**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**ĐẠI HỌC KINH TẾ TP HỒ CHÍ MINH (UEH)**

**TRƯỜNG CÔNG NGHỆ VÀ THIẾT KẾ**

🖎🕮✍



**ĐỒ ÁN MÔN HỌC**

**ĐỀ TÀI**

**XÂY DỰNG CHƯƠNG TRÌNH   
QUẢN LÝ XUẤT NHẬP KHO HÀNG HÓA**

**Học Phần: Lập trình hướng đối tượng**

**Danh Sách Nhóm**:

1. NGUYỄN PHAN TUẤN ĐỨC
2. NGUYỄN HOÀNG BẢO
3. HUỲNH KIM NGUYÊN
4. VÕ ĐỨC HƯNG

**Chuyên Ngành**: CÔNG NGHỆ THÔNG TIN

**Khóa**: K49

**Giảng Viên**: TS. Đặng Ngọc Hoàng Thành

**Tp. Hồ Chí Minh, Ngày 8 tháng 11 năm 2021**

# MỤC LỤC

[MỤC LỤC 2](#_Toc181967515)

[CHƯƠNG 1. PHẦN MỞ ĐẦU 3](#_Toc181967516)

[1.1. Tính cấp thiết của đề tài 3](#_Toc181967517)

[1.2. Giới thiệu về Kĩ thuật lập trình hướng đối tượng (LTHĐT) 3](#_Toc181967518)

[1.3. Tầm quan trọng của LTHĐT trong Công nghệ thông tin 4](#_Toc181967519)

[CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ LỚP 5](#_Toc181967520)

[2.1. Phân tích bài toán Quản lý xuất nhập kho hàng hóa 5](#_Toc181967521)

[2.2. Thiết kế lớp và Sơ đồ lớp (Class diagrams) 10](#_Toc181967522)

[2.3. Cài đặt các lớp chức năng (4 tính chất của OOP và các kỹ thuật khác) 11](#_Toc181967523)

[CHƯƠNG 3. XÂY DỰNG ỨNG DỤNG 20](#_Toc181967524)

[3.1. Thiết kế giao diện chương trình 20](#_Toc181967525)

[3.2. Phát triển các chức năng của ứng dụng 23](#_Toc181967526)

[3.3. Các kịch bản thực thi ứng dụng 25](#_Toc181967527)

[CHƯƠNG 4. THẢO LUẬN & ĐÁNH GIÁ 27](#_Toc181967528)

[4.1. Các Kết Quả Nhận Được 27](#_Toc181967529)

[4.2. Một Số Tồn Tại 31](#_Toc181967530)

[4.3. Hướng Phát Triển 32](#_Toc181967531)

[PHỤ LỤC 34](#_Toc181967532)

[TÀI LIỆU THAM KHẢO 36](#_Toc181967533)

# CHƯƠNG 1. PHẦN MỞ ĐẦU

## 1.1. Tính cấp thiết của đề tài

Nhóm nhận thấy nhu cầu thiết kế một chương trình quản lý xuất nhập kho bằng C# WinForm là cần thiết nhằm giảm thiểu sai sót và tối ưu hóa quy trình làm việc. Hiện nay, nhiều doanh nghiệp vẫn quản lý kho hàng theo cách thủ công như ghi chép sổ sách hoặc sử dụng bảng Excel. Phương pháp này dễ dẫn đến sai sót trong quá trình nhập liệu, gây ra các vấn đề nghiêm trọng như thất thoát hàng hóa và khó khăn trong việc theo dõi chính xác lượng tồn kho. Một hệ thống quản lý tự động sẽ giúp khắc phục những hạn chế này, đặc biệt hiệu quả trong các mùa cao điểm khi khối lượng hàng hóa tăng cao.

Chương trình quản lý xuất nhập kho sẽ giúp nhân viên tiết kiệm thời gian và nâng cao hiệu suất làm việc. Thay vì ghi chép thủ công, nhân viên có thể cập nhật dữ liệu nhanh chóng và chính xác qua hệ thống, đảm bảo việc quản lý kho hàng trở nên dễ dàng và hiệu quả hơn.

Ngoài ra, chương trình giúp theo dõi hàng hóa chặt chẽ hơn, điều này đặc biệt quan trọng trong các ngành như thực phẩm, nơi cần giám sát kỹ lưỡng tình trạng hàng hóa để tránh lãng phí. Hệ thống còn cho phép quản lý hàng hóa theo lô, đảm bảo không có mặt hàng nào bị bỏ sót hoặc hư hỏng do không được xử lý kịp thời.

Cuối cùng, chương trình này có khả năng tích hợp cao, dễ dàng đồng bộ dữ liệu kho với các hệ thống khác như hệ thống bán hàng hoặc kế toán, giúp thông tin luôn nhất quán và hỗ trợ ra quyết định dựa trên dữ liệu chính xác. Tất cả những yếu tố này giúp chương trình quản lý xuất nhập kho trở thành một công cụ hỗ trợ đắc lực, góp phần nâng cao hiệu quả và sức cạnh tranh của doanh nghiệp trên thị trường.

## 1.2. Giới thiệu về Kĩ thuật lập trình hướng đối tượng (LTHĐT)

Lập trình hướng đối tượng (Object-Oriented Programming - OOP) là một phương pháp lập trình mô phỏng các đối tượng trong thế giới thực vào mã nguồn, giúp tổ chức mã nguồn khoa học, dễ bảo trì và mở rộng. Hai khái niệm cơ bản trong OOP là đối tượng (object) và lớp (class).

**Đối tượng** là các thực thể trong chương trình, có **thuộc tính** (đặc điểm) và **phương thức** (hành động). Ví dụ, đối tượng máy tính có thuộc tính như màu sắc, dung lượng bộ nhớ và phương thức như khởi động hay tắt máy.

**Lớp** là khuôn mẫu để tạo ra các đối tượng có chung thuộc tính và phương thức, giúp nhóm các đối tượng tương tự lại với nhau, như lớp Máy Vi Tính chứa các thuộc tính chung của các loại máy tính.

Lập trình hướng đối tượng (OOP) giúp tổ chức mã nguồn hiệu quả, mô phỏng thế giới thực và dễ dàng bảo trì, mở rộng. Các tính chất quan trọng trong OOP như **kế thừa, đóng gói, đa hình** và **trừu tượng** giúp giảm thiểu sự phức tạp và hỗ trợ phát triển phần mềm linh hoạt.

***1.2.1. Encapsulation (Tính đóng gói – bao đóng)***

Tính đóng gói trong lập trình hướng đối tượng giúp ẩn các chi tiết thực thi và dữ liệu bên trong đối tượng khỏi sự truy cập trực tiếp từ bên ngoài, đảm bảo bảo vệ tính toàn vẹn của dữ liệu. Người dùng chỉ có thể truy cập và thay đổi dữ liệu thông qua các phương thức do đối tượng cung cấp.

***1.2.2. Abstraction (Tính trừu tượng)***

Tính trừu tượng trong lập trình hướng đối tượng là quá trình che giấu các chi tiết cài đặt không cần thiết và chỉ cung cấp các thông tin quan trọng, cần thiết để người dùng tương tác với đối tượng. Điều này giúp giảm bớt sự phức tạp và chỉ tập trung vào các yếu tố cốt lõi của đối tượng.

***1.2.3. Inheritance (Tính thừa kế)***

Kế thừa là một trong những tính năng cơ bản của lập trình hướng đối tượng, cho phép một lớp con thừa hưởng các thuộc tính và phương thức của lớp cha. Lớp con có thể mở rộng hoặc thay đổi các phương thức, thuộc tính của lớp cha để phù hợp với yêu cầu của mình, giúp giảm thiểu sự lặp lại mã nguồn và tăng tính tái sử dụng.

***1.2.4. Polymorphism (Tính đa hình)***

Đa hình là một khái niệm trong lập trình hướng đối tượng cho phép các phương thức có cùng tên nhưng thực thi các hành động khác nhau tùy thuộc vào đối tượng cụ thể mà chúng được gọi. Điều này giúp tăng tính linh hoạt và mở rộng của hệ thống, đặc biệt khi các đối tượng thuộc các lớp khác nhau nhưng có thể sử dụng chung một phương thức với cách thực thi khác nhau.

