ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

------------



BÁO CÁO ĐỒ ÁN

CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

CHỦ ĐỀ:

**OBJECT MOVING**

Nhóm:

Vòng Chí Tài - 1512474

Nguyễn Duy Tâm - 1512479

Nguyễn Đào Xuân Trường - 1512624

Tôn Thất Vĩnh - 1512679

Mục lục

[1. Cấu trúc chung của ứng dụng: 3](#_Toc484526543)

[2. Chi tiết cài đặt ứng dụng: 4](#_Toc484526544)

[2.1. Level 1 - Đọc dữ liệu và thể hiện trên – Console: 4](#_Toc484526545)

[2.2. Level 2 - Di chuyển kiện hàng – Console: 4](#_Toc484526546)

[2.3. Level 3 - Đồ họa GUI cho ứng dụng: 5](#_Toc484526547)

[2.3.1. Hiển thị dữ liệu lên màn hình: 5](#_Toc484526548)

[2.3.2. Cập nhật (refresh) màn hình: 5](#_Toc484526549)

[2.3.3. Chức năng Zoom In và Zoom Out: 6](#_Toc484526550)

[2.3.4. Di chuyển kiện hàng: 7](#_Toc484526551)

[2.4. Level 4 – Grow up – Xem thông tin một khu vực: 8](#_Toc484526552)

[2.4.1. Hiển thị khu vực được chọn: 8](#_Toc484526553)

[2.4.2. Xem thông tin và chỉnh sửa thông tin kiện hàng: 10](#_Toc484526554)

[2.5. Level 5 – Super man – Đồng bộ dữ liệu: 12](#_Toc484526555)

[3. Hướng dẫn sử dụng: 12](#_Toc484526556)

[3.1. Console: 12](#_Toc484526557)

[3.2. Giao diện GUI: 13](#_Toc484526558)

[4. Kết quả: 15](#_Toc484526559)

[4.1. Một số testcase: 15](#_Toc484526560)

[4.2. Mức độ hoàn thành đồ án: 15](#_Toc484526561)

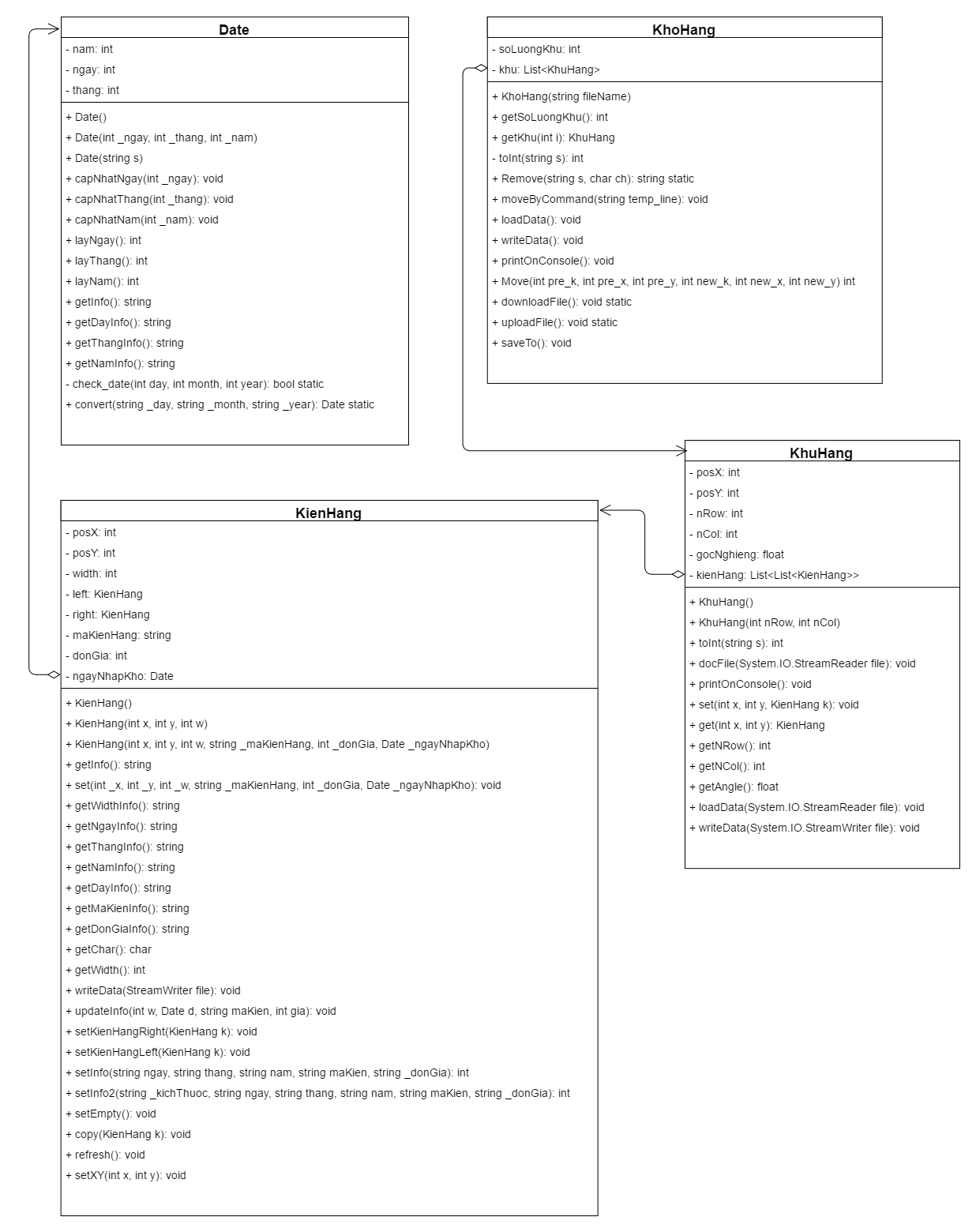
[5. Đánh giá hoạt động của nhóm: 15](#_Toc484526562)

