**BỘ GIÁO DỤC VÀ ĐÀO TẠO TRƯỜNG ĐẠI HỌC NÔNG LÂM**

**Khoa Công Nghệ Thông Tin**

🙠🙠🙠 🕮 🙢🙢🙢

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | z4105743354911_1ad6537fc1823f55d020dbb5941efff1 | |  |
| **ĐỒ ÁN THIẾT KẾ HƯỚNG ĐỐI TƯỢNG** | | | | |
| **Đề tài: Game Tetris sử dụng javaSwing** | | | | |
|  | Giáo viên hướng dẫn | | : Lê Phi Hùng | | |
|  | Sinh viên thực hiện  Mssv | | : Phạm Thùy Tiên  : 20130432 | | |
|  | Lớp | | : DH20DTC | | |

Ngày … Tháng … Năm 2013

**Mục Lục**

[Phần I. Giới thiệu 4](#_Toc1433)

[1. Giới thiệu game 4](#_Toc26376)

[2. Mô tả game 4](#_Toc8965)

[3. Xác định yêu cầu của game Tetris 5](#_Toc30242)

[3.1. Phần giao diện 5](#_Toc12223)

[3.2. Phần xử lí sự kiện 5](#_Toc20861)

[Phần II. Biểu đồ Use-case 5](#_Toc2607)

[1. Biểu đồ Use-case tổng quát 5](#_Toc10567)

[Phần III. Áp dụng Pattern vào bài làm 6](#_Toc18347)

[1. Pattern Factory 6](#_Toc13291)

[1.1. Phân tích chức năng 6](#_Toc10646)

[1.2. Trách nhiệm từng class của Pattern 6](#_Toc10052)

[1.3. Design Class Diagram 7](#_Toc10332)

[2. Pattern Strategy 8](#_Toc13345)

[2.1. Phân tích chức năng 8](#_Toc11094)

[2.2. Kiến trúc class của Pattern 8](#_Toc24343)

[2.3. Design Class Diagram 9](#_Toc31974)

[3. Pattern Observer 10](#_Toc10422)

[3.1. Phân tích chức năng 10](#_Toc17472)

[3.2. Trách nhiệm từng class của Pattern 10](#_Toc18589)

[3.3. Design Class Diagram 11](#_Toc23236)

[12](#_Toc6662)

[4. Pattern Decorator 12](#_Toc16592)

[4.1. Phân tích chức năng 12](#_Toc3508)

[4.2. Trách nhiệm từng class của Pattern 12](#_Toc22128)

[4.3. Design Class Diagram 12](#_Toc21623)

[5. Pattern Composite 13](#_Toc24159)

[5.1. Phân tích chức năng 13](#_Toc20124)

[5.2. Trách nhiệm từng class của Pattern 13](#_Toc17969)

[5.3. Design Class Diagram 14](#_Toc28673)

[1. Các pattern sử dụng trong bài: 15](#_Toc17562)

[TÀI LIỆU THAM KHẢO 16](#_Toc1444)

**Phần I. Giới thiệu**

1. **Giới thiệu game**

* Tetris là một game quen thuộc với cái tên “game xếp hình”, đó là một trong những game yêu thích của nhóm từ nhỏ.
* Game có cách chơi đơn giản, giúp nhóm có thể tiếp cận để lập trình bằng những kiến thức đã được học.

1. **Mô tả game**

* Lập trình 1 chương trình game Tetris dựa trên ngôn ngữ Java.
* Luật chơi:
  + Di chuyển các khối gạch đang rơi từ từ xuống trong kích thước hình chữ nhật 20 hàng x 10 cột trên màn hình. Chỗ nào có gạch rồi thì không di chuyển được tới vị trí đó.
  + Người chơi xếp những khối hình sao cho khối hình lấp đầy 1 hàng ngang để ghi điểm và hàng ngang ấy sẽ biến mất.
  + Một nhóm 4 khối sẽ rơi từ phía trên cùng của màn hình, di chuyển các khối và xoay chúng cho đến khi chúng rơi xuống phía dưới cùng của màn hình, sau đó nhóm 4 khối tiếp theo sẽ rơi xuống.
  + Nếu để cho những khối hình cao quá màn hình, trò chơi sẽ kết thúc. Trò chơi kết thúc khi khối gạch không rơi xuống được nữa.
  + 4 phím điều khiển chính:
    - Phím mũi tên lên: xoay khối.
    - Phím mũi tên trái: di chuyển sang trái.
    - Phím mũi tên phải: di chuyển sang phải.
    - Phím mũi tên xuống: tăng tốc độ rơi.