Lưu ý: Ngoài ra còn một số kỹ thuật khác trong OOP có thể kể đến như **Delegate, Event, Bounding/Unbounding, v.v, Interface, Generic Collection - List<T>, Exception Handling), XML Serializer**

## 1.3. Tầm quan trọng của LTHĐT trong Công nghệ thông tin

Lập trình hướng đối tượng (OOP) đóng vai trò quan trọng trong việc phát triển phần mềm hiện đại, giúp tạo ra các hệ thống dễ bảo trì, dễ mở rộng và hiệu quả hơn. Một trong những ưu điểm lớn của OOP là khả năng **tái sử dụng mã nguồn**, giảm thiểu sự lặp lại khi phát triển các ứng dụng. Các lớp và đối tượng có thể được tái sử dụng trong các dự án khác nhau, giúp tiết kiệm thời gian và giảm thiểu lỗi.

Bên cạnh đó, OOP cũng mang lại khả năng **mở rộng và bảo trì dễ dàng**. Khi cần thay đổi hoặc bổ sung tính năng mới, lập trình viên chỉ cần điều chỉnh các lớp hoặc đối tượng mà không ảnh hưởng đến toàn bộ hệ thống. Điều này giúp giảm độ phức tạp và làm cho phần mềm trở nên dễ quản lý hơn.

OOP còn nổi bật với tính **bảo mật**, nhờ vào tính đóng gói (encapsulation), giúp bảo vệ dữ liệu khỏi sự truy cập trái phép. Ngoài ra, khả năng **quản lý độ phức tạp** trong các hệ thống lớn, cùng với tính **linh hoạt** trong việc tích hợp với các hệ thống khác, khiến OOP trở thành công cụ lý tưởng cho việc phát triển phần mềm phức tạp. Cuối cùng, OOP thúc đẩy việc **lập trình theo nhóm**, tạo điều kiện cho sự hợp tác hiệu quả giữa các lập trình viên.

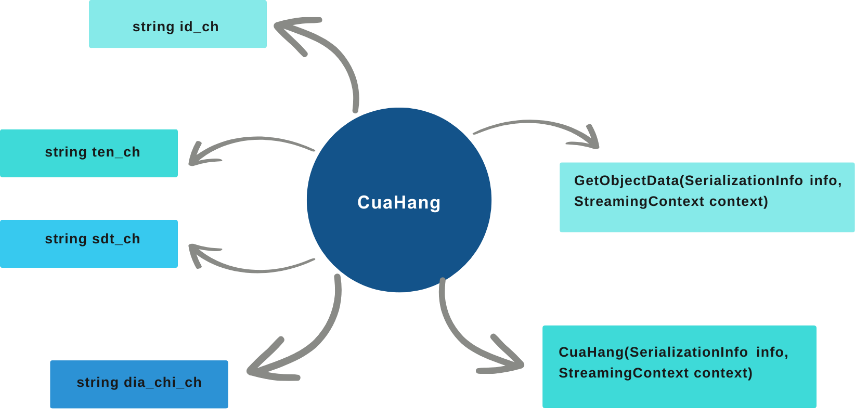
Nhờ vào những đặc điểm trên, OOP đã trở thành nền tảng quan trọng trong công nghệ thông tin, giúp các hệ thống phần mềm có thể đáp ứng được yêu cầu ngày càng cao về tính linh hoạt, bảo trì, và mở rộng.

# CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ LỚP

## 2.1. Phân tích bài toán Quản lý xuất nhập kho hàng hóa

**Các đối tượng có những thuộc tính cũng như phương thức phù hợp với bài toán**

**1. CuaHang (Cửa Hàng)**



Hình 1. Đối tượng CuaHang (các thuộc tính và phương thức)

Phân tích: Cửa hàng là đối tượng có liên quan tới việc lưu trữ và cung cấp hàng hóa. Việc lưu trữ các thông tin cửa hàng giúp dễ dàng theo dõi các mặt hàng nhập xuất từ từng cửa hàng cụ thể.

**2. NhaCungCap (Nhà Cung Cấp)**

A diagram of a blue circle with arrows

Description automatically generated

Hình 2. Đối tượng NhaCungCap (các thuộc tính và phương thức)

Phân tích: Nhà cung cấp là nguồn cung cấp hàng hóa cho kho hàng và các cửa hàng. Thông tin nhà cung cấp sẽ được sử dụng để theo dõi các giao dịch cung cấp hàng hóa.

**3. NhanVien (Nhân Viên)**

A diagram of a diagram

Description automatically generated

Hình 3. Đối tượng NhanVien (các thuộc tính và phương thức)

Phân tích: Nhân viên là những người thực hiện các công việc liên quan đến kho hàng, như nhập xuất hàng hóa, kiểm tra số lượng hàng tồn kho, tạo hóa đơn, và các hoạt động quản lý kho khác.

**4. HangHoa (Hàng Hóa)**

A diagram of a diagram

Description automatically generated

Hình 4. Đối tượng HangHoa (các thuộc tính và phương thức)

Phân tích: Hàng hóa là yếu tố quan trọng trong bài toán xuất nhập kho. Các thao tác nhập kho, xuất kho đều liên quan đến hàng hóa, vì vậy cần quản lý số lượng và giá trị của từng mặt hàng trong kho.

**5. QuanLyNhapXuat (Quản Lý Nhập Xuất)**

A diagram of a blue circle with white text

Description automatically generated

Hình 5. Đối tượng QuanLyNhapXuat (các thuộc tính và phương thức)

Phân tích: Đây là đối tượng chịu trách nhiệm quản lý hoạt động nhập và xuất hàng hóa. Các phương thức này đảm bảo rằng hàng hóa được thêm, xóa và cập nhật đúng cách trong quá trình nhập xuất.

**6. HoaDon (Hóa Đơn)**

A diagram of a diagram

Description automatically generated with medium confidence

Hình 6. Đối tượng HoaDon (các thuộc tính và phương thức)

Phân tích: Hóa đơn là kết quả của quá trình giao dịch trong hệ thống kho hàng. Nó ghi lại thông tin về các mặt hàng đã được mua hoặc bán, cùng với thông tin về nhân viên lập hóa đơn và tổng giá trị của giao dịch.

**7. KhoHang (Kho Hàng)**

*A diagram of a diagram

Description automatically generated with medium confidence*

Hình 7. Đối tượng KhoHang (các thuộc tính và phương thức)

Phân tích: Kho hàng là trung tâm lưu trữ và quản lý hàng hóa. Kho cần quản lý tất cả các hóa đơn nhập, xuất, cũng như số lượng sản phẩm hiện có, cùng với các nhà cung cấp và cửa hàng liên kết.

**2 đối tượng kế thừa từ HoaDon**

|  |  |  |
| --- | --- | --- |
| Lớp | Thuộc tính | Phương thức |
| HoaDonXuat | CuaHang cua\_hang | * HoaDonXuat() * HoaDonXuat(string ma\_don\_hang, QuanLyNhapXuat ds\_san\_pham, NhanVien nv\_lap, CuaHang cua\_hang * virtual void GetObjectData(SerializationInfo info, StreamingContext context) * HoaDonXuat(SerializationInfo info, StreamingContext context) |
| HoaDonNhap | NhaCungCap nha\_cung\_cap | * HoaDonNhap() * HoaDonNhap(string ma\_don\_hang, QuanLyNhapXuat ds\_san\_pham, NhanVien nv\_lap, NhaCungCap nha\_cung\_cap) * virtual void GetObjectData(SerializationInfo info, StreamingContext context * HoaDonNhap(SerializationInfo info, StreamingContext context) |

**3 đối tượng kế thừa từ HangHoa**

|  |  |  |
| --- | --- | --- |
| Lớp | Thuộc tính (Attributes) | Phương thức (Methods) |
| DoGiaDung | - String id  - String ten\_hang  - Ulnt so\_luong  - Long don\_gia | - override void GetObjectData(SerializationInfo info, StreamingContext context)  - DoGiaDung(SerializationInfo info, StreamingContext context) |
| DienTu | - String id  - String ten\_hang  - Ulnt so\_luong  - Long don\_gia | - override void GetObjectData(SerializationInfo info, StreamingContext context)  - DienTu(SerializationInfo info, StreamingContext context) |
| ThucPham | - String id  - String ten\_hang  - Ulnt so\_luong  - Long don\_gia | - override void GetObjectData(SerializationInfo info, StreamingContext context)  - ThucPham(SerializationInfo info, StreamingContext context) |