[5.1. Kế hoạch, phân công nhiệm vụ: 15](#_Toc484526563)

[5.2. Mức độ đóng góp của các thành viên: 16](#_Toc484526564)

[Tài liệu tham khảo 17](#_Toc484526565)

# Cấu trúc chung của ứng dụng:



* Đồ án được thực hiện bằng ngôn ngữ **C#.**

# Chi tiết cài đặt ứng dụng:

## Level 1 - Đọc dữ liệu và thể hiện trên – Console:

* Theo mục 1:
  + **KienHang**: cấu trúc lưu trữ thông tin của một kiện hàng.
    - **posX, posY:** vị trí của kiện hàng trong khu.
    - **width**: kích thước kiện hàng.
    - Các thông tin phụ : mã kiện hàng, đơn giá, ngày nhập kho.
  + **KhuHang**: cấu trúc lưu trữ thông tin của một khu hàng.
    - **posX, posY:** vị trí trên bản đồ.
    - **nRow**, **nCol**: số kiện hàng ở mỗi chiều tương ứng.
    - **kienHang**: danh sách các kiện hàng mà khuHang này quản lý.
  + **KhoHang**: cấu trúc lưu trữ thông tin của một kho hàng.
    - **soLuongKhu**: số lượng khu hàng trong kho.
    - **khu**: danh sách các khu hàng được quản lý trong kho hàng.
* Thông tin về kho hàng được lưu trữ trong file **input**.**txt** (chi tiết cấu trúc ở mục 1)
* Đọc dữ liệu:
  + Lúc khởi tạo kho hàng:
    - Đọc thông tin về số khu hàng từ file.
    - Khởi tạo số khu hàng tương ứng.
    - Việc đọc thông tin khu hàng đưa về cho các khu hàng tương ứng thực hiện.
  + Đọc thông tin khu hàng:
    - Các khu hàng đọc thông tin rồi khởi tạo danh sách các kiện hàng mình quản lý.
    - Đọc thông tin tương ứng của các kiện hàng mà khu hàng quản lý, cập nhật thông tin kiện hàng tương ứng.
* Thể hiện trên Console:
  + Các kiện hàng đều có trạng thái riêng của mình (0 – không có hàng, 1 – có kiện hàng một ô, 2 – có kiện hàng 2 ô).
  + Duyệt thông tin toàn bộ các kiện hàng, đưa thông tin tương ứng lên màn hình console.
* Cài đặt cụ thể:
  + Đọc dữ liệu: **KhoHang.loadData(), KhuHang.docFile(…).**
  + Xuất dữ liệu: **KhoHang.printOnConsole(), KhuHang.printOnConsole().**

## Level 2 - Di chuyển kiện hàng – Console:

* Cấu trúc câu lệnh điều khiển: **M [k0] (x0,y0) [k1] (x1,y2)**
  + k0, x0, y0: tương ứng với khu hàng và vị trí của kiện hàng cần di chuyển.
  + k1, x1, y2: tương ứng với khu hàng và vị trí của kiện hàng cần di chuyển đến.
* Một số lỗi không lệ của lệnh nhập vào:
  + Cú pháp không đúng.
  + Khu vực hay kiện hàng không tồn tại
  + Khu vực hay vị trí đặt không phù hợp như đã có kiện hàng ở đó hay kích thước không phù hơp để đặt.
* Cài đặt:
  + Lệnh được nhập vào được kiểm tra xem có đúng cú pháp yêu cầu hay không. Nếu đúng thì chuyển các giá trị tương ứng trong lệnh được nhập vào lưu trữ vào các biến tương ứng.
  + Thực hiện việc di chuyển kiện hàng theo lệnh được nhập vào. Cài đặt cần quan tâm
    - Khu hàng có tồn tại hay không.
    - Vị trí của kiện hàng xuất phát hay vị trí đích có hợp lệ hay không (có vượt quá giới hạn của khu hàng tương ứng không).
    - Kiểm tra vị trí đích đã tồn tại hàng chưa (kiểm tra ô kề cạnh nếu kiện hàng di chuyển độ dài 2).
    - Nếu các yêu cầu trên thỏa, thực hiện di chuyển kiện hàng.
  + Cập nhật lại thông tin của kho hàng, vẽ lại bản đồ kho lên màn hình.
* Cài đặt cụ thể: xem các hàm **KhoHang.Move(…).**

## Level 3 - Đồ họa GUI cho ứng dụng:

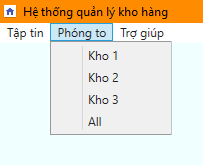
### Hiển thị dữ liệu lên màn hình:

