

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM  
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG  
KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO CUỐI KỲ  
MÔN HỌC MẪU THIẾT KẾ**

# **XÂY DỰNG ỨNG DỤNG QUẢN LÝ VÀ TÍNH TIỀN NƯỚC**

*Người hướng dẫn:* **ThS. VŨ ĐÌNH HỒNG**

*Người thực hiện:* **HUỲNH MINH HẢI – 51800373**

**NGUYỄN TẤN TÀI – 51800112**

**NGUYỄN QUỐC BẢO – 51800011**

**NGUYỄN HỮU CẢNH – 51702066**

**Lớp: 17050202 – 18050201**

**Khoá: 21 – 22**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022**

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM  
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG  
KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO CUỐI KỲ  
MÔN HỌC MẪU THIẾT KẾ**

# **XÂY DỰNG ỨNG DỤNG QUẢN LÝ VÀ TÍNH TIỀN NƯỚC**

*Người hướng dẫn:* **ThS. VŨ ĐÌNH HỒNG**

*Người thực hiện:* **HUỲNH MINH HẢI – 51800373**

**NGUYỄN TẤN TÀI – 51800112**

**NGUYỄN QUỐC BẢO – 51800011**

**NGUYỄN HỮU CẢNH – 51702066**

**Lớp: 17050202 – 18050201**

**Khoá: 21 – 22**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022**

## LỜI CẢM ƠN

Chúng em xin chân thành cảm ơn **ThS. Vũ Đình Hồng** – giảng viên lý thuyết và thực hành bộ môn *Mẫu thiết kế (504077)* đã nhiệt tình giúp đỡ cũng như giải đáp những thắc mắc của chúng em trong suốt quá trình chúng em thực hiện bài báo cáo cuối kỳ này. Nhờ có sự hướng dẫn, gợi ý tận tình của Thầy về những kiến thức, tài liệu và các kỹ năng liên quan mà chúng em mới hoàn thành được bài báo cáo cuối kỳ với đề tài “*Xây dựng ứng dụng quản lý và tính tiền nước*”.

Dĩ nhiên, với khả năng và kiến thức còn hạn hẹp, bài báo cáo của chúng em vẫn còn nhiều hạn chế và thiếu sót. Do vậy, chúng em mong rằng sẽ nhận được những lời góp ý và đánh giá từ Thầy để bài báo cáo cuối kỳ của chúng em được hoàn thiện hơn.

Một lần nữa, chúng em xin chân thành cảm ơn Thầy!

## **BÁO CÁO ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Chúng tôi xin cam đoan đây là báo cáo của riêng chúng tôi và được sự hướng dẫn của ThS. Vũ Đình Hồng. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa từng công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính chúng tôi thu thập từ các nguồn khác nhau và có ghi rõ trong phần tài liệu tham khảo.

**Nếu phát hiện có bất kỳ sự gian lận nào chúng tôi xin hoàn toàn chịu trách nhiệm về nội dung báo cáo của mình.** Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do chúng tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày 06 tháng 01 năm 2022*

*Tác giả*

*(ký và ghi rõ họ tên)*

*Huỳnh Minh Hải*

*Nguyễn Tấn Tài*

*Nguyễn Quốc Bảo*

*Nguyễn Hữu Cảnh*

## **PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN**

### **Phần xác nhận của GV hướng dẫn**

---

---

---

---

---

---

---

Tp. Hồ Chí Minh, ngày      tháng      năm  
(ký và ghi rõ họ tên)

### **Phần đánh giá của GV chấm bài**

---

---

---

---

---

---

---

Tp. Hồ Chí Minh, ngày      tháng      năm  
(ký và ghi rõ họ tên)

## **TÓM TẮT**

Trong CHƯƠNG 1 – MÔ TẢ ĐỀ TÀI sẽ trình bày lý do xây dựng và các chức năng của ứng dụng. Các pattern mà nhóm đã thực hiện trong ứng dụng sẽ được trình bày trong CHƯƠNG 2 – ÁP DỤNG PATTERN.

## MỤC LỤC

LỜI CẢM ƠN .....	i
BÁO CÁO ĐƯỢC HOÀN THÀNH.....	ii
PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN .....	iii
TÓM TẮT .....	iv
MỤC LỤC.....	1
DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ .....	2
CHƯƠNG 1 – MÔ TẢ ĐỀ TÀI.....	4
1.1 Lý do xây dựng ứng dụng .....	4
1.2 Các chức năng của ứng dụng .....	4
1.2.1 Đối với khách hàng .....	4
1.2.2 Đối với công ty sử dụng ứng dụng .....	9
CHƯƠNG 2 – ÁP DỤNG PATTERN.....	15
2.1 Singleton Pattern .....	15
2.2 Strategy Pattern .....	16
2.3 Template Method Pattern.....	18
2.4 Factory Method Pattern.....	20
2.5 Command Pattern.....	22
2.6 Adapter Pattern .....	25
2.7 Observer Pattern.....	27
TÀI LIỆU THAM KHẢO.....	30

## DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

### DANH MỤC HÌNH

Hình 1.1 Giao diện Đăng ký .....	5
Hình 1.2 Giao diện Đăng nhập.....	5
Hình 1.3 Chức năng lắp đặt thiết bị (1).....	6
Hình 1.4 Chức năng lắp đặt thiết bị (2).....	6
Hình 1.5 Chức năng ON/OFF thiết bị .....	7
Hình 1.6 Remove thiết bị .....	7
Hình 1.7 Chức năng xem tiền trước .....	8
Hình 1.8 Chức năng chuyển đơn vị tiền.....	8
Hình 1.9 Giao diện Đăng nhập.....	9
Hình 1.10 Giao diện Quản lý .....	9
Hình 1.11 Tìm kiếm bằng số điện thoại và họ tên khách hàng.....	10
Hình 1.12 Button chuyển giao diện Đăng ký đồng hồ nước.....	10
Hình 1.13 Tạo đồng hồ nước .....	11
Hình 1.14 Xóa đồng hồ nước .....	11
Hình 1.15 Xóa khách hàng.....	12
Hình 1.16 Hủy đồng hồ nước của khách hàng.....	13
Hình 1.17 Đăng ký đồng hồ nước .....	13
Hình 1.18 Button thực hiện Tính tiền nước .....	14
Hình 1.19 Thông tin Tính tiền nước .....	14
Hình 2.1 Sơ đồ lớp Singleton Pattern .....	15
Hình 2.2 Code áp dụng Singleton Pattern.....	15
Hình 2.3 Sơ đồ lớp Strategy Pattern .....	16
Hình 2.4 Code áp dụng Strategy Pattern (1) .....	16
Hình 2.5 Code áp dụng Strategy Pattern (2) .....	17
Hình 2.6 Code áp dụng Strategy Pattern (3) .....	17
Hình 2.7 Code áp dụng Strategy Pattern (4) .....	17
Hình 2.8 Sơ đồ lớp Template Method Pattern .....	18



Hình 2.9 Code áp dụng Template Method Pattern (1).....	19
Hình 2.10 Code áp dụng Template Method Pattern (2).....	19
Hình 2.11 Code áp dụng Template Method Pattern (3).....	20
Hình 2.12 Code áp dụng Template Method Pattern (4).....	20
Hình 2.13 Code áp dụng Template Method Pattern (5).....	20
Hình 2.14 Sơ đồ lớp Factory Method Pattern .....	21
Hình 2.15 Code áp dụng Factory Method Pattern (1).....	21
Hình 2.16 Code áp dụng Factory Method Pattern (2).....	22
Hình 2.17 Code áp dụng Factory Method Pattern (3).....	22
Hình 2.18 Code áp dụng Factory Method Pattern (4).....	22
Hình 2.19 Sơ đồ lớp Command Pattern .....	23
Hình 2.20 Code áp dụng Command Pattern (1).....	23
Hình 2.21 Code áp dụng Command Pattern (2).....	24
Hình 2.22 Code áp dụng Command Pattern (3).....	24
Hình 2.23 Code áp dụng Command Pattern (4).....	25
Hình 2.24 Sơ đồ lớp Adapter Pattern .....	25
Hình 2.25 Code áp dụng Adapter Pattern (1).....	26
Hình 2.26 Code áp dụng Adapter Pattern (2).....	26
Hình 2.27 Code áp dụng Adapter Pattern (3).....	26
Hình 2.28 Code áp dụng Adapter Pattern (4).....	26
Hình 2.29 Sơ đồ lớp Observer Pattern .....	27
Hình 2.30 Code áp dụng Observer Pattern (1).....	27
Hình 2.31 Code áp dụng Observer Pattern (2).....	28
Hình 2.32 Code áp dụng Observer Pattern (3).....	28
Hình 2.33 Code áp dụng Observer Pattern (4).....	29
Hình 2.34 Code áp dụng Observer Pattern (5).....	29