1. **Xác định yêu cầu của game Tetris**
   1. **Phần giao diện**

Giao diện của game bao gổm :

- Màn hình hiển thị người chơi tương tác với game:

* Màn hình menu : là màn hình đầu tiên được hiện lên khi bắt đầu trò chơi. Bao gồm các thành phần :Hình nền menu(backGround), button bắt đầu trò chơi (btnNewgame), button thoát(btnQuit). Khi người dùng bấm vào btnNewgame thì sẽ hiển thị cho người chơi chọn câp độ từ 1 -> 9, bao btnLeft giảm level, btnRight tăng level, level, btnPlay để bắt đầu chơi, btnCancel thoát hệ thống.

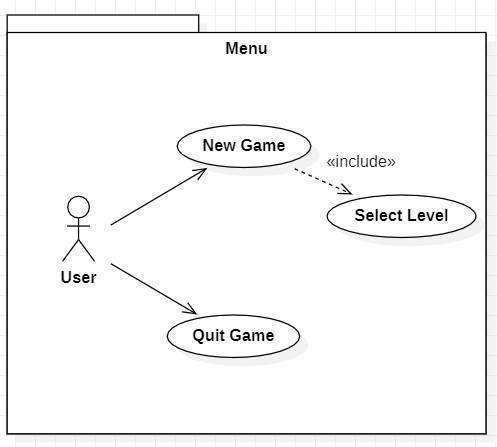
* Màn hình Game: đại diện cho màn hình trò chơi trong game Tetris.Bao gồm các thành phần: Hình nền màn hình trò chơi (backGround), màn hình khi thua game (gameOver), nút quay trờ lại màn hình Menu(backtoMenu), nút dừng trò chơi (pause), nút tiếp tục trò chơi(playAgainst), thành phần hiển thị điểm số, thành phần hiển thị khối tiếp theo (shapeNext), hiển thị cấp độ, hiển thị số đường line phá được.

- Các thành phần hiển thị hổ trợ người chơi:

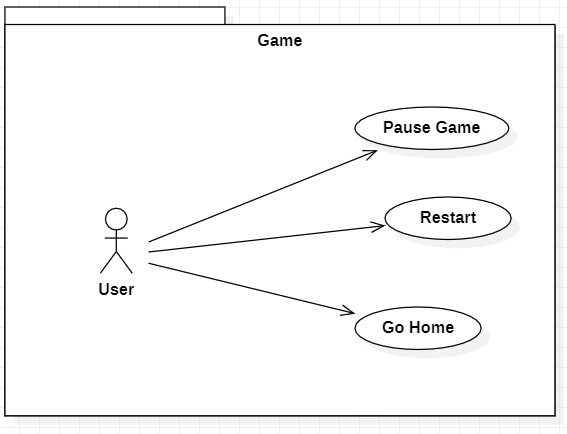
* Khối hình hiện tại (currentShape)
* Khối hình tiếp theo (nextShape)
* Bảng chơi (Board)
* Điểm chơi (Score)
* Các nút (Button)
* Khối nhỏ (Block)
  1. **Phần xử lí sự kiện**

**Phần II. Biểu đồ Use-case**

1. **Biểu đồ Use-case màn hình menu**



1. **Biểu đồ Use-case màn hình game**



**Phần III. Áp dụng Pattern vào bài làm**

1. **Pattern Factory**
   1. **Phân tích chức năng**

**-** Thiết kế 1 nhà máy để tạo ra các khối hình cho hình tiếp theo được hiển thị trong game Tetris.

* 1. **Trách nhiệm từng class của Pattern**

**+** ShapeFactory: là lớp trừu tượng định nghĩa phương thức tạo hình dạng khối hình và danh sách block cho khối hình tiếp theo đó. Phương thức tạo hình sẽ nhận vào chỉ số từ 0 -> 6 mỗi số sẽ trả về đối tượng shapeFactory tương ứng .Các lớp con của shapeFactory sẽ triển khai phương thức này để tạo hình dạng cụ thể. Được sử dụng trong class InGame khi mà cập nhật hình tiếp theo thì phương thức tạo hình tiếp theo sẽ được gọi.