* Khởi tạo khu vực vẽ (DrawArea) có kiểu ScrollView để thuận lợi cho việc thu phóng hình vẽ
* Khởi tạo một StackPanel để lưu trữ kho hàng gồm các khu hàng
* Khởi tạo các WrapPanel để lưu trữ các khu hàng gồm các KButton (Một Button được tùy chỉnh để lưu dữ liệu kiện hàng)
* Thiết lập ban đầu:
  + DrawArea.HorizontalScrollBarVisibility = ScrollBarVisibility.Auto; Thiết lập tự động ẩn hiện thanh cuộn ngang
  + DrawArea.VerticalScrollBarVisibility = ScrollBarVisibility.Auto; Thiết lập ẩn hiện thanh cuộn dọc
  + sp.Orientation = Orientation.Horizontal; Thiết lập hướng cho StackPanel để co dãn theo chiều ngang để tiện xử lý
* Cài đặt:
  + Ứng với mỗi kho hàng tạo một WrapPanel và add các KButton để thể hiện dữ liệu và dàn trang hợp lý
  + Xoay khu hàng bằng các tạo một RotateTransform và gán vào LayoutTransform để áp dụng lên tất cả phần tử trong khu
* Cài đặt cụ thể: Xem hàm **GenerateMap()**

### Cập nhật (refresh) màn hình:

* Khởi tạo biến có kiểu: System.Windows.Threading.DispatcherTimer dispatcherTimer.
* Thiệt lập ban đầu:
  + dispatcherTimer.Tick += new EventHandler(dispatcherTimer\_Tick); Thêm hàm xử lý refresh màn hình sau một khoảng thời gian (hàm dispatcherTimer\_Tick).
  + dispatcherTimer.Interval = new TimeSpan(0, 0, 10); Đặt thời gian refresh màn hình là 10s.
  + dispatcherTimer.Start(); : Bắt đầu tính thời gian.
* Cài đặt:
  + Hàm dispatcherTimer\_Tick được gọi sau mỗi khoảng thời gian t được thiết lập trên.
  + Hàm này thực hiện việc cập nhật thông tin về kiện hàng trong mỗi button, về background của button. Sau đó vẽ lại toàn bộ thông tin lên màn hình.
* Cài đặt cụ thể: Xem hàm **MainWindow.dispatcherTimer\_Time(…)**

### Chức năng Zoom In và Zoom Out:

* Zoom In và Zoom Out vào một khu hàng:
  + Người dùng chọn khu hàng trong Menu “Phóng to” để zoom in một khu hàng
  + Người dùng chọn “All” trong menu “Phóng to” để trở lại trạng thái ban đầu.
  + 
  + Cài đặt:
    - Tạo một **MenuItem** chứa các danh sách khu hàng trong kho.
    - Tạo một xử lý event **zoom\_1kho(…)** để xử lý việc nhảy vào một khu vực quan tâm (việc click vào tên khu hàm). Trong hàm này, thông tin về khu quan tâm được lưu lại, sao đó gọi hàm **GenerateMapZoom1Khu(…)** để vẽ lại khu hàng đó trên màn hình.
    - Tạo một xử lý event **zoom\_all(…)** tương ứng với việc người dùng click vào mục All (quay lại màn hình làm việc chính). Hàm này load thông tin của tất cả các kiện hàng rồi gọi hàm **GenerateMap()** để lại toàn bộ kho hàng.
    - Khu hàng được vẽ lại có vị trí chính giữa màn hình, việc thay đổi thông tin các kiện hàng ở đây cũng sẽ được lưu lại khi người dùng trở về màn hình làm việc chính.
  + Cài đặt cụ thể: Xem các hàm:
    - **MainWindow.zoom\_1kho(…), MainWindow.GenerateMapZoom1Khu(…).**
    - **MainWindow.zoom\_all(…), MainWindow.GenerateMap().**
* Hiển thị một khu vực được chọn bằng kéo thả chuột: Xem mục 2.4.1.
* Phóng to, thu nhỏ màn hình:
  + Ở góc dưới phải màn hình có một Slider, người dùng dùng chuột để kéo trên thanh đó để thay đổi kích thước màn hình.
  + C:\Users\Vinh Ton That\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot (108).png
  + Cài đặt:
    - Tạo một Slider **sld.**
    - Tạo một hàm xử lý event kẻo thả trên sld (**sldZoom\_ValueChanged**). Trong hàm này, thông tin trên thanh slider được lấy, dựa vào thông tin này, ta suy ra được độ phóng to, thu nhỏ màn hình. Từ đó đưa thông tin này vào trong thuộc tính Transform, sau đó cập nhật lại màn hình.
    - Độ phóng to, thu nhỏ: 0.5 -> 1.5.
  + Cài đặt cụ thể: Xem hàm **MainWindow.sldZoom\_ValueChanged(…).**

