموعة عناصر مختلفة بطريقة موحدة.**
4. **التفويض والأحداث: (Delegates & Events)**
   * **تم تعريف Delegate مخصص لإشعار المستخدم عند حدوث حدث مهم**

**(استعارة أو إرجاع).**

* + **عند استعارة أو إرجاع عنصر يتم إطلاق Event يظهر للمستخدم تفاصيل العملية (العنصر + المستعير + التاريخ).**
  + **تمت إضافة معالج مخصص (Custom Handler) لمتابعة الأحداث الخاصة بالكتب كميزة إضافية.**
  + **هذا يضيف للنظام ديناميكية وتفاعلية بدلًا من كونه برنامجًا ثابتًا.**

1. **الفئات والأعضاء الساكنة: (Static Classes & Members)**
   * **تم إنشاء فئة ساكنة ValidationHelper للتحقق من صحة البيانات,**

**(مثال: التأكد من أن العنوان ليس فارغًا).**

* + **في فئة LibraryItem تمت إضافة عداد ساكن (Static Counter) ليتتبع إجمالي عدد العناصر التي تم إنشاؤها.**
  + **هذا يعكس أهمية الأعضاء الساكنة في مشاركة البيانات عبر جميع الكائنات دون الحاجة إلى إنشاء نسخة جديدة من الكائن.**

**أمثلة على التشغيل :(Program Output)**

=== Catalog Items ===

[Book] Title: The Pragmatic Programmer | Year: 1999 | Status: Available | ID: ...

[DVD] Title: Inception | Year: 2010 | Status: Available | ID: ...

[Magazine] Title: National Geographic | Year: 2021 | Status: Available | ID: ...

=====================

Borrowing book1...

[EVENT] Item 'The Pragmatic Programmer' borrowed by Alice

Borrowing dvd1...

[EVENT] Item 'Inception' borrowed by Bob

=== Catalog Items ===

[Book] Title: The Pragmatic Programmer | Status: Borrowed by Alice

[DVD] Title: Inception | Status: Borrowed by Bob

...

Calculating late fees for 3 days late:

- The Pragmatic Programmer (Book): $1.50

- Inception (DVD): $4.50

- National Geographic (Magazine): $0.60

Returning items updates status back to Available

* **يظهر استلاف العناصر، تغيير الحالة، حساب الرسوم، وإرجاعها بشكل صحيح.**

**الخاتمة (Conclusion):**

**لقد تم في هذا المشروع بناء نظام مكتبة متكامل يعتمد بشكل كامل على مبادئ البرمجة غرضية التوجه (OOP)، بحيث لا يقتصر على التطبيق الأكاديمي للمفاهيم النظرية فقط، بل يقدم مثالًا عمليًا قابلًا للتوسع والاستخدام في سيناريوهات مشابهة.**

أهم ما يميز النظام أنه دمج بشكل متوازن بين:

1. **التجريد (Abstraction):** حيث تم عزل العمليات المشتركة في واجهة وفئة مجردة، مما جعل من السهل إضافة أنواع جديدة من العناصر في المستقبل دون الحاجة إلى تعديل البنية الأساسية.
2. **الوراثة وتعدد الأشكال (Inheritance & Polymorphism):** إذ استُخدمت الوراثة لإعادة استخدام الشيفرة المشتركة بين العناصر المختلفة (كتاب، مجلة، قرص DVD)، بينما مكّن تعدد الأشكال من التعامل مع العناصر جميعها بطريقة موحّدة عبر قائمة الكتالوج دون الحاجة إلى معرفة نوع العنصر مسبقًا.
3. **التغليف (Encapsulation):** من خلال التحكم في الوصول إلى البيانات باستخدام الخصائص (Properties) بدلًا من الحقول المباشرة، مما وفر أمانًا أكبر للبيانات ومنع التلاعب غير الصحيح بها.
4. **التفويض والأحداث (Delegates & Events):** أضافت هذه الميزة ديناميكية للنظام، بحيث يمكن مراقبة التغيرات المهمة مثل عمليات الاستعارة والإرجاع، مما يفتح المجال لربط النظام مع واجهات مستخدم رسومية (GUI) أو إشعارات مستقبلية.
5. **الفئات والإجراءات الساكنة (Static Classes & Members):** أتاحت إنشاء أدوات مساعدة عامة (Utility Functions) وعداد موحد للعناصر في الكتالوج، مما وفر مرونة في متابعة حالة النظام ككل.

**من خلال هذه العناصر، أصبح النظام سهل الصيانة، قابلًا للتوسع، وواضح البنية، حيث يمكن لطلاب آخرين أو مطورين مستقبليين إضافة خصائص جديدة مثل:**

* البحث عن عنصر حسب المؤلف أو السنة.
* إدارة حسابات المستعيرين وربطها بمدة الاستعارة.
* تطوير واجهة رسومية رسومية (WPF/WinForms) أو حتى واجهة ويب.
* إضافة قاعدة بيانات لحفظ العناصر بشكل دائم بدلًا من تخزينها في الذاكرة.

**وبذلك، فإن هذا المشروع لا يحقق فقط أهداف المقرر الأكاديمية، بل يشكل أيضًا نواة أولية لتطبيق عملي يمكن تطويره ليصبح نظام مكتبة حقيقي يخدم مؤسسة تعليمية أو ثقافية.**

**رابط الاستضافة على موقع GitHub:**