+ShapeOFactory : triển khai phương thức tạo hình của lớp cha chứa vai trò nhà máy tạo hình shapeO

+ShapeLFactory : vai trò nhà máy taọ hình shapeL

+ShapeSFactory : vai trò nhà máy tạo hình shapeS

+ShapeIFactory : vai trò nhà máy tạo hình shapeI

+ShapeTFactory : vai trò nhà máy tạo hình shapeT

+ShapeZFactory : vai trò nhà máy tạo hình shapeZ

+ShapeO : Khởi tạo hình dạng và vị trí tọa độ cho hình dạng O

+ShapeJ : Khởi tạo hình dạng và vị trí tọa độ cho hình dạng J

+ShapeS : Khởi tạo hình dạng và vị trí tọa độ cho hình dạng S

+ShapeT : Khởi tạo hình dạng và vị trí tọa độ cho hình dạng T

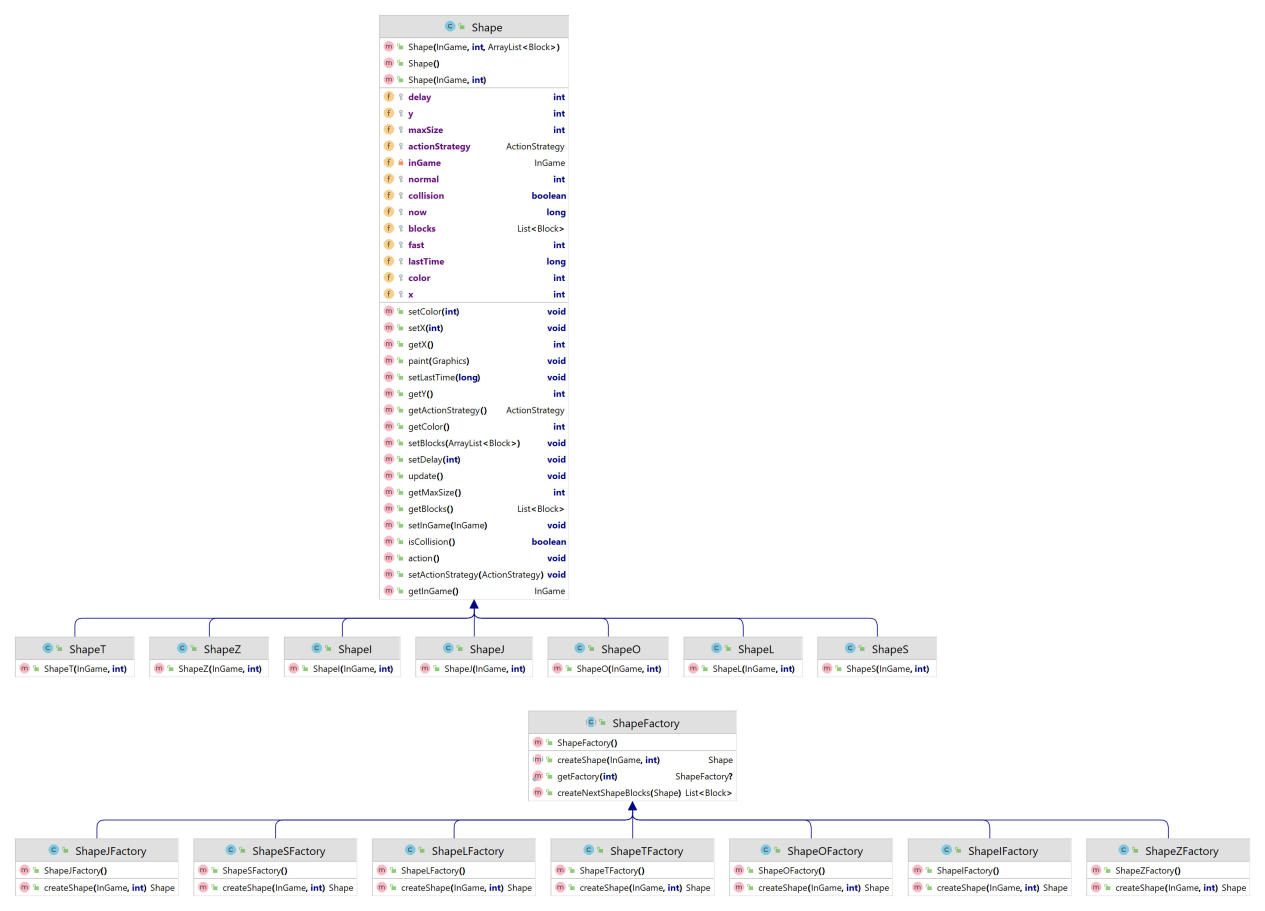
+ShapeL : Khởi tạo hình dạng và vị trí tọa độ cho hình dạng L