**Phân tích các mối quan hệ giữa các đối tượng:**

* **KhoHang** và **HangHoa** có mối quan hệ chặt chẽ vì kho hàng chứa các sản phẩm, và mỗi sản phẩm có thể xuất hiện trong nhiều hóa đơn (nhập và xuất).
* **HoaDon** kết nối với **NhanVien** và **HangHoa**, cho thấy rằng mỗi hóa đơn được lập bởi một nhân viên và liên quan đến các sản phẩm cụ thể.
* **NhaCungCap** cung cấp hàng hóa cho **KhoHang**, giúp duy trì nguồn cung cấp hàng hóa trong kho.
* **CuaHang** có thể sử dụng kho hàng để cung cấp sản phẩm cho khách hàng, và **KhoHang** có thể có nhiều **CuaHang** liên kết với nhau.
* **QuanLyNhapXuat** sẽ xử lý các thao tác như thêm, xóa, cập nhật sản phẩm trong kho khi có sự thay đổi về số lượng.

## 2.2. Thiết kế lớp và Sơ đồ lớp (Class diagrams)

A computer screen shot of a computer

Description automatically generated

Hình 8. Class Diagram liên kết nhau của toàn bộ class trong chương trình

1. **HangHoa (Abstract Class)** có mối quan hệ **kế thừa** với các lớp:

* GiaDung
* ThucPham
* DienTu

Các lớp này đều là các loại hàng hóa cụ thể và kế thừa từ lớp **trừu tượng** **HangHoa**.

1. **KhoHang** có mối quan hệ liên kết với:

* HangHoa: Kho hàng chứa danh sách các hàng hóa (ds\_hang\_hoa).
* HoaDonNhap và HoaDonXuat: Kho hàng có danh sách hóa đơn nhập (ds\_hoa\_don\_nhap) và hóa đơn xuất (ds\_hoa\_don\_xuat).
* NhanVien: Kho hàng có danh sách các nhân viên (ds\_nhan\_vien).
* NhaCungCap: Kho hàng có danh sách các nhà cung cấp (ds\_ncc).
* CuaHang: Kho hàng có danh sách các cửa hàng (ds\_cua\_hang).

1. **QuanLyNhapXuat** có mối quan hệ liên kết với:

* HangHoa: Quản lý nhập xuất có danh sách hàng hóa (ds\_hang\_hoa) để quản lý số lượng và trong lô hàng của hàng hóa.

1. **HoaDon (Abstract Class)** có mối quan hệ kế thừa với:

* HoaDonNhap: Là một loại hóa đơn dành cho việc nhập hàng.
* HoaDonXuat: Là một loại hóa đơn dành cho việc xuất hàng.

1. **HoaDon** có mối quan hệ liên kết với:

* NhanVien: Hoá đơn có thông tin của nhân viên lập hoá đơn

1. **HoaDonNhap** có mối quan hệ liên kết với:

* NhaCungCap: Hóa đơn nhập hàng có thông tin về nhà cung cấp (nha\_cung\_cap).

1. **HoaDonXuat** có mối quan hệ liên kết với:

* CuaHang: Hóa đơn xuất hàng có thông tin về cửa hàng (cua\_hang).

## 2.3. Cài đặt các lớp chức năng (4 tính chất của OOP và các kỹ thuật khác)

Nhóm em tích hợp các chức năng có sẵn trong các class đồng thời khai báo các thuộc tính cần thiết trong mỗi đối tượng cần thiết

### 2.3.1. Cấu trúc code các lớp được cài đặt và thiết kế liên kết với nhau

**1. Cấu trúc các lớp**

* **CuaHang**:
  + Lớp này đại diện cho cửa hàng với các thuộc tính như **id\_ch**, **ten\_ch**, **sdt\_ch**, và **dia\_chi\_ch**.
  + Cài đặt **ISerializable** để cho phép tuần tự hóa đối tượng.
* **HangHoa** (Lớp trừu tượng):
  + Là lớp cha cho các loại hàng hóa khác (DienTu, GiaDung, ThucPham).
  + Có các thuộc tính chung cho hàng hóa như **id**, **ten\_hang**, **so\_luong**, **don\_gia**, và **img**.
  + Cài đặt **ICloneable** cho phép sao chép đối tượng và **ISerializable** để tuần tự hóa.
  + Phương thức **Clone** cho phép tạo bản sao của hàng hóa.
* **DienTu**, **GiaDung** và **ThucPham**:
  + Là các lớp con kế thừa từ **HangHoa**, đại diện cho các loại hàng hóa cụ thể.
* **HoaDon** (Lớp trừu tượng):
  + Là lớp cha cho các hóa đơn (HoaDonNhap, HoaDonXuat).
  + Chứa các thuộc tính như **id\_hoa\_don**, **ngay\_tao\_don**, **qlnx**, **nv\_lap**, và **tong\_tien**.
  + Cài đặt **ISerializable** để tuần tự hóa.
* **HoaDonNhap** và **HoaDonXuat**:
  + Là các lớp con kế thừa từ **HoaDon**, đại diện cho hóa đơn nhập và hóa đơn xuất.
  + Mỗi lớp có thêm thuộc tính riêng (nha\_cung\_cap cho HoaDonNhap và cua\_hang cho HoaDonXuat).
* **KhoHang**:
  + Quản lý danh sách hàng hóa, nhân viên, cửa hàng, nhà cung cấp, và hóa đơn.
  + Cung cấp các phương thức để cập nhật kho, thêm/xóa hàng hóa, lưu trữ và tải dữ liệu.
* **NhaCungCap**:
  + Đại diện cho nhà cung cấp với các thuộc tính như **id\_ncc**, **ten\_ncc**, **sdt\_ncc**, và **dia\_chi\_ncc**.
* **NhanVien**:
  + Đại diện cho nhân viên với các thuộc tính như **id\_nv**, **ten\_nv**, **tuoi**, **gioi\_tinh**, **dia\_chi\_nv**, **username**, và **password**.
* **QuanLyNhapXuat**:
  + Quản lý danh sách hàng hóa, cho phép thêm/xóa hàng hóa, tăng/giảm số lượng và kiểm tra sự tồn tại của hàng hóa.

**2. Liên kết giữa các lớp**

* **KhoHang** sử dụng các lớp khác để quản lý dữ liệu:
  + Danh sách hàng hóa (**ds\_hang\_hoa**) là danh sách của các đối tượng **HangHoa**.
  + Danh sách hóa đơn nhập (**ds\_hoa\_don\_nhap**) và hóa đơn xuất (**ds\_hoa\_don\_xuat**) là danh sách của các đối tượng **HoaDonNhap** và **HoaDonXuat**.
  + Danh sách cửa hàng (**ds\_cua\_hang**) là danh sách của các đối tượng **CuaHang**.
  + Danh sách nhà cung cấp (**ds\_ncc**) là danh sách của các đối tượng **NhaCungCap**.
  + Danh sách nhân viên (**ds\_nhan\_vien**) là danh sách của các đối tượng **NhanVien**.
* **HoaDonNhap** và **HoaDonXuat** chứa các thuộc tính tham chiếu đến **NhaCungCap** và **CuaHang**, cho phép xác định nguồn gốc và đích đến của hàng hóa trong giao dịch.
* **QuanLyNhapXuat** là lớp quản lý hàng hóa, có thể được sử dụng trong **HoaDon** để tính toán tổng tiền cho hóa đơn dựa trên danh sách hàng hóa.

**3. Phương thức chính**

* Quản lý hàng hóa: Thêm, xóa, và cập nhật thông tin hàng hóa trong kho.