# CHƯƠNG 1 – MÔ TẢ ĐỀ TÀI

## 1.1 Lý do xây dựng ứng dụng


Hiện nay, có rất nhiều ứng dụng trên thị trường nhằm cung cấp cho khách hàng cũng như các công ty cung cấp dịch vụ để quản lý và sử dụng như điện, nước, internet,... Đa số các ứng dụng đều có thể tra cứu số lượng, khối lượng dịch vụ tiêu thụ từ đó khách hàng sẽ xem được chi phí phải trả hàng tháng. Tuy nhiên, chưa có ứng dụng nào tích hợp việc điều khiển các thiết bị đi kèm với dịch vụ từ nhà cung cấp (ví dụ: ứng dụng quản lý điện có thể bật/tắt đèn, quạt,... trong nhà).

Vì vậy, nhóm chúng em quyết định xây dựng một ứng dụng có thể thực hiện tích hợp việc điều khiển các thiết bị (vòi sen, vòi nước thường,...) cũng như quản lý và tính tiền nước. Với đề tài “*Xây dựng ứng dụng quản lý và tính tiền nước*”, chúng em sẽ áp dụng vào đó các mẫu thiết kế đã học nhằm tối ưu mã nguồn và dễ dàng cho công việc thêm/bớt các chức năng, thiết bị cũng như bảo trì về sau.

## 1.2 Các chức năng của ứng dụng

### 1.2.1 Đối với khách hàng

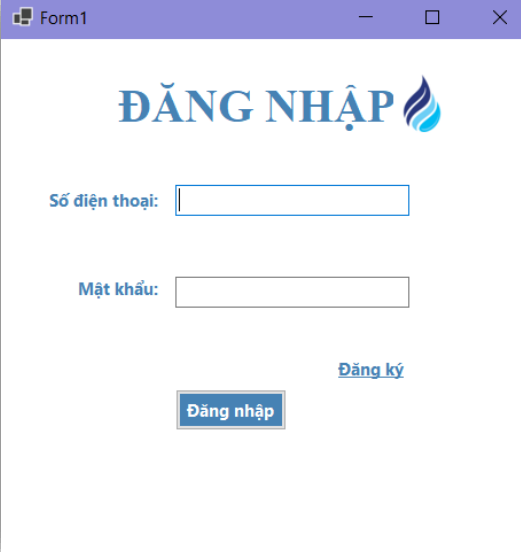
- Đăng ký: Những khách hàng chưa có tài khoản sẽ đăng ký để có thể đăng nhập vào ứng dụng và thực hiện các chức năng khác.



The screenshot shows a window titled "Register" with a blue header bar. The main content area has a white background. At the top, the text "ĐĂNG KÝ" is displayed in blue, followed by a blue flame icon. Below this, there are five input fields: "Họ và tên:" (Name and surname), "Số điện thoại:" (Phone number), "Mật khẩu:" (Password), "Địa chỉ đăng ký:" (Registration address), and "Loại hình:" (Account type) which is a dropdown menu. At the bottom right, there is a blue button labeled "Đăng ký" (Register) and a blue link labeled "Đăng nhập" (Login).

Hình 1.1 Giao diện Đăng ký

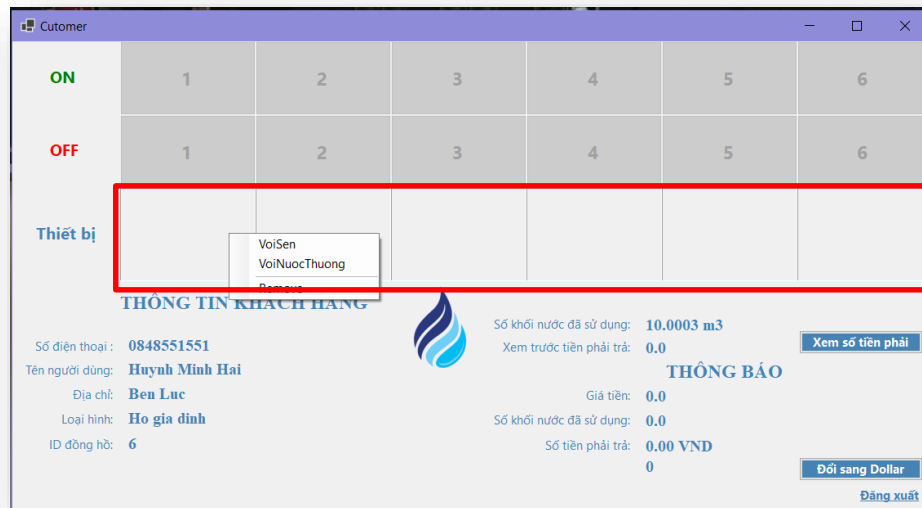
- Đăng nhập: Sau khi đã đăng ký hoặc khách hàng đã có tài khoản, khách hàng có thể đăng nhập vào ứng dụng.



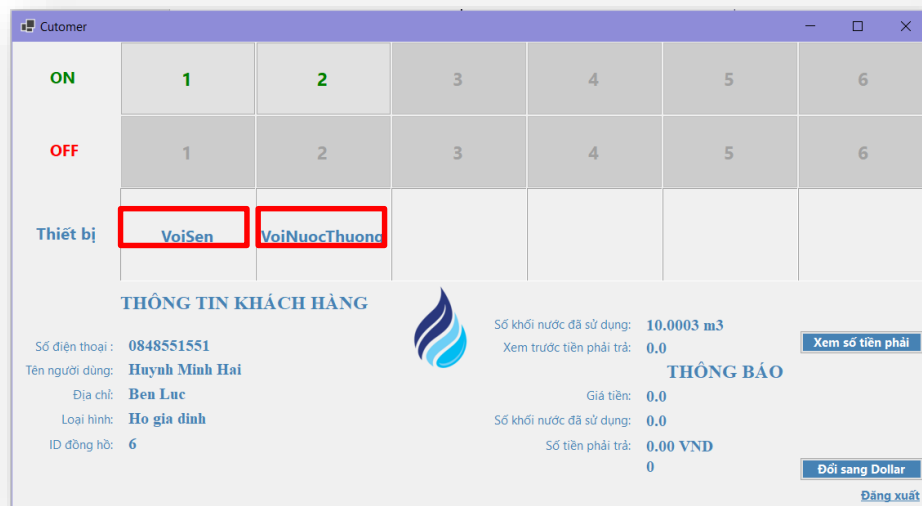
The screenshot shows a window titled "Form1" with a blue header bar. The main content area has a white background. At the top, the text "ĐĂNG NHẬP" is displayed in blue, followed by a blue flame icon. Below this, there are two input fields: "Số điện thoại:" (Phone number) and "Mật khẩu:" (Password). At the bottom right, there is a blue button labeled "Đăng nhập" (Login) and a blue link labeled "Đăng ký" (Register).

Hình 1.2 Giao diện Đăng nhập

- Sử dụng nước: Khách hàng có thể sử dụng nước do công ty cung cấp bằng các thiết bị sử dụng nước như vòi sen, vòi nước thường,... Khách hàng có thể lắp đặt thiết bị bằng cách nhấn chuột phải vào ô thiết bị và chọn thiết bị sử dụng nước.