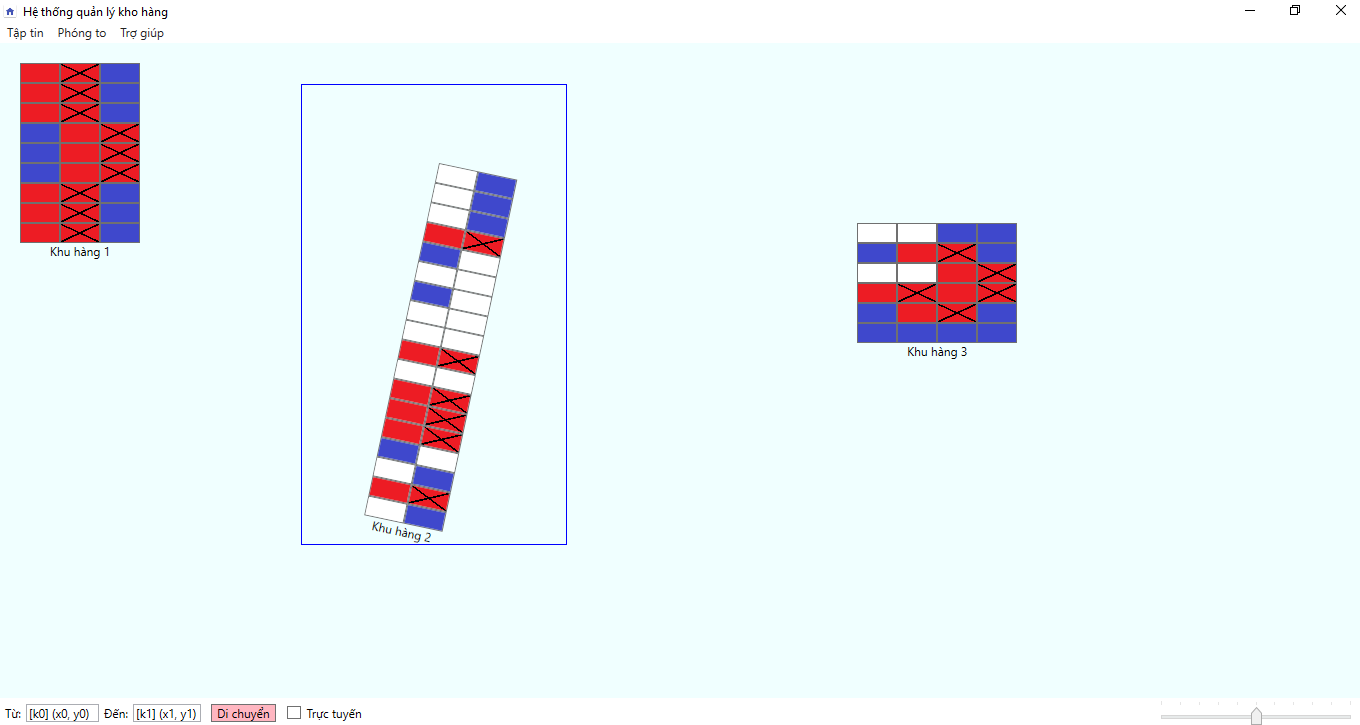
### Di chuyển kiện hàng:

* Cấp độ 1: Chọn ô có kiện hàng bắt đầu, chọn ô vị trí cần đến. Sau đó thực hiện di chuyển.
  + Giao diện:
    - C:\Users\Vinh Ton That\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot (98).png
    - Cấu trúc:
      * Gồm 2 TextBox:
        + **startPos**: lưu trữ thông tin vị trí xuất phát của kiện hàng – tương ứng với TextBox đầu tiên.
        + **endPos**: lưu trữ thông tin vị trí kết thúc của kiện hàng – tương ứng với TextBox thứ hai.
      * Một Button – **move**: thực hiện lệnh di chuyển với thông tin về vị trí xuất phát và kết thúc từ 2 TextBox trên.
  + Cài đặt:
    - Tạo một **TextBox selTb:** là một TextBox tạm thời ***lưu trữ TextBox nào đang được chọn (được trỏ đến).***
    - Tạo hàm xử lý event **GotMouseCapture** cho hai TextBox (**startPos** và **endPos**). Khi event trên được xử lý (tức là ô textbox được trỏ chuột đến), selTb (định nghĩa trên) sẽ **lưu lại TextBox nào gọi hàm xử lý event này**, từ đó ta biết được mình đang trỏ đến TextBox nào.
    - Tạo hàm xử lý event **GotMouseCapture** cho **button** (tương ứng với các kiện hàng). Khi event này được xử lý (tức là button nào được bấm vào), ta **lấy thông tin về vị trí kiện hàng của button được trỏ đến, lưu vào Text của selTb,** như vậy, TextBox tương ứng đang được trỏ đến sẽ được cập nhật thông tin.
    - Tạo hàm xử lý event **Click** cho button move. Khi event này được xử lý (tức là người dùng bấm vào nút move), ta sẽ gọi hàm di chuyển kiện hàng (tương tự ở level2 – phần 2.2), sau khi di chuyển, cập nhật lại thông tin của kiện hàng, thông tin các button tương ứng, cập nhật lại màn hình. Nếu lệnh di chuyển lỗi, giữ nguyên trạng thái hiện tại và thông báo lỗi.
  + Cài đặt cụ thể, xem:
    - **MainWindow.EndPos\_GotMouseCapture(…), MainWindow.StartPos\_GotMouseCapture(…).**
    - **MainWindow.Btn\_GotMouseCapture(…).**
    - **MainWindow.move\_Click(…);**
    - **Hàm khởi tạo MainWindow.**
* Cấp độ 2: Người dùng nhấn giữ chuột để di chuyển sang vị trí khác.
  + Cài đặt:
    - Tạo phương thức **PreviewMouseMove** để theo dõi việc di chuyển chuột, nếu đã bấm chuột trái thì ta lưu lại giá trị của Button vừa click, và theo dõi điểm đến bằng phương thức **DragDrop.DoDragDrop().**
    - Gán thuộc tính **AllowDrop** vào KButton để cho phép Drop.
    - Thiết lập phương thức Drop cho Button, đồng thời gán vào từng Button. Khi hàng được drop vào sẽ kích hoạt Event này. Ta thực hiện di chuyển và gán giá trị của Button đang theo dõi về null.
  + Cài đặt cụ thể, xem:
    - **MainWindow.Btn\_Drop(…)**
    - **MainWindow.Btn\_PreviewMouseMove(…)**
    - **Hàm khởi tạo MainWindow**