### Quản lý hóa đơn: Tạo hóa đơn nhập và xuất, lưu trữ và truy xuất thông tin hóa đơn.

### Quản lý nhân viên và nhà cung cấp: Lưu trữ thông tin về nhân viên và nhà cung cấp, hỗ trợ đăng nhập và xác thực.

### Lưu trữ dữ liệu: Sử dụng XML để lưu trữ và tải dữ liệu cho các danh sách hàng hóa, hóa đơn, nhân viên và nhà cung cấp.

### 2.3.2. Các kỹ thuật OOP áp dụng cho bài toán

#### 2.3.2.1. Bốn tính chất của OOP

**1. Encapsulation**

Hầu hết các lớp đều sử dụng các trường dữ liệu và phương thức để đóng gói thông tin của đối tượng trong chương trình này. Ví dụ, HangHoa có các thuộc tính id, ten\_hang, so\_luong, don\_gia, img và đặc biệt là phương thức protected HangHoa() giúp kiểm soát truy cập và thao tác với dữ liệu bên trong đối tượng.

public abstract class HangHoa : *ISerializable*, *ICloneable*

{

    public string id;

    public string ten\_hang;

    public uint so\_luong;

    public ulong don\_gia;

    public string img;

    protected HangHoa() { }

}

Trong lớp HangHoa, tính đóng gói được thể hiện thông qua phương thức dựng protected:

* protected HangHoa() { } là một phương thức dựng được khai báo với mức truy cập protected, nghĩa là chỉ có các lớp dẫn xuất của HangHoa mới có thể gọi đến nó.
* Điều này ngăn cản việc tạo trực tiếp một đối tượng từ lớp HangHoa bên ngoài, giúp kiểm soát cách mà các thể hiện HangHoa được khởi tạo.

VD khác về class DienTu được kế thừa từ HangHoa và sử dung private để đóng gói thông tin

public class DienTu : *HangHoa*

{

    private DienTu() { }

}

Phương thức dựng private trong DienTu:

* private DienTu() { } là phương thức dựng riêng tư, ngăn không cho khởi tạo đối tượng DienTu mà không đi qua phương thức công khai khác, như public DienTu(...).
* Điều này cho phép kiểm soát cách các đối tượng DienTu được khởi tạo và bảo vệ chúng khỏi các khởi tạo không mong muốn.

**2. Abstraction**

Lớp HangHoa và HoaDon là các lớp trừu tượng, khai báo những đặc điểm cơ bản của sản phẩm và hóa đơn, giúp nhóm các thuộc tính và phương thức chung cho các loại hàng hóa và hóa đơn.

Trừu tượng hóa được thể hiện qua lớp trừu tượng HangHoa, trong đó khai báo các thuộc tính và phương thức chung cho tất cả các loại hàng hóa. Các lớp con như DienTu và GiaDung sẽ kế thừa lớp HangHoa và triển khai chi tiết:

public abstract class HangHoa : *ICloneable*, *ISerializable*

Lớp HoaDon là một lớp trừu tượng, đại diện cho các thuộc tính và hành vi chung của một hóa đơn. Các lớp con như HoaDonNhap và HoaDonXuat kế thừa từ HoaDon và triển khai chi tiết cho các loại hóa đơn khác nhau (hóa đơn nhập, hóa đơn xuất):

public abstract class HoaDon : *ISerializable*

**3. Inheritance**

Kế thừa được sử dụng để tái sử dụng mã nguồn và tạo ra các lớp con với các đặc tính của lớp cha. Ví dụ, DienTu, GiaDung, và ThucPham kế thừa từ lớp HangHoa, trong khi HoaDonNhap và HoaDonXuat kế thừa từ HoaDon.

Kế thừa được thể hiện qua việc lớp DienTu, GiaDung và ThucPham kế thừa từ lớp HangHoa. Điều này giúp chúng chia sẻ các thuộc tính và phương thức từ lớp cha HangHoa mà không cần phải định nghĩa lại:

public class DienTu : HangHoa

public class GiaDung : HangHoa

public class ThucPham : HangHoa

Lớp HoaDonNhap và HoaDonXuat kế thừa từ HoaDon, sử dụng lại các thuộc tính và phương thức chung từ lớp cha, và có thể mở rộng hoặc ghi đè các phương thức theo nhu cầu cụ thể:

public class HoaDonNhap : HoaDon

public class HoaDonXuat : HoaDon

**4. Polymorphism**

Tính đa hình được thể hiện trong chương trình bằng cách ghi đè phương thức GetObjectData trong các lớp con như DienTu, GiaDung, ThucPham, HoaDonNhap, và HoaDonXuat, cho phép mỗi lớp con cung cấp một phiên bản riêng của phương thức.

Ví dụ về lớp HangHoa là lớp cha của lớp ThucPham và được ghi đè dữ liệu:

public abstract class HangHoa : *ICloneable*, *ISerializable*

{

    public virtual void GetObjectData(*SerializationInfo* *info*, *StreamingContext* *context*)

    {

    }

}

public class ThucPham : *HangHoa*

{

    public override void GetObjectData(*SerializationInfo* *info*, *StreamingContext* *context*)

    {

*base*.GetObjectData(*info*, *context*); // Gọi phương thức từ lớp cha

    }

}

Lớp HangHoa định nghĩa phương thức GetObjectData với từ khóa virtual, cho phép các lớp con ghi đè phương thức này nếu cần. Điều này có nghĩa là HangHoa cung cấp một phiên bản cơ bản của GetObjectData, nhưng các lớp con như ThucPham có thể thay đổi cách triển khai của phương thức.

ThucPham kế thừa từ HangHoa và ghi đè GetObjectData với từ khóa override. Nhờ tính đa hình, khi GetObjectData được gọi trên một đối tượng ThucPham, phiên bản trong ThucPham sẽ được thực thi thay vì phiên bản trong HangHoa, ngay cả khi đối tượng được tham chiếu bằng kiểu HangHoa.

Tính đa hình cho phép một phương thức có thể thực thi theo cách khác nhau, dựa vào loại đối tượng thực tế đang sử dụng nó. Trong đoạn mã này:

* Khi GetObjectData được gọi trên một đối tượng kiểu ThucPham, hệ thống sẽ sử dụng phiên bản của GetObjectData trong ThucPham, không phải phiên bản trong HangHoa, nhờ vào việc ThucPham đã ghi đè phương thức này.
* Trong GetObjectData của ThucPham, dòng base.GetObjectData(info, context); gọi lại phiên bản của GetObjectData từ lớp cha (HangHoa). Điều này cho phép ThucPham tận dụng mã của HangHoa để thực hiện các thao tác cơ bản, đồng thời có thể thêm logic đặc thù cho ThucPham nếu cần.

Hoặc một ví dụ khác đó là lớp HoaDon khai báo SetID là một phương thức abstract, bắt buộc các lớp con phải triển khai. Điều này có nghĩa HoaDon không có logic cụ thể cho SetID, mà để các lớp con như HoaDonNhap tự định nghĩa.

public abstract class HoaDon : *ISerializable*

{

    public abstract string SetID();

}

public class HoaDonNhap : *HoaDon*

{

    public override string SetID()

    {

        return "id\_hoa\_don" //Định nghĩa phương thức abstract

    }

}

Lớp HoaDonNhap kế thừa HoaDon và ghi đè SetID bằng từ khóa override. Nhờ tính đa hình, khi SetID được gọi trên đối tượng kiểu HoaDonNhap, phiên bản của HoaDonNhap sẽ được thực thi, ngay cả khi đối tượng được tham chiếu dưới dạng HoaDon. Điều này cho phép HoaDonNhap thực hiện các thao tác riêng mà không phụ thuộc vào lớp cha.

#### 2.3.2.2. Các kỹ thuật khác của OOP

**1. Interface ISerializable**

Trong chương trình:

Các lớp CuaHang, NhanVien, HangHoa, HoaDon, cùng các lớp con như HoaDonNhap, HoaDonXuat, DienTu, GiaDung, và ThucPham đều triển khai ISerializable.

Khi một lớp triển khai ISerializable, nó phải định nghĩa phương thức GetObjectData, để xác định cách các thuộc tính của lớp sẽ được serial hóa.