Hình 1.3 Chức năng lắp đặt thiết bị (1)



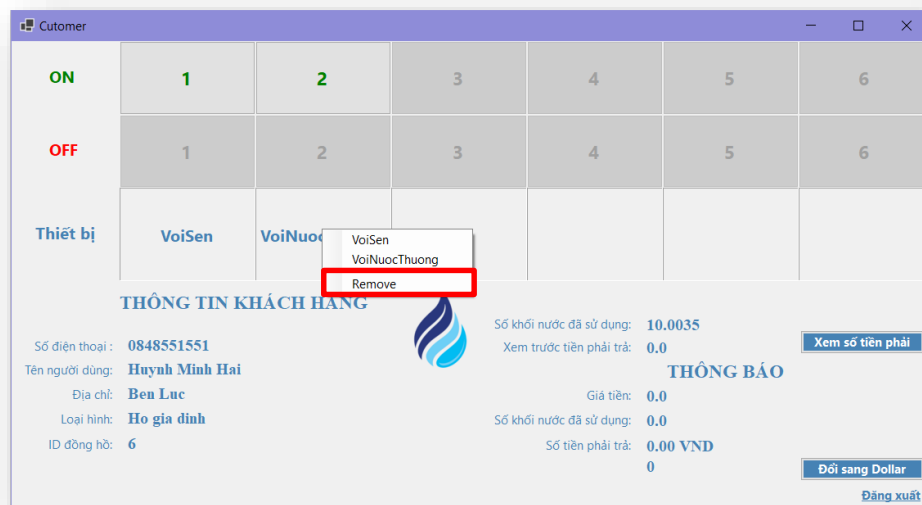
Hình 1.4 Chức năng lắp đặt thiết bị (2)

Sau khi chọn thiết bị sử dụng nước và lắp đặt. Khách hàng có thể sử dụng thiết bị bằng cách bấm vào nút ON tương ứng với mỗi thiết bị hoặc khách hàng có thể bấm vào nút OFF để ngừng sử dụng thiết bị.



Hình 1.5 Chức năng ON/OFF thiết bị

Khách hàng có thể gỡ cài đặt thiết bị sử dụng nước bằng cách nhấn chuột phải vào thiết bị đã lắp đặt và nhấn vào Remove.



Hình 1.6 Remove thiết bị

Xem số tiền phải trả: Khách hàng có thể xem trước số tiền mà mình phải trả bằng cách nhấn vào nút “Xem số tiền phải trả”. Số tiền phải trả sẽ dựa trên số khối nước mà khách hàng đã sử dụng.

The screenshot shows a web application window titled 'Cutomer'. It features a control panel with 'ON' and 'OFF' buttons and a 'Thiết bị' section. Below this is a 'THÔNG TIN KHÁCH HÀNG' section with customer details. To the right, a 'THÔNG BÁO' section displays usage statistics. A red box highlights the 'Xem số tiền phải trả' button next to the value '70.545 VND'.

ON	1	2	3	4	5	6
OFF	1	2	3	4	5	6
Thiết bị						

**THÔNG TIN KHÁCH HÀNG**

Số điện thoại: 0848551551  
 Tên người dùng: Huỳnh Minh Hai  
 Địa chỉ: Ben Luc  
 Loại hình: Ho gia đình  
 ID đồng hồ: 6

**THÔNG BÁO**

Số khối nước đã sử dụng: 10.0035 m<sup>3</sup>  
 Xem trước tiền phải trả: 70.545 VND  
 Giá tiền: 0,0  
 Số khối nước đã sử dụng: 0,0  
 Số tiền phải trả: 0,00 VND  
 0

[Xem số tiền phải trả](#)  
[Đổi sang Dollar](#)  
[Đăng xuất](#)

Hình 1.7 Chức năng xem tiền trước

Đổi tiền phải trả từ đơn vị VND sang đơn vị Dollar bằng cách nhấn vào nút “Đổi sang Dollar”.

This screenshot shows the same web application after the currency conversion. The 'Xem trước tiền phải trả' value is now '0,0'. The 'Giá tiền' is updated to '5,973 VND/m<sup>3</sup>'. The 'Số tiền phải trả' is now '3,21 \$', which is highlighted by a red box. The 'Đổi sang Dollar' button is also highlighted.

ON	1	2	3	4	5	6
OFF	1	2	3	4	5	6
Thiết bị						

**THÔNG TIN KHÁCH HÀNG**

Số điện thoại: 0848551551  
 Tên người dùng: Huỳnh Minh Hai  
 Địa chỉ: Ben Luc  
 Loại hình: Ho gia đình  
 ID đồng hồ: 6

**THÔNG BÁO**

Số khối nước đã sử dụng: 10.0035 m<sup>3</sup>  
 Xem trước tiền phải trả: 0,0  
 Giá tiền: 5,973 VND/m<sup>3</sup>  
 Số khối nước đã sử dụng: 10.0035 m<sup>3</sup>  
 Số tiền phải trả: 70,5 VND  
 3,21 \$

[Xem số tiền phải trả](#)  
[Đổi sang Dollar](#)  
[Đăng xuất](#)

Hình 1.8 Chức năng chuyển đơn vị tiền

### 1.2.2 Đối với công ty sử dụng ứng dụng

- Đăng nhập: Admin của công ty có thể đăng nhập vào ứng dụng.

The screenshot shows a web form titled "ĐĂNG NHẬP" (Login) with a blue flame logo. It contains two input fields: "Số điện thoại:" (Phone number) and "Mật khẩu:" (Password). Below the password field is a "Đăng ký" (Register) link and a "Đăng nhập" (Login) button.

Hình 1.9 Giao diện Đăng nhập

Sau khi đăng nhập thành công, chỉ có những khách hàng đã đăng ký đồng hồ nước mới hiển thị trên giao diện chính.

The screenshot shows a web interface titled "QUẢN LÝ" (Management) with a blue flame logo. It includes search fields for "Số điện thoại khách hàng:" (Customer phone number) and "Tên khách hàng:" (Customer name), each with a "Tìm" (Search) button. Below the search fields is a table with the following data:

customer_phone	customer_name	customer_address	customer_kind	cubicmetre_id	cubicmetre_count
0848551551	Huynh Minh Hai	Ben Luc	Ho gia dinh	6	10.0035
0848551552	Nguyen Quoc Bao	Sai Gon	Ho ngheo	1003	11
0848551553	Nguyen Tan Tai	Sai Gon	Doanh nghiep	1006	8

Hình 1.10 Giao diện Quản lý

- Tìm khách hàng: Admin có thể tìm thông tin của khách hàng thông qua họ tên và số điện thoại.

The screenshot shows a web application window titled 'Manager' with a 'QUẢN LÝ' (Management) header. On the left is a 'Menu' button. On the right are 'Đăng xuất' (Logout) and 'Tính tiền nước' (Calculate Water Bill) buttons. Below the header, there are two search filters: 'Số điện thoại khách hàng:' (Customer Phone Number) with a text input containing '0848551551' and a 'Tìm' (Search) button, and 'Tên khách hàng:' (Customer Name) with an empty text input and a 'Tìm' (Search) button. Below these filters is a table with the following data:

customer_phone	customer_name	customer_address	customer_kind	cubicmetre_id	cubicmetre_count
0848551551	Huỳnh Minh Hai	Ben Luc	Họ gia đình	6	10.0035

Hình 1.11 Tìm kiếm bằng số điện thoại và họ tên khách hàng

- Đăng ký đồng hồ nước: Admin có thể đăng ký đồng hồ nước cho khách hàng bằng cách chọn Menu trên thanh công cụ, chọn “Đăng ký đồng hồ nước”. Sau đó, giao diện đăng ký đồng hồ nước sẽ được hiển thị.

The screenshot shows the same 'QUẢN LÝ' (Management) interface. The 'Menu' button on the left is highlighted with a red box, and a dropdown menu is visible showing the option 'Đăng ký đồng hồ nước' (Register Water Meter). The search filters and the table of customers remain the same as in the previous screenshot.

Hình 1.12 Button chuyển giao diện Đăng ký đồng hồ nước

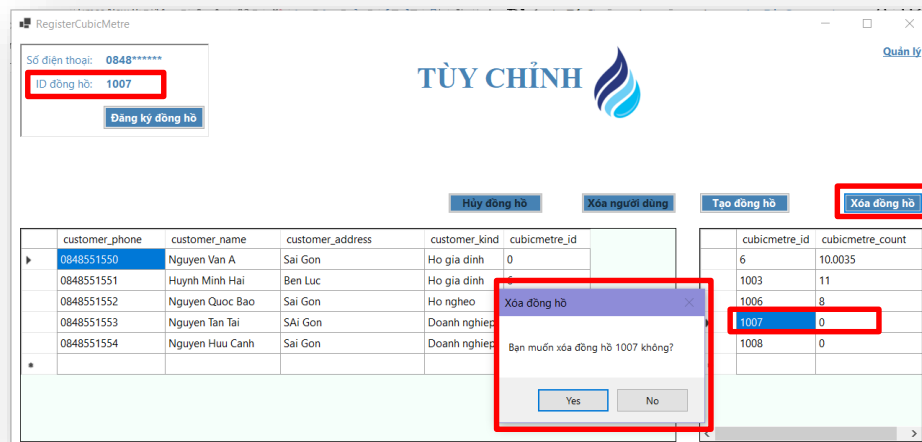


- Tạo đồng hồ nước: Admin tạo đồng hồ nước bằng cách nhấn vào nút “Tạo đồng hồ” trên giao diện.