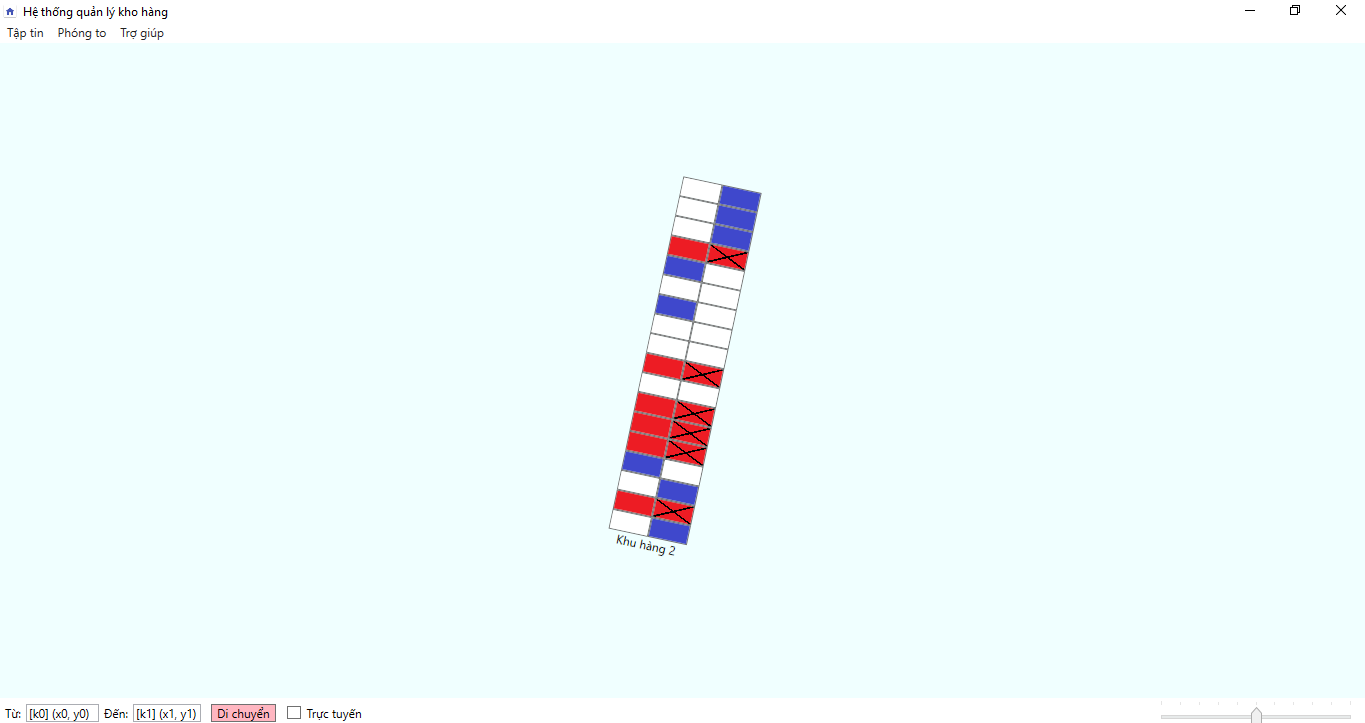
## Level 4 – Grow up – Xem thông tin một khu vực:

### Hiển thị khu vực được chọn:

* Dùng **chuột phải** để chọn một khu vực trên màn hình, khu hàng nào có kiện hàng nằm trong vùng được chọn sẽ được phóng to (nếu có nhiều khu thì chọn khu trái trên nhất).
* Giao diện:
  + Chọn 1 vùng:

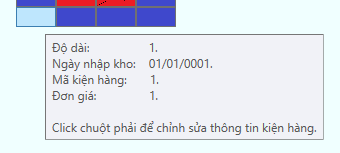
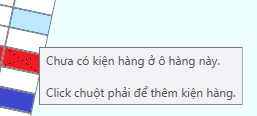
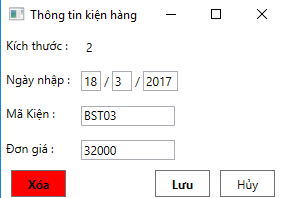
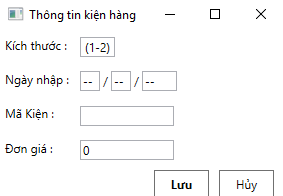


* + Kết quả:



* Cài đặt:
  + Cài đặt các hàm xử lý các event liên quan đến việc chọn một khu vực trên màn hình (**Grid\_MouseDown, Grid\_MouseUp, Grid\_MouseMove**), các hàm này sẽ lấy thông tin về tọa độ góc trái trên cũng như kích thước ngang và dọc của khu vực được chọn.
  + Từ thông tin trên, ta tìm ra kiện hàng nào nằm trong khu vực được chọn, đồng thời là kiện hàng ở góc trái trên của khu vực đó, thì vẽ lại toàn bộ khu hàng chứa kiện hàng đó lên màn hình. (Dùng các hàm vẽ ở mục 2.3.3.)
* Cài đặt cụ thể: Xem các hàm sau:
  + **MainWindow.Grid\_MouseDown(…).**
  + **MainWindow.Grid\_MouseUp(…).**
  + **MainWindow.Grid\_MouseMove(…).**