Ví dụ về lớp CuaHang triển khai ISerializable:

public class CuaHang : *ISerializable*

{

    public string id\_ch;

    public string ten\_ch;

    public string sdt\_ch;

    public string dia\_chi\_ch;

    public void GetObjectData(*SerializationInfo* *info*, *StreamingContext* *context*)

    {

*info*.AddValue("id\_ch", id\_ch);

*info*.AddValue("ten\_ch", ten\_ch);

*info*.AddValue("sdt\_ch", sdt\_ch);

*info*.AddValue("dia\_chi\_ch", dia\_chi\_ch);

    }

    public CuaHang(*SerializationInfo* *info*, *StreamingContext* *context*)

    {

        id\_ch = *info*.GetString("id\_ch");

        ten\_ch = *info*.GetString("ten\_ch");

        sdt\_ch = *info*.GetString("sdt\_ch");

        dia\_chi\_ch = *info*.GetString("dia\_chi\_ch");

    }

}

Trong phương thức GetObjectData của CuaHang, các thuộc tính id\_ch, ten\_ch, sdt\_ch, dia\_chi\_ch được thêm vào SerializationInfo để lưu trữ. Khi đối tượng CuaHang được serial hóa, các giá trị này sẽ được lưu và có thể được tái tạo lại thông qua constructor CuaHang(SerializationInfo info, StreamingContext context).

Các lớp khác như HoaDon, NhanVien, HangHoa và các lớp con cũng triển khai phương thức GetObjectData tương tự, nhưng với các thuộc tính và dữ liệu đặc trưng của từng lớp.

**2. Interface ICloneable**

Ví dụ về lớp HangHoa triển khai ICloneable:

public abstract class HangHoa : *ICloneable*, *ISerializable*

{

public string id;

public string ten\_hang;

public uint so\_luong;

public ulong don\_gia;

public string img;

public object Clone()

{

HangHoa clone = (HangHoa)*this*.MemberwiseClone();

clone.id = *this*.id;

clone.ten\_hang = *this*.ten\_hang;

clone.don\_gia = *this*.don\_gia;

return clone;

}

}

Trong phương thức Clone, HangHoa sử dụng MemberwiseClone() để tạo bản sao của đối tượng. MemberwiseClone() tạo một bản sao nông (shallow copy) của đối tượng, nghĩa là sao chép giá trị của các thuộc tính cơ bản. Sau đó, Clone gán giá trị cụ thể của các thuộc tính id, ten\_hang, và don\_gia cho đối tượng clone.

Điều này cho phép các lớp con của HangHoa, chẳng hạn như DienTu, GiaDung, và ThucPham, thừa hưởng khả năng sao chép đối tượng từ lớp cha mà không cần triển khai lại phương thức Clone.

**3. Event trong các lớp Form**

Sự kiện (event) là một cơ chế giúp kết nối các hành động của người dùng với các phương thức xử lý logic. Sự kiện có thể được kích hoạt bởi hành động của người dùng, chẳng hạn như nhấp chuột, thay đổi giá trị, hoặc thậm chí các sự kiện hệ thống như tải lại giao diện. Mỗi sự kiện trong tất cả Form của chương trình đều được gắn với một phương thức xử lý sự kiện (event handler) để thực thi một hành động nhất định khi sự kiện đó xảy ra.

Ví dụ: Phương thức tatca\_btn\_Click là một phương thức xử lý sự kiện (event handler) trong lập trình hướng đối tượng (OOP), được gắn với sự kiện Click của nút tatca\_btn.

public partial class FormTrangChu : *System*.*Windows*.*Forms*.*Form*

{

    private void tatca\_btn\_Click(object *sender*, *EventArgs* *e*)

    {

        tatca\_btn.Checked = true;

        giadung\_btn.Checked = false;

        dientu\_btn.Checked = false;

        thucpham\_btn.Checked = false;

        dshh\_flp.Controls.Clear();

        Reload\_flp();

    }

}

Khi người dùng nhấp vào nút, phương thức này được gọi để cập nhật trạng thái các nút và làm mới giao diện:

* Đặt Checked của tatca\_btn là true, và các nút khác là false.
* Xóa các điều khiển trong dshh\_flp.
* Gọi Reload\_flp() để tải lại dữ liệu, hiển thị tất cả danh mục.
* Nhờ event handler, OOP giúp phân tách logic xử lý sự kiện khỏi giao diện, dễ mở rộng và bảo trì mã.

**4. Sử dụng generic collection List<T>**

Trong chương trình, em đã sử dụng List<T> để lưu trữ các đối tượng của các lớp khác nhau, như HangHoa, CuaHang, NhanVien, và các lớp hóa đơn. List<T> là một generic collection trong C#, cho phép bạn lưu trữ một danh sách các đối tượng mà không cần phải xác định kiểu cụ thể của chúng tại thời điểm biên dịch.

Ví dụ vế lớp KhoHang triển khai việc sử dụng generic collection List<T>

    public *List*<*HangHoa*> ds\_hang\_hoa = new *List*<*HangHoa*>();

    public *List*<*NhanVien*> ds\_nhan\_vien = new *List*<*NhanVien*>();

    public *List*<*CuaHang*> ds\_cua\_hang = new *List*<*CuaHang*>();

    public *List*<*NhaCungCap*> ds\_ncc = new *List*<*NhaCungCap*>();

    public *List*<*HoaDonNhap*> ds\_hoa\_don\_nhap = new *List*<*HoaDonNhap*>();

    public *List*<*HoaDonXuat*> ds\_hoa\_don\_xuat = new *List*<*HoaDonXuat*>();

**5. Có sử dụng exception handling (try catch)**

Chương trình sử dụng try-catch để xử lý các ngoại lệ, giúp ngăn chặn chương trình bị lỗi khi gặp các tình huống không mong muốn và cải thiện độ bền vững. try-catch đảm bảo rằng khi xảy ra ngoại lệ, chương trình sẽ không dừng đột ngột mà sẽ có thể xử lý hoặc ghi lại lỗi.

Ví dụ, trong phương thức thêm hàng hóa vào cửa hàng, có thể sử dụng try-catch để bắt ngoại lệ khi có lỗi xảy ra trong quá trình thao tác với dữ liệu:

public void ThemHangHoa(*HangHoa* *hangHoa*)

{

    try

    {

        danhSachHangHoa.Add(*hangHoa*);

    }

    catch (*Exception* ex)

    {

        Console.WriteLine("Lỗi khi thêm hàng hóa: " + ex.Message); // In thông báo lỗi ra màn hình

    }

}

**6. Sử dụng XML Serializer để lưu trữ các dữ liệu cần thiết**

Chương trình sử dụng XML Serializer để lưu trữ dữ liệu dưới dạng XML, giúp đảm bảo tính dễ đọc và dễ dàng truyền tải của dữ liệu. XML Serializer cho phép chuyển đổi đối tượng thành định dạng XML để lưu trữ, sau đó có thể đọc lại dữ liệu từ XML và tái tạo đối tượng ban đầu.

Ví dụ, dưới đây là cách sử dụng XML Serializer để lưu trữ và tải lại dữ liệu của CuaHang:

[*Serializable*]

public class CuaHang : *ISerializable*

{

    public void GetObjectData(*SerializationInfo* *info*, *StreamingContext* *context*)

    {

    }

    public CuaHang(*SerializationInfo* *info*, *StreamingContext* *context*)

    {

    }

}

Trong đoạn mã, XML Serializer được sử dụng để lưu trữ dữ liệu của các lớp như CuaHang vào file XML, giúp dễ dàng quản lý dữ liệu lâu dài và tái sử dụng.

Trong ví dụ này:

* Phương thức GetObjectData ghi các thuộc tính của CuaHang vào SerializationInfo, để lưu trữ chúng trong XML.
* Constructor CuaHang(SerializationInfo info, StreamingContext context) được sử dụng để khôi phục dữ liệu từ file XML, tái tạo lại đối tượng CuaHang với các giá trị trước đó.