+ShapeZ : Khởi tạo hình dạng và vị trí tọa độ cho hình dạng Z

+ShapeI : Khởi tạo hình dạng và vị trí tọa độ cho hình dạng I.

+Shape : là đối tượng mà phương thức factory tạo ra, là lớp cơ sở của các hình dạng trong trò chơi. Triển khai giao diện.

* 1. **Design Class Diagram**



(Hinh class diagram của pattern Factory)

1. **Pattern Strategy**
   1. **Phân tích chức năng**

**-** Sử dụng để tạo ra các chiến lược hành động cho các khối hình trong game Tetris. Hiện thực các hành vi chuyển động của khối hình như di chuyển trái, di chuyển phải, xoay, di chuyển xuống đều , di chuyển xuống nhanh.

* 1. **Kiến trúc class của Pattern**

**+**ActionStategy : định nghĩa phương thức hành động của hình.

**+**Shape : Chứa ActionStategy sử dụng phương thức xác định các chiến lược hành động cho các hình dạng.

+MoveLeftActionStrategy : triển khai interface ActionStategy, cung cấp logic để thực hiện hành động di chuyển qua trái của hình dạng.

+MoveRightActionStrategy : triển khai interface ActionStategy, cung cấp logic để thực hiện hành động di chuyển qua phải của hình dạng.

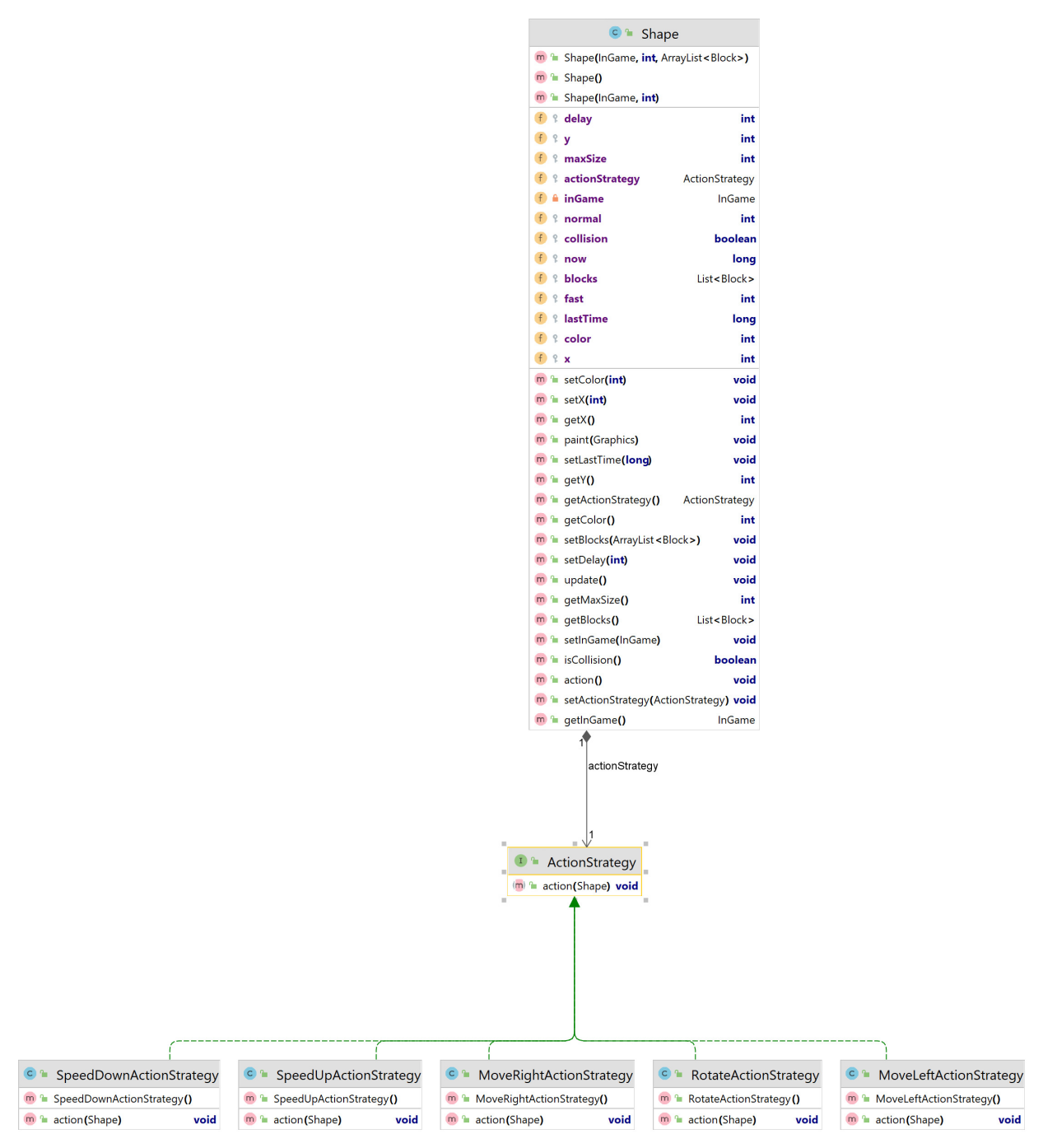
+RotateActionStrategy : triển khai interface ActionStategy, cung cấp logic để thực hiện hành động xoay của hình dạng.