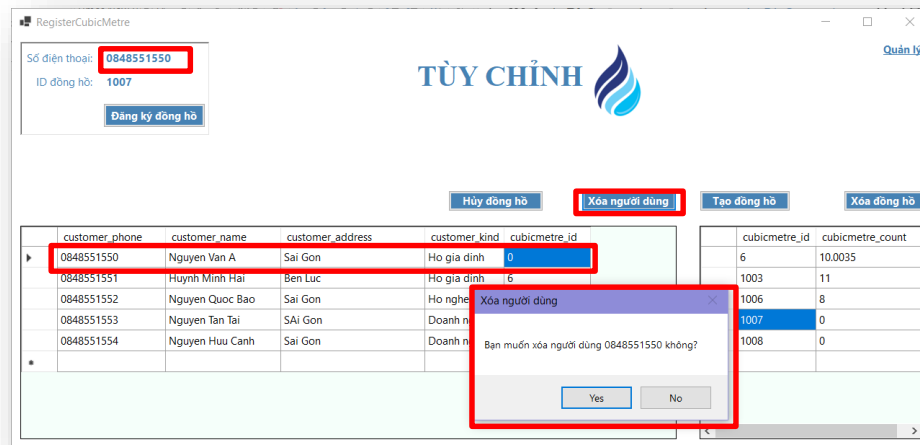
Hình 1.13 Tạo đồng hồ nước

- Xóa đồng hồ: Admin có thể xóa đồng hồ nước bằng cách chọn vào ô đồng hồ nước mà admin muốn xóa và nhấn vào nút “Xóa đồng hồ”. Thông báo xác nhận sẽ hiện lên giao diện để xác nhận việc xóa đồng hồ. Ngoài ra, admin có thể xem ID đồng hồ mà admin đã chọn ở góc trên bên trái màn hình.



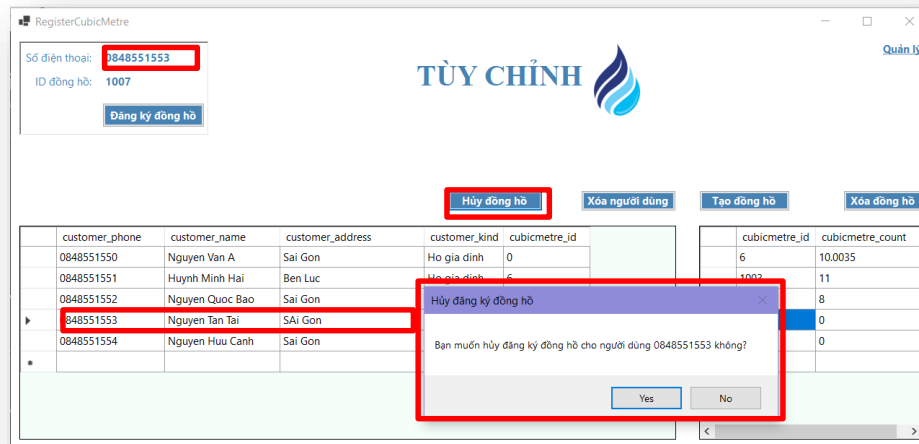
Hình 1.14 Xóa đồng hồ nước

- Xóa khách hàng: Admin xóa khách hàng bằng cách chọn vào ô khách hàng mà admin muốn xóa và nhấn nút “Xóa người dùng”. Thông báo xác nhận sẽ hiện lên giao diện để xác nhận việc xóa khách hàng. Ngoài ra, admin có thể xem số điện thoại mà admin đã chọn ở góc trên bên trái màn hình.



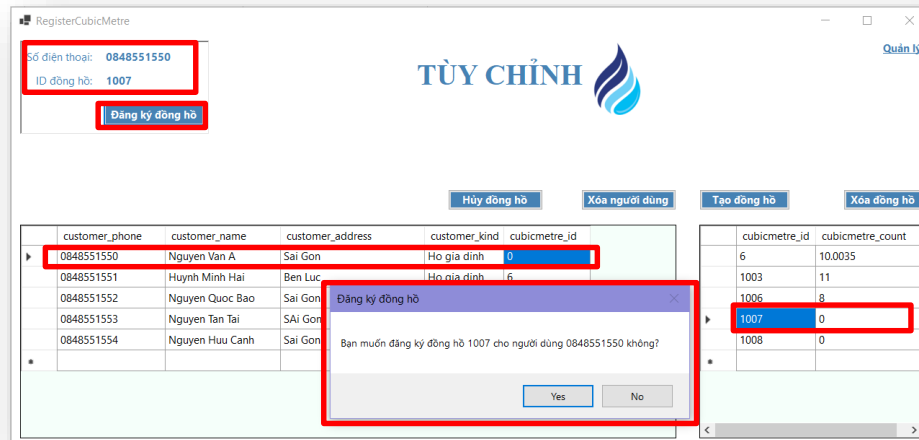
Hình 1.15 Xóa khách hàng

- Hủy đồng hồ: Admin hủy đăng ký đồng hồ của khách hàng bằng cách chọn ô khách hàng mà người dùng muốn hủy đăng ký đồng hồ và nhấn “Hủy đồng hồ”. Thông báo xác nhận hủy đồng hồ sẽ được hiển thị lên giao diện để các nhận việc hủy. Ngoài ra, admin có thể xem ô người dùng mà mình đã chọn ở góc trên bên trái màn hình.



Hình 1.16 Hủy đồng hồ nước của khách hàng

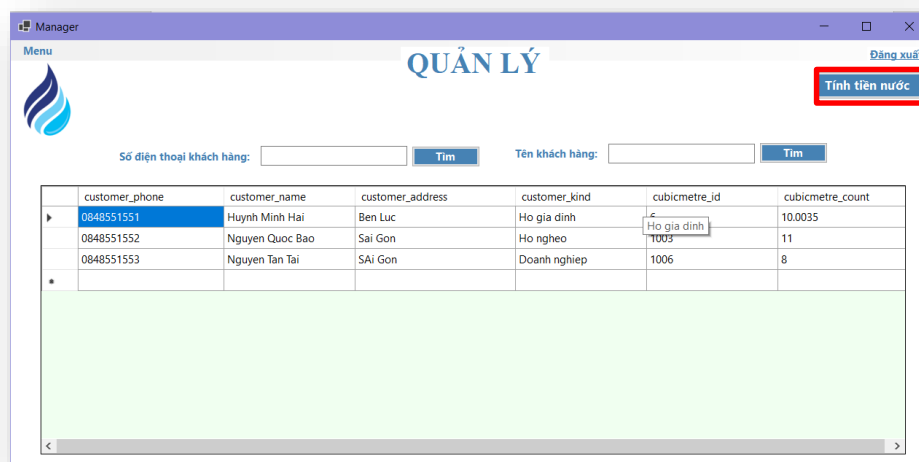
- Đăng ký đồng hồ: Admin đăng ký đồng hồ cho khách hàng bằng cách chọn ô khách hàng đăng ký đồng hồ và ô đồng hồ mà admin muốn đăng ký cho khách hàng. Thông tin sẽ được hiển thị ở góc trên bên trái màn hình. Sau đó, người sẽ nhấn vào “Đăng ký đồng hồ”. Thông báo xác nhận cũng sẽ được hiển thị.