Ví dụ khác dưới đây là cách sử dụng XmlSerializer trong lớp KhoHang để tải dữ liệu:

public class KhoHang

{

    public void LoadData()

    {

        string filePath\_ch = "Resources/cua\_hang.dat";

        using (*StreamReader* reader = new *StreamReader*(filePath\_ch))

        {

*XmlSerializer* serializer = new *XmlSerializer*(typeof(*List*<*CuaHang*>));

            ds\_cua\_hang = (*List*<*CuaHang*>)serializer.Deserialize(reader);

        }

    }

}

Phương thức LoadData():

* Đọc dữ liệu từ tệp XML: Sử dụng StreamReader để mở và đọc nội dung tệp XML (cua\_hang.dat).
* XmlSerializer: Tạo một đối tượng XmlSerializer với kiểu dữ liệu là List<CuaHang>. Điều này có nghĩa là XmlSerializer sẽ giải mã tệp XML thành một danh sách các đối tượng CuaHang.
* Deserialize(reader): Phương thức này giải mã dữ liệu XML từ reader (tệp đọc được) và chuyển đổi nó thành danh sách các đối tượng CuaHang. Dữ liệu này được gán cho biến ds\_cua\_hang (danh sách các cửa hàng).

# CHƯƠNG 3. XÂY DỰNG ỨNG DỤNG

## 3.1. Thiết kế giao diện chương trình

Chương trình sẽ có tổng 9 Form cùng với 5 Component sẽ được thực thi luân phiên nhau tương ứng với mỗi chức năng được gọi đến gồm:

* Giao diện đăng nhập hệ thống: FormDangNhap
* Giao diện nền: Form
* Giao diện trang chủ: FormTrangChu
  + HangHoaTrangChuComponent
* Giao diện thêm hàng: FormHangHoa
* Giao diện quản lý nhà cung cấp: FormNhaCungCap
* Giao diện quản lý cửa hàng phân phối: FormCuaHang
* Giao diện phiếu hóa đơn: FormPhieuHoaDon
  + HangHoaBillComponent
  + BillTailComponent
* Giao diện nhập xuất hàng hóa: FormNhapXuat
  + HangHoaNhapXuatComponent
  + HangHoaLoComponent
* Giao diện kiểm đơn hàng hóa: FormHoaDon

**1. Giao diện đăng nhập hệ thống: FormDangNhap**

* Chức năng chính: Cho phép người dùng đăng nhập vào hệ thống bằng tên tài khoản và mật khẩu.
* Các trường thông tin:
  + Tên đăng nhập: Trường để nhập tên đăng nhập.
  + Mật khẩu: Trường để nhập mật khẩu
* Các nút:
  + Đăng nhập: Xác thực thông tin đăng nhập. Nếu thông tin đúng, chuyển đến giao diện trang chủ. Nếu sai, hiển thị thông báo lỗi.
  + Thoát: Đóng giao diện đăng nhập và thoát khỏi ứng dụng.
* Tính năng bổ sung:
  + Kiểm tra tên đăng nhập và mật khẩu để đảm bảo chỉ những người có quyền mới có thể truy cập vào hệ thống. (cụ thể ở đây là nhân viên của công ty)

**2. Giao diện nền: Form**

* Chức năng chính: Đóng vai trò làm nền tảng cho các giao diện khác trong hệ thống.
* Cấu trúc:
  + Thanh menu chính: Chứa các tùy chọn để điều hướng đến các giao diện khác như trang chủ, quản lý hàng hóa, quản lý nhà cung cấp, kiểm đơn hàng,...
* Các nút điều hướng:
  + Trang chủ: Dẫn đến giao diện trang chủ.
  + Nhập xuất hàng hóa: Dẫn đến giao diện nhập xuất hàng.
  + Quản lý nhà cung cấp: Dẫn đến giao diện quản lý nhà cung cấp.
  + Quản lý cửa hàng: Dẫn đến giao diện quản lý cửa hàng.
  + Kiểm đơn hàng hóa: Dẫn đến giao diện kiểm tra đơn hàng.

**3. Giao diện trang chủ: FormTrangChu**

* Chức năng chính: Cung cấp cái nhìn tổng quan về hệ thống cho người dùng ngay sau khi đăng nhập.
* Nội dung hiển thị:
  + Thông tin tóm tắt về thông tin hàng hóa là chủ yếu, kèm theo thanh điều hướng
  + Có nút thêm hàng hóa khi cần nhập hàng hóa mới về kho
* Các nút điều hướng nhanh: Các nút dẫn đến các chức năng chính như thêm hàng hóa, quản lý nhà cung cấp, nhập xuất hàng hóa,...

**4. Giao diện thêm hàng hóa và thông tin hàng hoá: FormHangHoa**

* Chức năng chính: Cho phép thêm mới, xem thông tin hoặc xóa hàng hóa trong kho.
* Các trường thông tin:
  + Mã hàng: Mã định danh hàng hóa.
  + Tên hàng hóa: Tên của sản phẩm.
  + Số lượng: Số lượng tồn trong kho hoặc mặc định bằng 0 nếu nhập thông tin hàng mới
  + Đơn giá: Đơn giá của sản phẩm.
  + Loại hàng hóa: Thực phẩm, gia dụng, điện tử, v.v.
* Các nút:
  + Thêm mới: Lưu thông tin hàng hóa mới vào kho khi ở chế độ thêm hàng.
  + Xoá: Xoá hàng hoá trong kho khi ở chế độ xem thông tin.

**5. Giao diện quản lý nhà cung cấp: FormNhaCungCap**

* Chức năng chính: Quản lý thông tin các nhà cung cấp cung ứng hàng hóa cho kho.
* Các trường thông tin:
  + Mã nhà cung cấp: Mã định danh cho nhà cung cấp.
  + Tên nhà cung cấp: Tên của nhà cung cấp.
  + Địa chỉ: Địa chỉ của nhà cung cấp.
  + Thông tin liên lạc: Số điện thoại,
* Các nút:
  + Thêm mới: Thêm một nhà cung cấp mới vào hệ thống.
  + Cập nhật: Cập nhật thông tin nhà cung cấp hiện có.
  + Xóa: Xóa nhà cung cấp khỏi hệ thống.

**6. Giao diện quản lý cửa hàng phân phối: FormCuaHang**

* Chức năng chính: Quản lý thông tin về các cửa hàng phân phối hàng hóa.
* Các trường thông tin:
  + Mã cửa hàng: Mã định danh cho cửa hàng.
  + Tên cửa hàng: Tên của cửa hàng.
  + Địa chỉ: Địa chỉ của cửa hàng.
  + Số điện thoại của cửa hàng
* Các nút:
  + Thêm mới: Thêm cửa hàng mới vào hệ thống.
  + Cập nhật: Cập nhật thông tin của cửa hàng đã có.
  + Xóa: Xóa cửa hàng khỏi hệ thống.

**7. Giao diện phiếu hóa đơn: FormPhieuHoaDon**

* Chức năng chính: Tạo và quản lý hóa đơn cho các giao dịch nhập xuất hàng hóa.
* Các trường thông tin:
  + Mã hóa đơn: Mã định danh của hóa đơn.
  + Ngày tạo hóa đơn: Ngày phát sinh hóa đơn.
  + Danh sách sản phẩm: Các sản phẩm có trong hóa đơn.
  + Nhân viên lập hóa đơn: Thông tin nhân viên tạo hóa đơn.
  + Nhà cung cấp hàng
  + Cửa hàng nhận hàng
  + Tổng tiền: Tổng giá trị của hóa đơn.
* Các nút:
  + Lưu hóa đơn: Lưu hóa đơn mới vào hệ thống.

**8. Giao diện nhập xuất hàng hóa: FormNhapXuat**

* Chức năng chính: Thực hiện các thao tác nhập và xuất hàng từ kho.
* Các trường thông tin:
  + Mã hàng hóa: Mã của hàng hóa cần nhập hoặc xuất.
  + Số lượng: Số lượng hàng hóa cần nhập hoặc xuất.
  + Loại giao dịch: Nhập hàng hay xuất hàng.
  + Nhà cung cấp hoặc cửa hàng phân phối: Tùy thuộc vào loại giao dịch là nhập hay xuất.
* Các nút:
  + Tìm kiếm hàng cần nhập hàng hoặc xuất hàng
  + Tăng giảm số lượng hàng hóa
  + Nhập thông tin nhà cung cấp
  + Nhập thông tin cửa hàng nhận
  + Xác nhận: Thực hiện việc nhập hoặc xuất hàng hóa sau khi kiểm tra thông tin.
  + Hủy: Hủy bỏ thao tác hiện tại.