+SpeedDownActionStrategy : triển khai interface ActionStategy, cung cấp logic tính toán thời gian rơi của khối hình theo từng level lên 1 level tốc độ sẽ được tăng để thực hiện hành động rơi của hình dạng.

+SpeedUpActionStrategy : triển khai interface ActionStategy, cung cấp hành động tăng tốc độ của hình, đặt thời gian di chuyển là 50.

+InGame : Chức năng thêm chiến lược hành động cho hình được áp dụng trong inGame bắt sự kiện bàn sẽ được set hành động tương ứng trên khối hình hiện tại trong InGame

* 1. **Design Class Diagram**



1. **Pattern Observer**
   1. **Phân tích chức năng**

Khi trò chơi hoạt động để điểm số (score) và khối hình tiếp theo (nextShape) hiển thị chính xác. score và nextShape là đối tượng đăng kí quan sát theo dõi InGame để cập nhật giá trị. InGame là chủ thể được theo dõi khi trạng thái thay đổi thông báo cho 2 đối tượng theo dõi là Score và Next.

* 1. **Trách nhiệm từng class của Pattern**

**+**SubjectNext : định nghĩa interface cung cấp chức năng cho Next đăng ký và hủy đăng ký, thông báo cho Next khi giá trị thay đổi.

+ SubjectScore : định nghĩa interface cung cấp chức năng cho Score đăng ký, hủy đăng ký, thông báo cho Next khi giá trị thay đổi

+ ObservableScore : định nghĩa phương thức cập nhật số điểm và số hàng cho lớp theo dõi