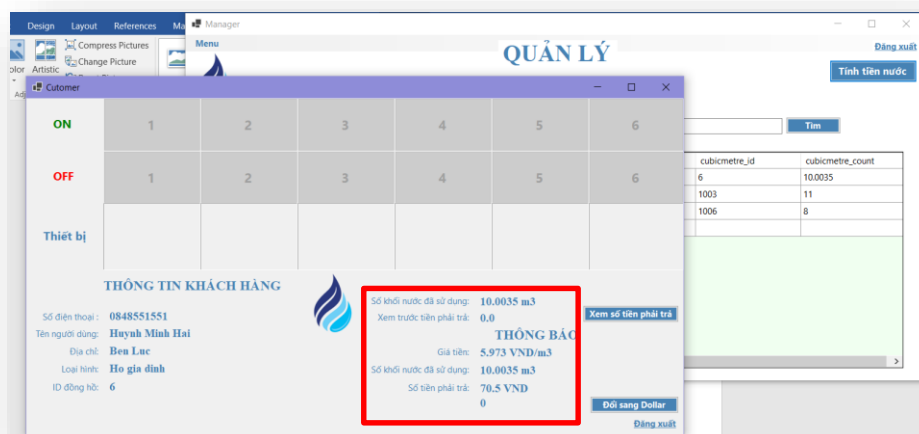
Hình 1.17 Đăng ký đồng hồ nước

- Tính tiền nước: Cuối mỗi tháng Công ty sẽ tính tiền nước đã sử dụng của mỗi khách hàng và thông báo về giao diện cho mỗi khách hàng sử dụng bằng cách nhấn vào

nút “Tính tiền nước” phía trên bên phải màn hình giao diện chính. Tiền nước sẽ dựa trên số khối nước mà khách hàng đó đã sử dụng.



Hình 1.18 Button thực hiện Tính tiền nước

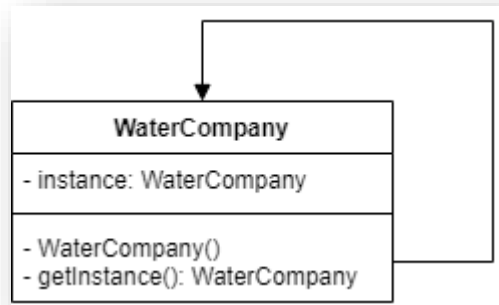


Hình 1.19 Thông tin Tính tiền nước

## CHƯƠNG 2 – ÁP DỤNG PATTERN

### 2.1 Singleton Pattern

Lý do áp dụng: Ứng dụng chỉ được một Công ty quản lý và cung cấp nước đến các khách hàng đăng ký, vậy nên ở đây chúng ta sẽ áp dụng Singleton Pattern cho WaterCompany để nó là duy nhất.



Hình 2.1 Sơ đồ lớp Singleton Pattern

```

10 references
public class WaterCompany
{
    //Thông tin công ty nước
    public string Company_name;
    public string Company_address;
    public string Company_phone;

    // Giá nước
    public float Water_HogiadinhCost = 5.973f;
    public float Water_HongheoCost = 3.6f;
    public float Water_DoanhnghiepCost = 9.955f;

    //Ham get set
    0 references
    public string company_name { get; set; }
    0 references
    public string company_address { get; set; }
    0 references
    public string company_phone { get; set; }

    //Áp dụng singleton(Vì app này sử dụng duy nhất cho 1 công ty)
    private static WaterCompany instance;

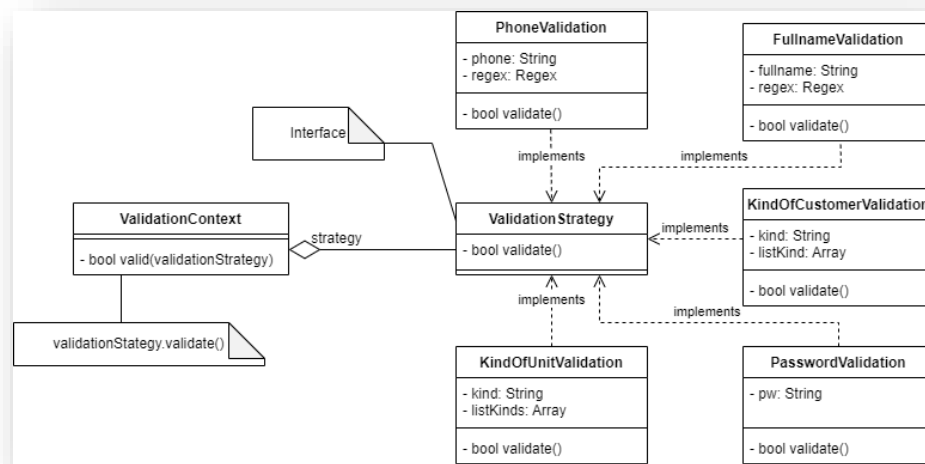
    1 reference
    private WaterCompany() { }

    6 references
    public static WaterCompany getInstance()
    {
        if (instance == null)
        {
            instance = new WaterCompany();
        }
        return instance;
    }
}
  
```

Hình 2.2 Code áp dụng Singleton Pattern

## 2.2 Strategy Pattern

Lý do áp dụng: Sử dụng Strategy Pattern để kiểm tra tính hợp lệ khi điền form đăng ký hoặc điền form đăng nhập. Mỗi ô input sẽ có hành vi kiểm tra hợp lệ khác nhau. Nếu như chúng ta muốn thêm một ô input và cần thêm hành vi kiểm tra hợp lệ cho nó, chúng ta chỉ cần thêm một class mới và implement từ class interface ValidationStrategy. Sử dụng Strategy Pattern trong trường hợp này giúp chúng ta dễ dàng mở rộng và kết hợp các hành vi mới mà không cần phải thay đổi ứng dụng.



Hình 2.3 Sơ đồ lớp Strategy Pattern

```

6 references
public interface ValidationStrategy
{
    9 references
    bool validate();
}
  
```

Hình 2.4 Code áp dụng Strategy Pattern (1)

```

class PhoneValidation: ValidationStrategy
{
    private string phone;
    private Regex regex = new Regex(@"^0\d{9,10}$");

    4 references
    public PhoneValidation(string phone)
    {
        this.phone = phone;
    }

    4 references
    public bool validate()
    {
        return regex.IsMatch(phone);
    }
}

//Class ConcreteStrategy
2 references
class FullnameValidation : ValidationStrategy
{
    private string fullname;
    private Regex regex = new Regex(@"^[a-zA-Z]{4,}(?:[a-zA-Z]*)(0,2)$");

    1 reference
    public FullnameValidation(string fullname)
    {
        this.fullname = fullname;
    }

    2 references
    public bool validate()
    {
        return regex.IsMatch(fullname);
    }
}

```

Hình 2.5 Code áp dụng Strategy Pattern (2)

```

class KindOfCustomerValidation : ValidationStrategy
{
    private string kind;
    private string[] listKinds = { "Ho gia dinh", "Doanh nghiep", "Ho ngheo" };

    2 references
    public KindOfCustomerValidation(string kind)
    {
        this.kind = kind;
    }

    2 references
    public bool validate()
    {
        int index = Array.IndexOf(listKinds, kind);
        if (index > -1)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
}

//Class ConcreteStrategy
2 references
class PasswordValidation : ValidationStrategy
{
    private string pw;

    1 reference
    public PasswordValidation(string pw)
    {
        this.pw = pw;
    }
}

```

Hình 2.6 Code áp dụng Strategy Pattern (3)

```

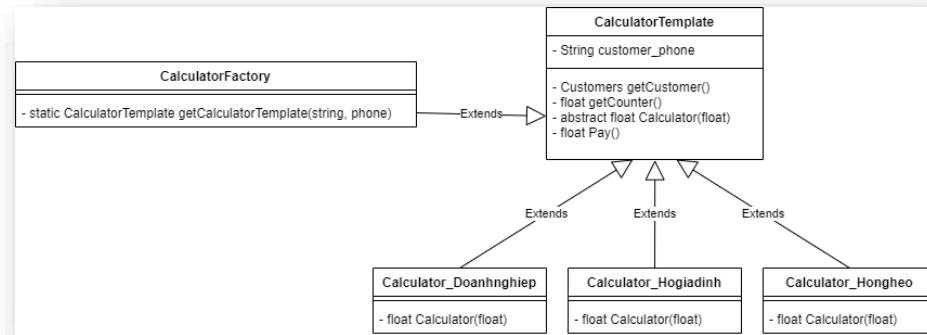
//Class strategy context
2 references
class ValidationContext
{
    6 references
    public bool valid(ValidationStrategy validationStrategy)
    {
        return validationStrategy.validate();
    }
}

```

Hình 2.7 Code áp dụng Strategy Pattern (4)

## 2.3 Template Method Pattern

Lý do áp dụng: Mỗi tháng, Công ty cấp nước cần tính tiền cho từng khách hàng với các loại hình đăng ký khác nhau như: Hộ gia đình, Hộ nghèo, Doanh nghiệp. Các bước tính tiền nước ở mỗi loại khách hàng là giống nhau như: Lấy thông tin khách hàng, lấy số khối nước mà khách hàng đã sử dụng và tính tiền theo loại khách hàng. Tuy nhiên, ở bước tính tiền theo loại khách hàng thì mỗi loại khách hàng sẽ có một cách tính khác nhau. Vì vậy, chúng ta sẽ áp dụng Template Method Pattern cho trường hợp này. Áp dụng Template Method sẽ giúp chúng ta tái sử dụng code, tránh trùng lặp code và đưa những phần trùng lặp vào lớp cha (class abstract CalculatorTemplate).