**9. Giao diện kiểm đơn hàng hóa: FormHoaDon**

* Chức năng chính: Kiểm tra, xem lại các hóa đơn đã lập.
* Các trường thông tin:
  + Danh sách hóa đơn: Hiển thị danh sách các hóa đơn đã được tạo, bao gồm mã hóa đơn, ngày lập, tổng tiền.
  + Chi tiết hóa đơn: Khi chọn một hóa đơn, hiển thị chi tiết bao gồm các sản phẩm trong hóa đơn, số lượng, và giá.
* Các nút:
  + Xem chi tiết: Hiển thị chi tiết của hóa đơn được chọn.

## 3.2. Phát triển các chức năng của ứng dụng

**1. Đăng nhập hệ thống (Yêu cầu tài khoản và mật khẩu)**

Để sử dụng hệ thống, người dùng phải đăng nhập với tài khoản và mật khẩu hợp lệ.

Chỉ những người thuộc quyền quản lý kho hàng mới có thể đăng nhập và truy cập các chức năng của hệ thống, đảm bảo an ninh và quyền hạn.

**2. Tìm kiếm hàng hóa theo mong muốn**

Hệ thống cho phép người dùng tìm kiếm các mặt hàng trong kho theo các tiêu chí như tên hàng hóa, mã hàng, hoặc đặc tính khác.

Chức năng này giúp dễ dàng tra cứu thông tin về các mặt hàng trong kho để thực hiện các thao tác nhập, xuất, hoặc quản lý khác.

**3. Thêm thông tin hàng hóa mới khi có nhập hàng**

Khi có hàng hóa mới được nhập từ nhà cung cấp, người quản lý kho có thể thêm các thông tin hàng hóa vào hệ thống, bao gồm mã hàng, tên hàng, số lượng, và đơn giá.

Thông tin này sẽ được lưu trữ trong cơ sở dữ liệu và có thể được truy cập cho các thao tác sau này như xuất hóa đơn hay quản lý tồn kho.

**4. Xóa thông tin hàng hóa cũ khi không còn nhập về**

Nếu một mặt hàng không còn được nhập về kho hoặc ngừng sản xuất, người quản lý kho có thể xóa thông tin mặt hàng đó khỏi hệ thống.

Chức năng này giúp hệ thống luôn được cập nhật với thông tin hàng hóa hiện có, tránh sự tồn đọng các mặt hàng không còn được sử dụng.

**5. Xuất hóa đơn nhập hàng khi nhập hàng từ nhà cung cấp**

Mỗi khi nhập hàng từ nhà cung cấp, hệ thống sẽ tự động tạo hóa đơn nhập hàng, ghi lại thông tin về các mặt hàng đã nhập, số lượng, và giá trị của chúng.

Hóa đơn này sẽ được lưu trữ để phục vụ cho việc kiểm tra, báo cáo, và quản lý tài chính.

**6. Xuất hóa đơn xuất hàng khi chuyển hàng đến các cửa hàng**

Khi có yêu cầu chuyển hàng đến các cửa hàng, hệ thống sẽ tạo hóa đơn xuất hàng, ghi lại các sản phẩm đã xuất, số lượng, và giá trị.

Hóa đơn này cũng sẽ được lưu trữ trong hệ thống để dễ dàng theo dõi và kiểm tra các giao dịch giữa kho và các cửa hàng.

**7. Quản lý thông tin nhà cung cấp hàng hóa**

Người quản lý kho có thể thêm, xóa, và sửa thông tin về các nhà cung cấp hàng hóa, bao gồm tên, địa chỉ, và mã số nhà cung cấp.

Chức năng này giúp duy trì mối quan hệ với các nhà cung cấp và theo dõi nguồn cung cấp hàng hóa cho kho.

**8. Quản lý thông tin cửa hàng**

Tương tự như nhà cung cấp, hệ thống cho phép người quản lý kho thêm, xóa, và sửa thông tin các cửa hàng liên kết với kho.

Thông tin về cửa hàng giúp hệ thống theo dõi các giao dịch nhập xuất hàng giữa kho và cửa hàng, đảm bảo rằng hàng hóa luôn được cung cấp kịp thời.

**9. Kiểm tra lại các hóa đơn**

Người dùng có thể truy vấn và kiểm tra lại tất cả các hóa đơn đã được xuất trong hệ thống, bao gồm hóa đơn nhập hàng và xuất hàng.

Điều này giúp dễ dàng theo dõi các giao dịch, phục vụ cho việc báo cáo, kiểm toán và xử lý các vấn đề phát sinh liên quan đến các giao dịch trước đó.

## 3.3. Các kịch bản thực thi ứng dụng

**Đăng nhập:**

Người dùng cần đăng nhập đúng tài khoản, mật khẩu được mã hóa trong file nhanvien.dat

* Nhập đúng sẽ được vào hệ thống
* Nhập sai sẽ xảy ra dòng thông báo:

A white rectangular object with black text

Description automatically generated

**Thêm hàng:**

Người dùng có thể thêm ảnh từ bên ngoài vào nhưng bắt buộc phải được lưu trong file resource của đồ án.

Người dùng cần nhập thông tin hàng hóa đầy đủ

* Điền không đủ thông tin sẽ hiện thông báo

A screenshot of a computer

Description automatically generated

* Nhập sai ID sẽ hiện thông báo:

A screenshot of a computer

Description automatically generated

**Nhập Xuất hàng:**

Người dùng cần nhập thông tin của cửa hàng (nhà cung cấp) cần nhập hoặc xuất hàng đến thì mới xuất hóa đơn được nếu không sẽ có thông báo:

**A screenshot of a chat

Description automatically generated** A screenshot of a computer

Description automatically generated

**Quản lý cửa hàng và nhà cung cấp**

Muốn thêm cửa hàng (nhà cung cấp) mới thì cần phải nhấn nút thêm trước khi lưu cửa hàng (nhà cung cấp) vào hệ thống, không sẽ hiện dòng thông báo:

A screenshot of a computer

Description automatically generated

Muốn chọn cửa hàng (nhà cung cấp) để cập nhật thông tin thì cần phải nhấn chọn thông tin cửa hàng (nhà cung cấp) đó, không sẽ hiện dòng thông báo:

A screenshot of a computer

Description automatically generated

# CHƯƠNG 4. THẢO LUẬN & ĐÁNH GIÁ

## 4.1. Các Kết Quả Nhận Được

Các nội dung của mỗi giao diện đã được nếu rõ trên mục ([3.1. Thiết kế giao diện chương trình](#_3.1._Thiết_kế)) nên nhóm chúng em sẽ đưa ra ảnh minh họa giao diện trực quan khi khởi chạy chương trình

### 4.1.1. Giao diện khi khởi động chương trình (giao diện đăng nhập)

A screenshot of a computer

Description automatically generated

### 4.1.2. Giao diện trang chủ

A screenshot of a computer

Description automatically generated

### 4.1.3. Giao diện khi nhập/xuất hàng hóa

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

### 4.1.4. Giao diện khi nhập/xuất hóa đơn

A screenshot of a document

Description automatically generated A screenshot of a document

Description automatically generated

### 4.1.5. Giao diện khi thêm xóa hàng hóa

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

### 4.1.6. Giao diện quản lý thông tin cửa hàng

A screenshot of a computer

Description automatically generated

### 4.1.7. Giao diện quản lý quản lý thông tin nhà cung cấp

A screenshot of a computer

Description automatically generated

### 4.1.8. Giao diện thống kê các hóa đơn xuất nhập kho (kiểm kho)

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## 4.2. Một Số Tồn Tại

- Chưa thể phân quyền truy cập hệ thống như cấp quản lý, cấp nhân viên, cấp quản trị, v.v để với mỗi quyền hàng khác nhau thì chỉ có thể thực hiện một số chức năng nhất định

- Thiếu tính năng tìm kiếm hóa đơn theo mã hóa đơn, tên nhân viên hoặc ngày lập hoặc xóa một hóa đơn khỏi hệ thống (có thể yêu cầu quyền hạn cao hơn hoặc xác nhận từ người dùng).