### Xem thông tin và chỉnh sửa thông tin kiện hàng:

* Xem thông tin:
  + 
  + 
  + Thêm hàm xử lý event **Btn\_MouseEnter** (để xử lý khi con trỏ chuột trỏ đến một button).
  + Trong hàm xử lý trên, ta **cập nhật thông tin về ToolTip** tương ứng với thông tin của kiện hàng mà button này đang quản lý.
  + Sau khi thiết lập giá trị cho ToolTip, thông tin về button sẽ được hiển thị trên màn hình khi con trỏ chuột trỏ đến button tương ứng.
* Chỉnh sửa thông tin:
  + **Click phải** vào ô tương ứng để chỉnh sửa thông tin.
  + Với ô tương ứng đã có kiện hàng, click phải để chỉnh sửa thông tin hoặc xóa kiện hàng:
    - Sau khi click chuột, một bảng như sau hiện ra:
    - 
    - Sau khi thay đổi thông tin, bấm **Lưu** để lưu kết quả. (Nếu các thông tin nhập vào hợp lệ, kết quả sẽ được lưu, ngược lại, thông báo lỗi sẽ xuất hiện).
    - Bấm **Xóa** để xóa kiện hàng này.
    - Bấm **Hủy** để bỏ các thay đổi.
  + Giao diện:
    - Gồm các **TextBox** để người dùng nhập thông tin vào.
    - **Button save** để lưu kết quả.
    - **Button delete** để xóa thông tin kiện hàng.
    - **Button cancel** để xóa thông tin kiện hàng.
  + Cài đặt:
    - Tạo một **Window** (tương ứng trong project là PopUp) với giao diện tương ứng như hình trên. (TextBox kích thước được ẩn đi).
    - Tạo hàm xử lý cho event tương ứng với việc bấm nút “**Lưu**” (xem hàm **PopUp.save\_click(…)**). Hàm này lấy các thông tin ở các TextBox trong window, nếu các thông tin này hợp lệ, thì cập nhật các thông tin này cho kiện hàng được button tương ứng lưu trữ (cập nhật kiện hàng bên cạnh nếu kiện hàng đang xét độ dài 2). Sau đó tắt window đang mở.
    - Tạo hàm xử lý cho event tương ứng với việc bấm nút “**Xóa**” (xem hàm **PopUp.delete\_click(…)**). Hàm này đặt lại thông tin của kiện hàng được trỏ đến bởi button tương ứng thành kiện hàng rỗng (cập nhật kiện hàng bên cạnh nếu kiện hàng đang xét độ dài 2). Sau đó tắt window đang mở
    - Tạo hàm xử lý cho event tương ứng với việc bấm nút “**Hủy**”, tắt window đang mở.
* Thêm kiện hàng vào kho:
  + Click **phải** vào ô trống để thêm kiện hàng.
    - Sau khi click chuột, một bảng như sau hiện ra:
    - 
    - Sau khi thay đổi thông tin, bấm **Lưu** để lưu kết quả. (Nếu các thông tin nhập vào hợp lệ, kết quả sẽ được lưu, ngược lại, thông báo lỗi sẽ xuất hiện).
    - Bấm **Hủy** để bỏ các thay đổi.
  + Giao diện: (tương tự cài đặt của Thay đổi thông tin kiện hàng – button Xóa được ẩn).
    - Gồm các **TextBox** để người dùng nhập thông tin vào.
    - **Button save** để lưu kết quả.
    - **Button delete** để xóa thông tin kiện hàng.
  + Cài đặt: (tương tự cài đặt của Thay đổi thông tin kiện hàng).
    - Tạo một **Window** (tương ứng trong project là PopUp) với giao diện tương ứng như hình trên. (TextBlock ở kích thước được ẩn đi).
    - Tạo hàm xử lý cho event tương ứng với việc bấm nút “**Lưu**” (xem hàm **PopUp.save\_click(…)**). Hàm này lấy các thông tin ở các TextBox trong window, nếu các thông tin này hợp lệ, thì cập nhật các thông tin này cho kiện hàng được button tương ứng lưu trữ (cập nhật kiện hàng bên cạnh nếu kiện hàng đang xét độ dài 2). Sau đó tắt window đang mở
    - Tạo hàm xử lý cho event tương ứng với việc bấm nút “**Hủy**”, tắt window đang mở.

## Level 5 – Super man – Đồng bộ dữ liệu:

FTP:

* Khởi tạo hàm downloadFile() và uploadFile().
  + downloadFile(); sử dụng WebClient để truy cập máy chủ và lấy file input.txt qua giao thức http để tối ưu tốc độ. Download toàn bộ nội dung file và copy vào file input.txt trong thư mục Resource và tiến hành load lại. Ta đặt bộ đếm giờ cứ 10s sẽ refresh và đồng bộ cùng server
  + uploadFile(); sử dụng FtpWebRequest để truy cập máy chủ và tiến hành upload. Để tránh các lỗi conflict ta đổi tên input.txt trên server thành input2.txt, và tiến hành upload file input.txt trong thư mục Resource lên server.
* Cài đặt:
  + downloadFile():
    - Khởi tạo một WebClient, thiết lập kết nối đến máy chủ và tiến hành download string lưu vào string tạm thời
    - Khởi tạo một StreamWrite, mở file input trong thư mục Resources và ghi nội dung vào và tiến hành đồng bộ ở ngoài màn hình.
  + uploadFile():
    - Khởi tạo một FtpWebRequest, thiết lập kết nối đến máy chủ hiện hành.
    - Thiết lập các định danh bảo mật bằng Credential để nhận quyền truy cập file
    - Thiết lập kiểu truy vấn là Rename và UploadFile để đổi tên và upload file input lên server
    - Khởi tạo một StreamReader để đọc dữ liệu từ file và Encode vào chuỗi dữ liệu và gửi lên server thông qua một Stream dữ liệu
    - Nhận phản hồi từ server và xử lý
* Cài đặt cụ thể, xem:
  + **KhoHang.downloadFile(…)**
  + **KhoHang.uploadFile(…)**