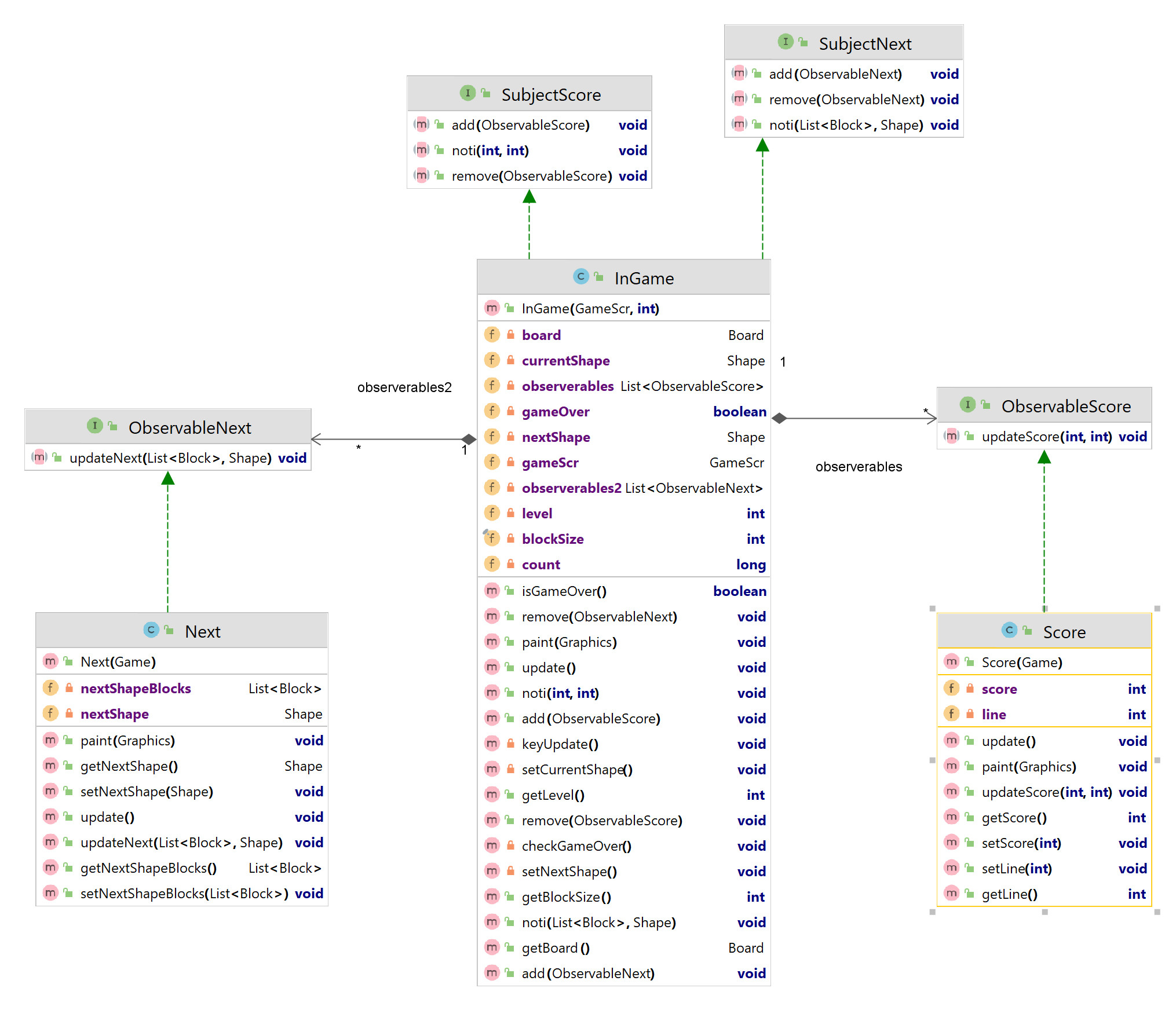
+ ObservebleNext: định nghĩa phương thức cập nhật với dữ liệu là danh sách block của hình tiếp theo và hình tiếp theo.