- Thiếu nơi cho các thông báo quan trọng hoặc các sự kiện gần đây (như hàng mới nhập, đơn hàng cần xuất,...) để làm nổi bật hiệu quả nhận biết mặt hàng của mọi người

## 4.3. Hướng Phát Triển

**1. Phân quyền truy cập hệ thống**

* **Mô tả**: Tạo một hệ thống phân quyền để xác định các quyền truy cập khác nhau (quản lý, nhân viên, quản trị) và giới hạn những chức năng mà mỗi cấp độ có thể thực hiện.
* **Giải pháp**:
  + **Thêm bảng quyền trong cơ sở dữ liệu**: Tạo bảng phân quyền chứa các quyền truy cập khác nhau, chẳng hạn như "Quản lý", "Nhân viên", "Quản trị", v.v. Bảng này nên liên kết với người dùng để xác định quyền của từng người.
  + **Kiểm tra quyền khi truy cập chức năng**: Ở mỗi chức năng của giao diện, kiểm tra quyền của người dùng trước khi cho phép truy cập hoặc thực hiện hành động. Ví dụ, chỉ người dùng có quyền "Quản trị" mới có thể xóa hóa đơn hoặc xóa hàng hóa khỏi kho.
  + **UI cho phân quyền**: Tạo giao diện quản trị để thiết lập quyền cho từng người dùng, giúp quản lý dễ dàng điều chỉnh quyền truy cập của nhân viên.
  + **Hướng dẫn người dùng**: Cung cấp hướng dẫn rõ ràng về các chức năng mà mỗi cấp độ quyền có thể thực hiện để tránh nhầm lẫn và sử dụng sai.

**2. Cải thiện tính năng tìm kiếm và xóa hóa đơn**

* **Mô tả**: Tăng cường tính năng tìm kiếm hóa đơn theo mã hóa đơn, tên nhân viên hoặc ngày lập, đồng thời cho phép xóa hóa đơn với quyền hạn cao.
* **Giải pháp**:
  + **Bổ sung trường tìm kiếm trên giao diện FormHoaDon**: Thêm các trường để tìm kiếm hóa đơn bằng mã hóa đơn, tên nhân viên, hoặc ngày lập. Hỗ trợ bộ lọc hoặc kết hợp nhiều tiêu chí để cải thiện trải nghiệm người dùng.
  + **Thêm xác nhận quyền trước khi xóa**: Khi người dùng cố gắng xóa hóa đơn, kiểm tra quyền của họ trước khi thực hiện thao tác. Nếu họ không có quyền, hiển thị thông báo từ chối. Người có quyền cao hơn có thể nhận thông báo yêu cầu xác nhận xóa từ nhân viên khác.
  + **Xác nhận trước khi xóa**: Cung cấp một hộp thoại xác nhận để người dùng kiểm tra lại thông tin hóa đơn trước khi xóa, nhằm tránh các thao tác nhầm lẫn.

**3. Tạo nơi hiển thị thông báo quan trọng và sự kiện gần đây**

* **Mô tả**: Cung cấp khu vực dành riêng để hiển thị các thông báo quan trọng hoặc sự kiện (hàng mới nhập, đơn hàng cần xuất).
* **Giải pháp**:
  + **Khu vực thông báo trên trang chủ FormTrangChu**: Thêm một khung hoặc danh sách hiển thị các thông báo quan trọng. Khu vực này sẽ cập nhật theo thời gian thực hoặc mỗi lần người dùng truy cập trang chủ.
  + **Loại thông báo**:
    - **Hàng mới nhập**: Hiển thị khi có hàng hóa mới được nhập vào kho, giúp người dùng nắm bắt kịp thời.
    - **Đơn hàng cần xuất**: Hiển thị các đơn hàng đến hạn xuất, nhắc nhở người dùng thực hiện đúng thời gian.
    - **Thông báo bảo trì**: Thông báo khi có các thay đổi hoặc bảo trì hệ thống.
  + **Tạo thông báo nổi bật**: Với các thông báo khẩn cấp, bạn có thể dùng hiệu ứng nổi bật hoặc màu sắc khác để thu hút sự chú ý của người dùng.
  + **Thiết lập và quản lý thông báo**: Cung cấp quyền cho cấp quản trị để tạo hoặc xóa thông báo, đồng thời cài đặt thời gian hiển thị của mỗi thông báo.

# PHỤ LỤC

- Mã nguồn của chương trình: <https://github.com/tundut/QuanLyNhapXuatKho>

- Hướng dẫn cách cài đặt để chạy

**Sử dụng GIT và GITHUB:**

**B1: Tạo tài khoản GITHUB (link** [**GitHub**](https://github.com/)**)**

**B2: Tải công cụ GIT (link:** [**Git - Downloads**](https://git-scm.com/downloads)**) và khởi động chương trình**

**B3: Khai báo thông tin với GIT bằng cách mở GitBash lên và chạy 2 câu lệnh**

*git config --global user.name "Your Name"* ***(you name thay bằng tên của mình)***

*git config --global user.email yourname@example.com***(yourname@example.com thay bằng mail đã tạo tài khoản trên github)**

A screenshot of a computer

Description automatically generated

**B4: Chọn đến nơi cần lưu trữ file và click chuột phải chọn GitBash**

**A screenshot of a computer

Description automatically generated**

**và gõ lệnh sau để clone code về:**

*git clone https://github.com/tundut/QuanLyNhapXuatKho*

**A screenshot of a computer

Description automatically generated**

- Phân công công việc

|  |  |
| --- | --- |
| **Thành Viên** | **Nhiệm Vụ** |
| Nguyễn Phan Tuấn Đức *(trưởng nhóm)* | * Định hướng đề tài cho cả nhóm * Code 2 class trọng tâm của cả bài toán: KhoHang, QuanLyNhapXuat * Thiết kế:   Giao diện quản lý cửa hàng phân phối: FormCuaHang  Giao diện phiếu hóa đơn: FormPhieuHoaDon  HangHoaBillComponent  BillTailComponent  Giao diện nhập xuất hàng hóa: FormNhapXuat  HangHoaNhapXuatComponent  HangHoaLoComponent  Giao diện kiểm đơn hàng hóa: FormHoaDon   * Làm báo cáo nội dung chương 2 |
| Nguyễn Hoàng Bảo | * Code 3 class đối tượng quản lý: NhaCungCap, NhanVien, CuaHang * Thiết kế:   Giao diện quản lý nhà cung cấp: FormNhaCungCap  Giao diện quản lý cửa hàng phân phối: FormCuaHang  Giao diện kiểm đơn hàng hóa: FormHoaDon   * Làm báo cáo nội dung chương 1 |
| Huỳnh Kim Nguyên | * Code 4 class về sản phẩm của kho: Hang Hoa, ThucPham, GiaDung, DienTu * Thiết kế:   Giao diện đăng nhập hệ thống: FormDangNhap  Giao diện trang chủ: FormTrangChu  HangHoaTrangChuComponent   * Làm báo cáo nội dung chương 3 |
| Võ Đức Hưng | * Code 3 class hóa đơn nhập xuất: HoaDon, HoaDonNhap, HoaDonXuat * Thiết kế:   Giao diện nền: Form  Giao diện thêm hàng: FormHangHoa   * Làm báo cáo nội dung chương 4 * Thiết kế slide để thuyết trình |

# TÀI LIỆU THAM KHẢO

1. *Chat Blackbox: AI Code Generation, Code Chat, Code Search - Blackbox*. (n.d.). Retrieved April 17, 2024, from https://www.blackbox.ai/

2. *How Kteam - Free Education | How Kteam*. (n.d.). Retrieved April 17, 2024, from https://howkteam.vn/

3. *GitHub*. (n.d.). Retrieved April 17, 2024, from https://github.com/

4. *Demos | Guna Framework - .NET UI/UX Controls and Components for Developers of Desktop, Reporting, Data Visualization*. (n.d.). Retrieved April 17, 2024, from https://gunaui.com/demos/

5. *UI Design Music Player Application - Winform C# Guna Framework - YouTube*. (n.d.). Retrieved April 17, 2024, from https://www.youtube.com/watch?v=ZH8X5zlCj6A