Hình 2.8 Sơ đồ lớp Template Method Pattern



```

abstract class CalculatorTemplate
{
    //Khai báo biến
    public string customer_phone;

    //Hàm get set
    2 references
    public Customers Customers { get; set; }
    3 references
    protected CalculatorTemplate(string phone)
    {
        this.customer_phone = phone;
    }

    //Lay nguoi dung
    1 reference
    protected Customers getCustomer()
    {
        CustomerSql customerSql = new CustomerSql();
        return customerSql.findCustomer(customer_phone);
    }

    //Lay so ky dien ma khach da su dung
    1 reference
    protected float getCounter()
    {
        CubicMetreSql cubicMetreSql = new CubicMetreSql();
        CubicMetre cubicMetre = cubicMetreSql.findCubicMetre(Customers.CubicMetre_ID);
        float counter = cubicMetre.CubicMetre_count;
        return counter;
    }

    //Hàm abstract
    4 references
    protected abstract float Calculator(float counter);

    //Hàm thực hiện tính tiền
    2 references
    public float Pay()
    {
        this.Customers = getCustomer();
        float pay = Calculator(getCounter());
        return pay;
    }
}

```

Hình 2.9 Code áp dụng Template Method Pattern (1)

```

class Calculator_Hongheo : CalculatorTemplate
{
    1 reference
    public Calculator_Hongheo(string phone) : base(phone){ }
    2 references
    protected override float Calculator(float counter)
    {
        float cost = WaterCompany.GetInstance().Water_HongheoCost;

        if (counter <= 10)
        {
            //cost 3.600f
            return counter * cost;
        }
        else if (counter > 10 && counter <= 20)
        {
            return counter * (cost + 0.9f);
        }
        else if (counter > 20 && counter <= 30)
        {
            return counter * (cost + 2.0f);
        }
        else
        {
            return counter * (cost + 3.1f);
        }
    }
}

```

Hình 2.10 Code áp dụng Template Method Pattern (2)

```

class Calculator_Hogiadinh : CalculatorTemplate
{
    1 reference
    public Calculator_Hogiadinh(string phone) : base(phone){ }

    2 references
    protected override float Calculator(float counter)
    {
        float cost = WaterCompany.GetInstance().Water_HogiadinhCost;

        if (counter <= 10)
        {
            //cost 5.973f
            return counter * cost;
        }
        else if (counter > 10 && counter <= 20)
        {
            return counter * (cost + 1.079f);
        }
        else if (counter > 20 && counter <= 30)
        {
            return counter * (cost + 2.696f);
        }
        else
        {
            return counter * (cost + 9.956f);
        }
    }
}

```

Hình 2.11 Code áp dụng Template Method Pattern (3)

```

class Calculator_Doanhnghiep: CalculatorTemplate
{
    1 reference
    public Calculator_Doanhnghiep(string phone) : base(phone){ }

    2 references
    protected override float Calculator(float counter)
    {
        float cost = WaterCompany.GetInstance().Water_DoanhnghiepCost;
        //cost 9.955f
        return counter * cost;
    }
}

```

Hình 2.12 Code áp dụng Template Method Pattern (4)

```

class CalculatorFactory
{
    1 reference
    public CalculatorFactory(){ }

    2 references
    public static CalculatorTemplate getCalculatorTemplate(string kind, string phone)
    {
        if (kind == "Ho gia dinh")
        {
            return new Calculator_Hogiadinh(phone);
        }
        else if (kind == "Ho ngheo")
        {
            return new Calculator_Hongheo(phone);
        }
        else
        {
            return new Calculator_Doanhnghiep(phone);
        }
    }
}

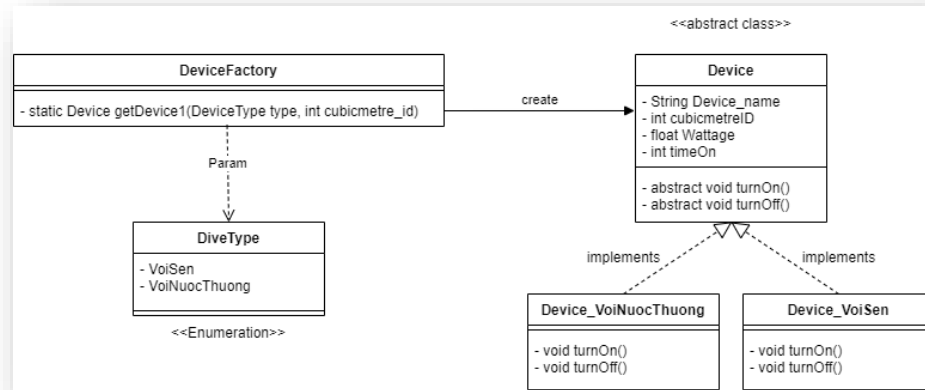
```

Hình 2.13 Code áp dụng Template Method Pattern (5)

## 2.4 Factory Method Pattern

Lý do áp dụng: Để thuận tiện cho việc khởi tạo các thiết bị sử dụng nước khác nhau và được khởi tạo tùy vào từng khách hàng. Vì vậy, chúng ta sẽ áp dụng Factory Method Pattern. Nếu như khách hàng muốn tạo một thiết bị sử dụng nước thì khách hàng chỉ cần truyền tên thiết bị cần sử dụng và thiết bị đó sẽ được tạo. Áp dụng Factory Method Pattern

giúp chúng ta đưa trách nhiệm của việc khởi tạo một lớp từ phía người dùng (client) sang lớp Factory.



Hình 2.14 Sơ đồ lớp Factory Method Pattern

```

public abstract class Device
{
    DeviceSql deviceSql = new DeviceSql();

    //Thông tin thiết bị sử dụng nước
    public string Device_name;
    public int cubicmetreID;
    //công suất
    public float Wattage;

    //Thời gian sử dụng nước
    public int timeOn = 0;
    public System.Timers.Timer timeout;
    private InterVal interVal = new InterVal();

    //Hàm get set
    0 references
    public string device_name { get; set; }
    0 references
    public int cubicmetreid { get; set; }
    0 references
    public float wattage { get; set; }
    0 references
    public int timeon { get; set; }

    //-----
    0 references
    public Device()
    {
    }

    2 references
    public Device(string Device_name, float Wattage, int cubicmetreID)
    {
        this.Device_name = Device_name;
        this.Wattage = Wattage;
        this.cubicmetreID = cubicmetreID;
    }

    //Hàm on off thiết bị
    4 references
    public abstract void turnOn();
    4 references
    public abstract void turnOff();
}
  
```

Hình 2.15 Code áp dụng Factory Method Pattern (1)

```
//AP Factory Pattern
//SubClass
//reference
public class Device_VoiNuocThuong: Device
{
    1 reference
    public Device_VoiNuocThuong(int cubicMetreID) : base("VoiNuocThuong", 0.14f, cubicMetreID)
    {
    }

    2 references
    public override void turnOff()
    {
        this.stop();
    }

    2 references
    public override void turnOn()
    {
        this.use();
    }
}
```

Hình 2.16 Code áp dụng Factory Method Pattern (2)

```
//AP Factory Pattern
//Subclass
public class Device_VoiSen : Device
{
    public Device_VoiSen(int cubicMetreID): base("VoiSen", 0.16f, cubicMetreID)
    {
    }

    public override void turnOff()
    {
        this.stop();
    }

    2 references
    public override void turnOn()
    {
        this.use();
    }
}
```