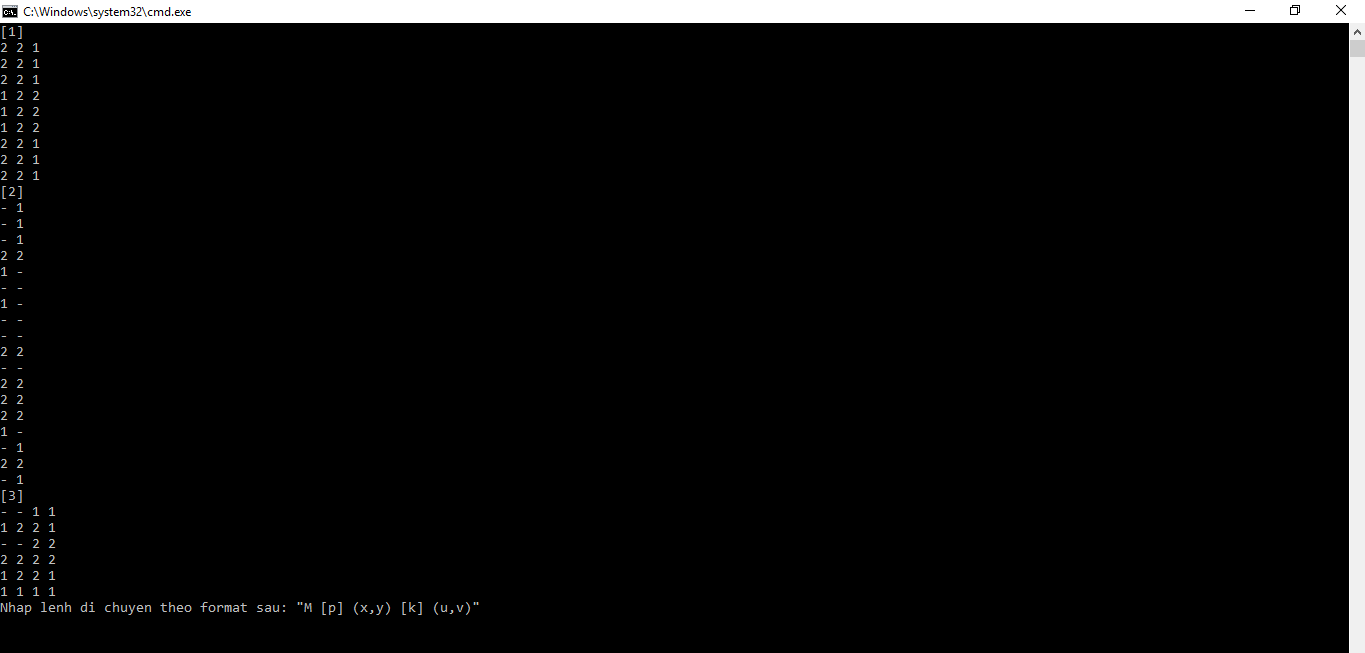
Socket.IO:

* Khởi tạo server đơn giản, xem thêm file server.js. Khởi động server xử lý một số sự kiện đơn giản để kết nối các client vào một nhóm và xử lý các request gửi data lên và truyền sang các máy khác.
* Trong chương trình, ta sử dụng công cụ Socket.IO được cung cấp miễn phí để kết nối với server Nodejs
  + emitData(); sử dụng hàm emit của class Socket được cung cấp sẵn để tiến hành gửi gói tin lên server
  + listenData(); sử dụng hàm listen on có sẵn của class Socket để tiến hành listen thông tin từ server khi các client up lên
* Cài đặt:
  + listenData();
    - Khởi tạo một đối tượng Socket để gửi nhận thông tin, thiết lập các listener sẵn sàng nhận thông tin từ máy chủ
    - Khởi tạo một StreamWrite, mở file input trong thư mục Resources và ghi nội dung vào và tiến hành đồng bộ ở ngoài màn hình.
  + emitData():
    - Sử dụng đối tượng Socket trên để gửi data lên server. Server có nhiệm vụ xử lý và gửi cho các đối tượng trong group để tiến hành đồng bộ.
    - Nhận phản hồi từ server và xử lý
* Cài đặt cụ thể, xem:
  + **KhoHang.listenData(…)**
  + **KhoHang.emitData(…)**

# Hướng dẫn sử dụng:

## Console:

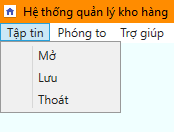
* Trong project, đặt **ObjectMovingConsole** là **Startup Project.**
* Giao diện:



* Ban đầu bản đồ sẽ được vẽ trên màn hình.
* Để tương tác, người dùng sẽ nhập lệnh để tương tác với chương trình.
* Cấu trúc lệnh: M [k0] (x0,y0) [k1] (x1,y1).
  + k0, x0, y0: khu hàng và tọa độ của kiện xuất phát.
  + k1, x1, y1: khu hàng và tọa độ cần được di chuyển đến của kiện xuất phát.
* Nếu di chuyển thành công, bản đồ sẽ được vẽ lại, người dùng tiếp tục tương tác.
* Nếu di chuyển không thành công, chương trình sẽ báo lỗi cụ thể với người dùng, các lỗi cụ thể sau:
  + Nhập sai cú pháp
  + Đích đến không hợp lệ (đã tồn tại hàng ở đích đến).
  + Không có hàng ở ô xuất phát.
  + Tọa độ vượt quá giới hạn (có thể tọa độ nguồn hoặc đích sai).