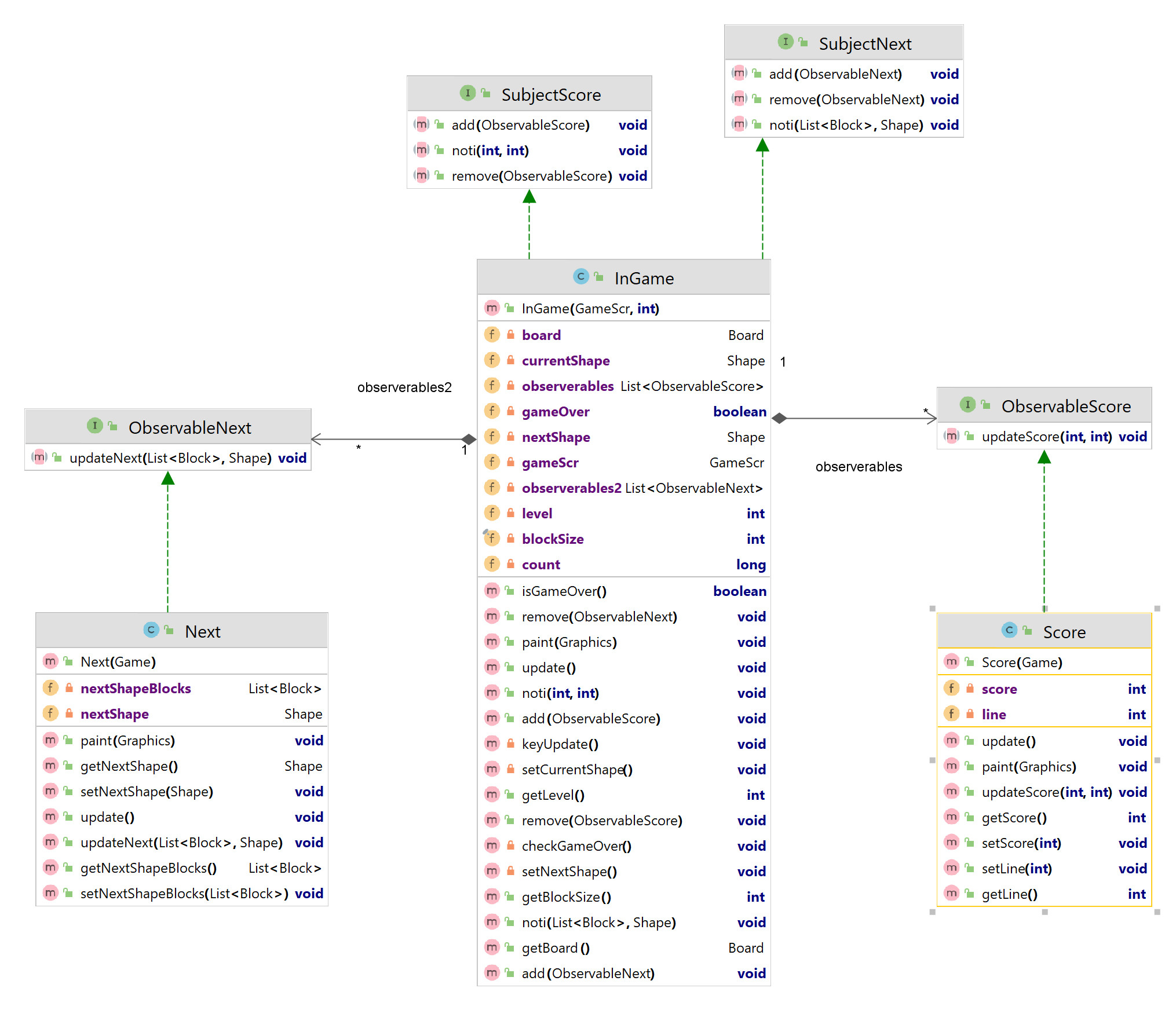
+ Score: Cài đặt phương thức cập nhật lại số điểm và số hàng phá được , được nhận thông báo cập nhật từ InGame

+Next : Cài đặt phương thức cập nhật khối hình hiển thị tiếp theo. Nhận vào danh sách block tiếp theo và hình tiếp theo để cập nhật.

+InGame:Cài đặt 2 interface của SubjectNext và SubjectScore quản lí 2 danh sách đối tượng quan sát là Score và Next. Khi mà trong bảng chơi có hàng được phá sẽ được gọi thông báo của Score truyền giá trị điểm. Khi factory tạo ra hình tiếp theo thì thông báo truyền dữ liệu cho Next.

* 1. **Design Class Diagram**





(Hình ảnh class diagram Observer Pattern được chạy trong intellij)