Hình 2.17 Code áp dụng Factory Method Pattern (3)

```
//Ap factory pattern
//Factory class
//reference
public class DeviceFactory
{
    6 references
    public enum DeviceType
    {
        VoiSen,
        VoiNuocThuong
    }

    0 references
    private DeviceFactory(){ }

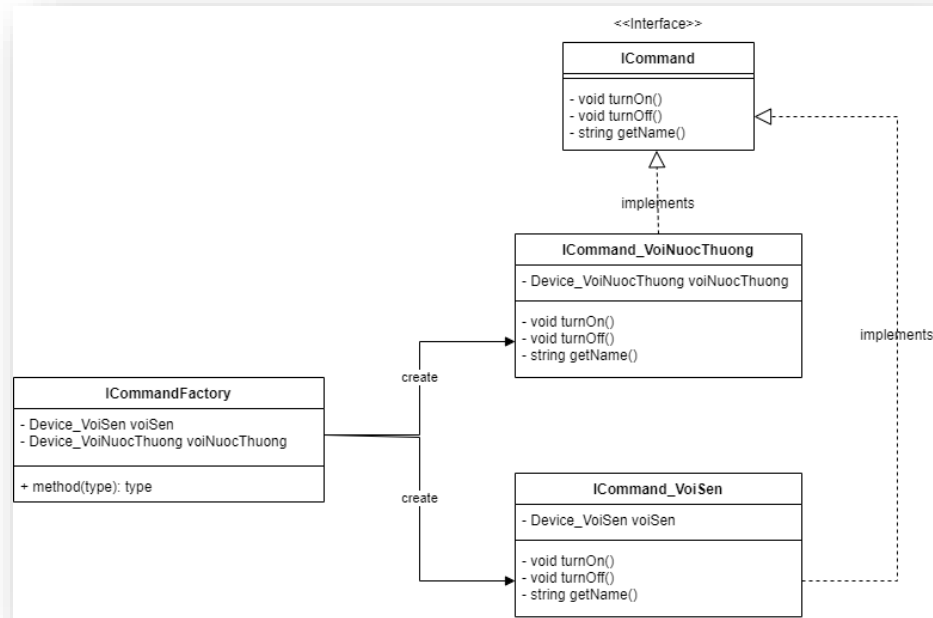
    1 reference
    public static Device getDevice1(DeviceType type, int cubicmetre_id)
    {
        if (type == DeviceType.VoiSen)
        {
            return new Device_VoiSen(cubicmetre_id);
        }
        else if(type == DeviceType.VoiNuocThuong)
        {
            return new Device_VoiNuocThuong(cubicmetre_id);
        }
        else
        {
            return null;
        }
    }
}
```

Hình 2.18 Code áp dụng Factory Method Pattern (4)

## 2.5 Command Pattern

Lý do áp dụng: Mỗi thiết bị nước gồm có hai chức năng: mở (turn on) và đóng (turn off). Để khách hàng có thể thực hiện được hai chức năng trên thì Command Pattern

là một sự lựa chọn hợp lý để áp dụng. Command Pattern là một pattern giúp người dùng thực hiện một hành động hoặc kích hoạt một sự kiện nào đó.



Hình 2.19 Sơ đồ lớp Command Pattern

```

//Ap Command pattern
//Command
5 references
public interface ICommand
{
    //void execute();
    3 references
    void turnOn();

    3 references
    void turnOff();

    3 references
    string getName();
}
    
```

Hình 2.20 Code áp dụng Command Pattern (1)

```

//Ap command pattern
//ConcreteCommand

2 references
public class ICommand_VoiSen: ICommand
{
    private Device_VoiSen voiSen;

    2 references
    public string getName()
    {
        return "VoiSen";
    }

    1 reference
    public ICommand_VoiSen(Device_VoiSen voiSen)
    {
        this.voiSen = voiSen;
    }

    2 references
    public void turnOff()
    {
        voiSen.turnOff();
    }

    2 references
    public void turnOn()
    {
        voiSen.turnOn();
    }
}

```

Hình 2.21 Code áp dụng Command Pattern (2)

```

//Ap command pattern
//ConcreteCommand
2 references
public class ICommand_VoiNuocThuong: ICommand
{
    private Device_VoiNuocThuong voiNuocThuong;

    1 reference
    public ICommand_VoiNuocThuong(Device_VoiNuocThuong voiNuocThuong)
    {
        this.voiNuocThuong = voiNuocThuong;
    }

    2 references
    public string getName()
    {
        return "VoiNuocThuong";
    }

    2 references
    public void turnOff()
    {
        voiNuocThuong.turnOff();
    }

    2 references
    public void turnOn()
    {
        voiNuocThuong.turnOn();
    }
}

```

Hình 2.22 Code áp dụng Command Pattern (3)

```

//Ap command pattern
//
1.reference
class ICommandFactory
{
    1.reference
    public ICommand createCommand(String type)
    {
        Customers customers = new Customers();
        Customers customer = customers.findCustomer(Program.phone);

        //Voisen
        Device_VoiSen voiSen;
        voiSen = (Device_VoiSen)Device.getDevice1(DeviceFactory.DeviceType.VoiSen, customer.CubicMetre_ID);

        //Voi nuoc thuong
        Device_VoiNuocThuong voiNuocThuong;
        voiNuocThuong = (Device_VoiNuocThuong)Device.getDevice1(DeviceFactory.DeviceType.VoiNuocThuong, customer.CubicMetre_ID);

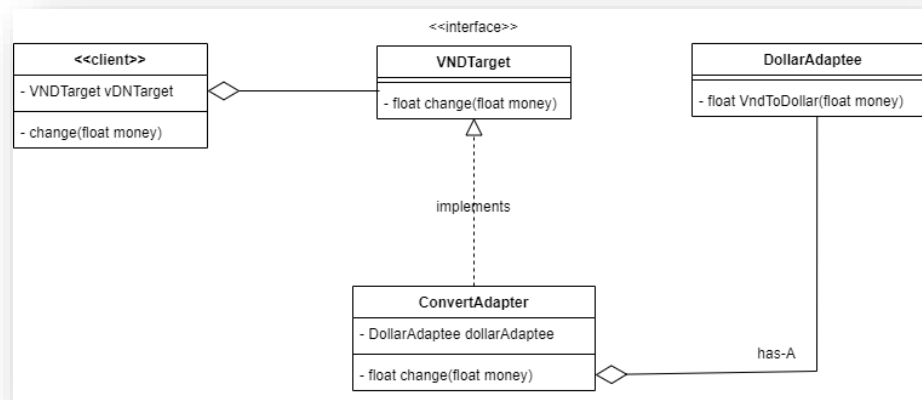
        if (type == "VoiSen")
        {
            return new ICommand_VoiSen(voiSen);
        }
        else
        {
            return new ICommand_VoiNuocThuong(voiNuocThuong);
        }
    }
}

```

Hình 2.23 Code áp dụng Command Pattern (4)

## 2.6 Adapter Pattern

Lý do áp dụng: Nhiều khách hàng sẽ có nhu cầu chuyển tiền thanh toán từ đơn vị tiền tệ VND sang các đơn vị tiền tệ khác để dễ dàng phục vụ cho việc thanh toán của mỗi người. Vì vậy, chúng ta sẽ áp dụng Adapter Pattern để chuyển từ đơn vị VND, đơn vị tiền tệ mặc định khi Công ty thông báo chi phí thanh toán về cho mỗi khách hàng, sang các đơn vị tiền tệ khác, ở trong trường hợp này là đơn vị Dollar. Nếu muốn thêm một số đơn vị tiền tệ khác chúng ta chỉ cần tạo 1 class Adaptee mới mà không cần thay đổi cấu trúc code.