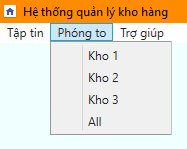
## Giao diện GUI:

* Các menu:
  + Menu Tập tin:



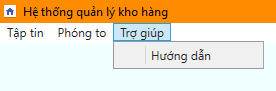
Dùng để thao tác với tập tin chứa dữ liệu kho hàng và thoát chương trình

* + Menu Phóng to:



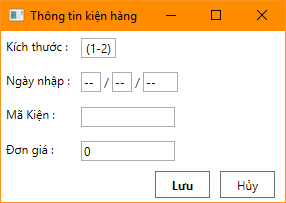
Dùng để zoom in một kho hàng. All để thoát zoom in, trở lại tất cả kho hàng.

* + Menu Trợ giúp:



Chứa hướng dẫn sử dụng:

* Thêm một kiện hàng: Chuột phải vào một ô trống và nhập thông tin của kiện hàng cần thêm:



* Di chuyển các kiện hàng: thực hiện 1 trong 2 cách sau:
  + Sử dụng thanh công cụ:



* + - Nhập ô cần được di chuyển vào ô Từ:
    - Nhập ô cần di chuyển đến vào ô Đến:
    - Nhấn nút di chuyển
  + Kéo thả các kiện hàng trên giao diện:
* Sử dụng thanh cuộn để phóng to – thu nhỏ:



* Đồng bộ hóa dữ liệu:



* + Bấm chọn vào ô Trực tuyến để đồng bộ trực tuyến
  + Bỏ chọn ô Trực tuyến để thao tác ngoại tuyến

# Kết quả:

## Một số testcase:

Xem file **Testcase.docx** đính kèm.

## Mức độ hoàn thành đồ án:

Các level hoàn thành 100%:

1. Đọc dữ liệu và thể hiện trên Console: 100%.
2. Di chuyển kiện hàng trên Console: 100%.
3. Đồ họa GUI cho ứng dụng: 100%.
4. Grow Up (Xem một khu vực): 100%.
5. Đồng bộ dữ liệu: 100%.

# Đánh giá hoạt động của nhóm:

## Kế hoạch, phân công nhiệm vụ:

* Tuần 1: (22/5 – 28/5):
  + Tìm hiểu về việc lập trình giao diện trên C#: Tài, Vĩnh.
  + Thực hiện yêu cầu 1, 2 (trên màn hình console) của đồ án: Trường, Tâm.
* Tuần 2: (29/5 – 4/5):
  + Thiệt lập cài đặt tổng quát (giao diện, các lớp KButton,…): Tài, Vĩnh.
  + Vẽ kho hàng lên màn hình: Tài.
  + Thêm, xóa, xem và chỉnh sửa thông tin kiện hàng: Vĩnh.
  + Thực hiện chức năng kéo thả kiện hàng trên màn hình: Tài.
  + Cập nhật file lưu trữ thông tin trạng thái kho hàng: Vĩnh.
  + Thực hiện chức năng chọn 2 điểm rồi di chuyển kiện hàng: Vĩnh.
  + Tìm hiểu và cài đặt server: Tài.
  + Thực hiện chức năng Zoom In và Zoom out một khu vực: Trường.
  + Thực hiện chức năng phóng to, thu nhỏ màn hình bằng slider: Trường.
* Tuần 3: (5/5 – 6/5):
  + Kiểm tra, đánh giá: Tâm.
  + Điều chỉnh lại giao diện: Tài.
  + Viết báo cáo: Vĩnh, Trường.
* Chi tiết: <https://github.com/thatvinhton/ObjectMoving/commits/master>

## Mức độ đóng góp của các thành viên:

|  |  |  |
| --- | --- | --- |
| Vòng Chí Tài | 1512474 | 35% |
| Nguyễn Duy Tâm | 1512479 | 15% |
| Nguyễn Đào Xuân Trường | 1512624 | 20% |
| Tôn Thất Vĩnh | 1512679 | 30% |

# Tài liệu tham khảo

|  |  |
| --- | --- |
| [1] | "C# Programming Guide," [Online]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/index. |
| [2] | "C# Tutorials," [Online]. Available: https://msdn.microsoft.com/en-us/library/aa288436(v=vs.71).aspx. |
| [3] | "Introduction to WPF," [Online]. Available: https://msdn.microsoft.com/en-us/library/aa970268(v=vs.100).aspx. |
| [4] | "Stack overflow," [Online]. Available: https://stackoverflow.com/. |
| [5] | "Designing a User Interface (Visual C#)," [Online]. Available: https://msdn.microsoft.com/en-us/library/ms173080(v=vs.90).aspx. |
| [6] | "C# Graphical User Interface Tutorial," [Online]. Available: http://csharp.net-informations.com/gui/gui\_tutorial.htm. |
| [7] | "C Sharp GUI Windows Tutorials," [Online]. Available: http://www.tutorialspoint.com/listtutorials/c-sharp/gui-windows/1. |