Hình 2.24 Sơ đồ lớp Adapter Pattern

```
//Ap adapter pattern
//Adapter
2 references
class ConvertAdapter: VNDTarget
{
    public DollarAdaptee dollarAdaptee;

    1 reference
    public ConvertAdapter(DollarAdaptee dollarAdaptee)
    {
        this.dollarAdaptee = dollarAdaptee;
    }

    2 references
    public float change(float money)
    {
        float dollar = this.dollarAdaptee.VndToDollar(money);
        return dollar;
    }
}
```

Hình 2.25 Code áp dụng Adapter Pattern (1)

```
//Ap adapter pattern
//Adaptee
3 references
class DollarAdaptee
{
    1 reference
    public float VndToDollar(float money)
    {
        return money / 22.0f;
    }
}
```

Hình 2.26 Code áp dụng Adapter Pattern (2)

```
//Ap adapter pattern
//Target
2 references
public interface VNDTarget
{
    2 references
    float change(float money);
}
```

Hình 2.27 Code áp dụng Adapter Pattern (3)

```
private void btn_convertToDollar_Click(object sender, EventArgs e)
{
    float money = float.Parse(lb_Pay.Text.ToString());

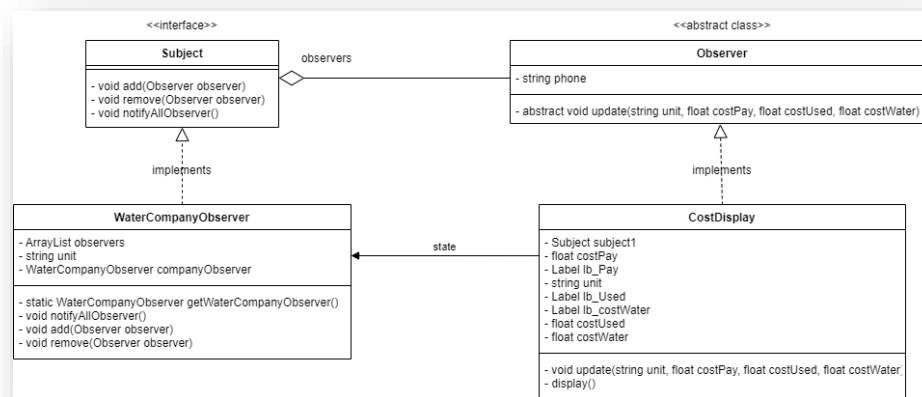
    //chuyen vnd sang dollar
    VNDTarget vNDTarget = new ConvertAdapter(new DollarAdaptee());
    float moneyConverted = vNDTarget.change(money);
    //Làm tròn 2 chữ số Math.Round
    lb_dollar.Text = Math.Round(moneyConverted, 2) + " $";
}
```

Hình 2.28 Code áp dụng Adapter Pattern (4)



## 2.7 Observer Pattern

Lý do áp dụng: Cuối mỗi tháng thì Công ty sẽ thông báo tiền nước về cho mỗi khách hàng. Thông tin chi tiết tiền nước sẽ được hiển thị trên mỗi giao diện khách hàng khi Công ty nhấn nút thông báo. Ở đây chúng ta có Observer Pattern giúp định nghĩa mỗi phụ thuộc một – nhiều giữa các đối tượng để khi mà một đối tượng có sự thay đổi trạng thái, tất cả các thành phần phụ thuộc của nó sẽ được thông báo và cập nhật một cách tự động. Vậy nên Observer Pattern rất phù hợp để áp dụng trong trường hợp này.



Hình 2.29 Sơ đồ lớp Observer Pattern

```

//Ap observer pattern
//subject class
3 references
public interface Subject
{
    2 references
    void add(Observer observer);
    1 reference
    void remove(Observer observer);
    2 references
    void notifyAllObserver();
}
  
```

Hình 2.30 Code áp dụng Observer Pattern (1)

```

//Ap observer pattern
//Observer class
9 references
public abstract class Observer
{
    public string phone;
    1 reference
    public Observer(string phone)
    {
        this.phone = phone;
    }

    0 references
    public string PhoneCustomer { get; set; }
    2 references
    public abstract void update(string unit, float costPay, float costUsed, float costWater);
}

```

Hình 2.31 Code áp dụng Observer Pattern (2)

```

//Ap observer pattern
//ConcreteSubject
7 references
public class WaterCompanyObserver : Subject
{
    public ArrayList observers = new ArrayList();
    public string unit;
    public static WaterCompanyObserver companyObserver = new WaterCompanyObserver();

    2 references
    public static WaterCompanyObserver getWaterCompanyObserver()
    {
        return companyObserver;
    }

    2 references
    public void notifyAllObserver()
    {
        CustomerSql customerSql = new CustomerSql();
        CubicMetre cubicMetre = new CubicMetre();

        for (int i = 0; i < observers.Count; ++i)
        {
            Observer observer = (Observer)observers[i];

            Customers customer = customerSql.findCustomer(observer.phone);
            CalculatorTemplate calculatorTemplate = CalculatorFactory.getCalculatorTemplate(customer.Customer_kind, customer.Customer_phone);

            float costWater;

            if (customer.Customer_kind == "Ho gia dinh")
            {
                costWater = WaterCompany.getInstance().Water_HogiadinhCost;
            }
            else if (customer.Customer_kind == "Ho ngheo")
            {
                costWater = WaterCompany.getInstance().Water_HongheoCost;
            }
            else
            {
                costWater = WaterCompany.getInstance().Water_DoanhnghiepCost;
            }

            CubicMetre c = cubicMetre.findCubicMetre(customer.CubicMetre_ID);

            observer.update(this.unit, calculatorTemplate.Pay(), c.CubicMetre_count, costWater);
        }
    }
}

```

Hình 2.32 Code áp dụng Observer Pattern (3)

```

1 reference
public void add(Observer observer)
{
    observers.Add(observer);
}

1 reference
public void remove(Observer observer)
{
    int i = observers.IndexOf(observer);
    if (i != -1)
    {
        observers.RemoveAt(i);
    }
}
}

```

Hình 2.33 Code áp dụng Observer Pattern (4)

```

//Ap observer pattern
//ConcreteObserver
3 references
public class CostDisplay : Observer
{
    public Subject subject1;
    public float costPay;
    public Label lb_Pay;
    public string unit;
    public Label lb_Used;
    public Label lb_costWater;
    public float costUsed;
    public float costWater;

    1 reference
    public CostDisplay(Subject subject, float costPay, string phone, Label lb_Pay, Label lb_Used, Label lb_costWater): base(phone)
    {
        subject1 = subject;
        this.costPay = costPay;
        this.lb_Pay = lb_Pay;
        this.lb_Used = lb_Used;
        this.lb_costWater = lb_costWater;
        subject.add(this);
    }

    2 references
    public override void update(string unit, float costPay, float costUsed, float costWater)
    {
        this.unit = unit;
        this.costPay = costPay;
        this.costUsed = costUsed;
        this.costWater = costWater;
        display();
    }

    1 reference
    public void display()
    {
        lb_Pay.Text = string.Format("{0:0.00}", costPay);
        lb_Used.Text = costUsed.ToString() + " m3";
        lb_costWater.Text = costWater.ToString() + " VND/m3";
    }
}

```

Hình 2.34 Code áp dụng Observer Pattern (5)

## TÀI LIỆU THAM KHẢO

- [1] GP Coder (2018). “Hướng dẫn Java Design Pattern – Singleton”, <https://gpcoder.com/4190-huong-dan-java-design-pattern-singleton/>, truy cập ngày 20/10/2021.
- [2] GP Coder (2019). “Hướng dẫn Java Design Pattern – Strategy”, <https://gpcoder.com/4796-huong-dan-java-design-pattern-strategy/>, truy cập ngày 23/10/2021.
- [3] GP Coder (2019). “Hướng dẫn Java Design Pattern – Template Method”, <https://gpcoder.com/4810-huong-dan-java-design-pattern-template-method/>, truy cập ngày 25/10/2021.
- [4] GP Coder (2018). “Hướng dẫn Java Design Pattern – Factory Method”, <https://gpcoder.com/4352-huong-dan-java-design-pattern-factory-method/>, truy cập ngày 25/10/2021.
- [5] GP Coder (2018). “Hướng dẫn Java Design Pattern – Command”, <https://gpcoder.com/4686-huong-dan-java-design-pattern-command/>, truy cập ngày 27/10/2021.
- [6] GP Coder (2018). “Hướng dẫn Java Design Pattern – Adapter”, <https://gpcoder.com/4483-huong-dan-java-design-pattern-adapter/>, truy cập ngày 23/10/2021.
- [7] GP Coder (2018). “Hướng dẫn Java Design Pattern – Observer”, <https://gpcoder.com/4747-huong-dan-java-design-pattern-observer/>, truy cập ngày 25/10/2021.