1. **Pattern Decorator**
   1. **Phân tích chức năng**

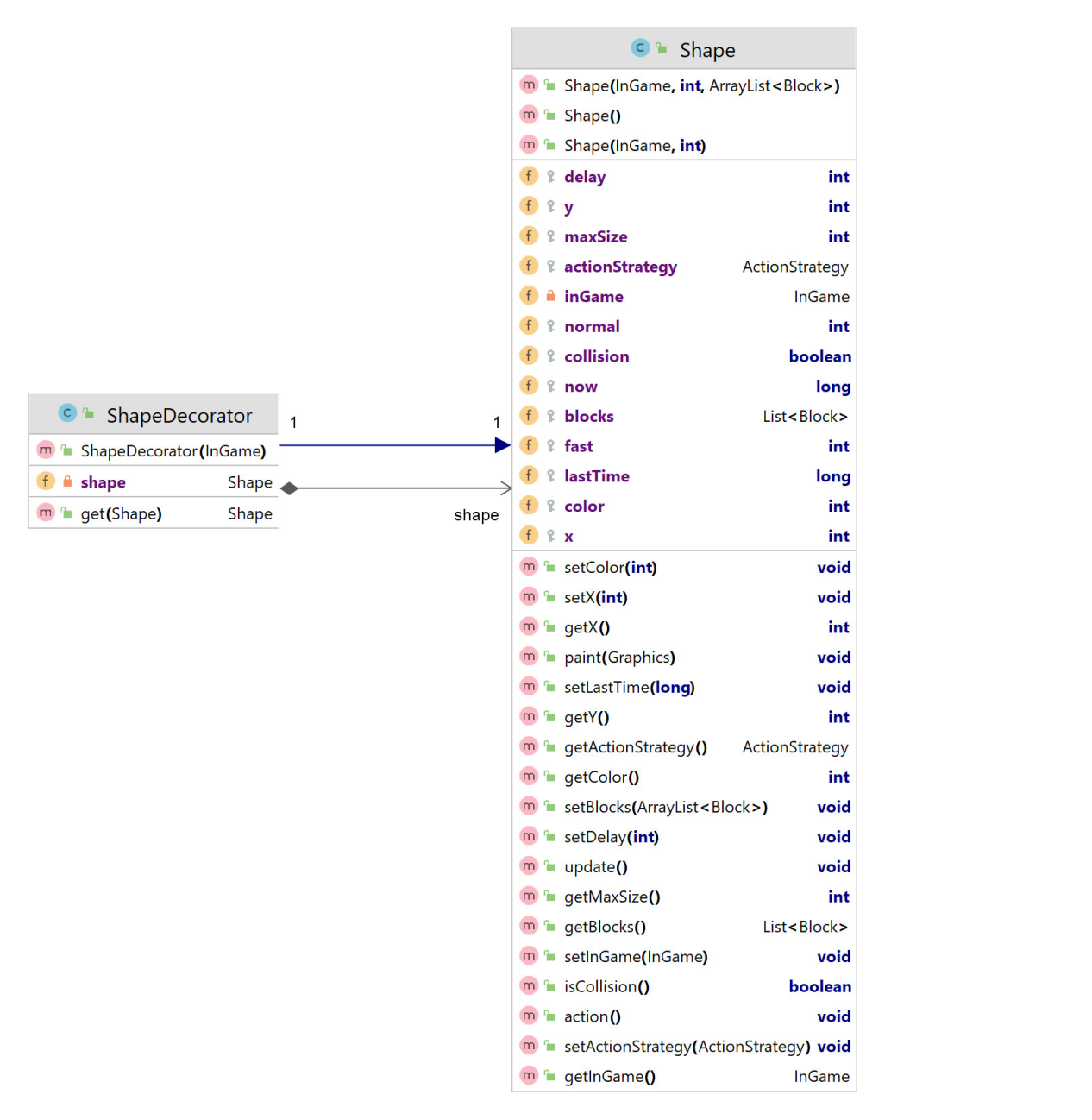
**-** Thực hiện trang trí(decorator) cho hình bằng cách thay đổi thành màu sắc đặc biêt. Đếm số lượng khối hình next tạo ra cứ 10 khối rơi xuống sẽ xuất hiện 1 hình đặc biệt.

* 1. **Trách nhiệm từng class của Pattern**

**+**ShapeDecorator : tạo ra hình với màu sắc đặc biệt bằng cách hiện thực phương thức get(Shape shape) nhận vào một hình là tham số đầu vào để trang trí cho nó, đặt màu sắc là 7

+Shape : là đối tượng được trang trí.

* 1. **Design Class Diagram**



1. **Pattern Composite**
   1. **Phân tích chức năng**

**-** Tổ chức thành phần phân cấp biễu diễn 1 cấu trúc cây phân câp đối tượng hình và các khối nhỏ trong hình. Đồng nhất việc cập nhật giá trị của cả thành phần và phần nhỏ. Các khối block được vẽ và cập nhật theo cùng 1 giao diện

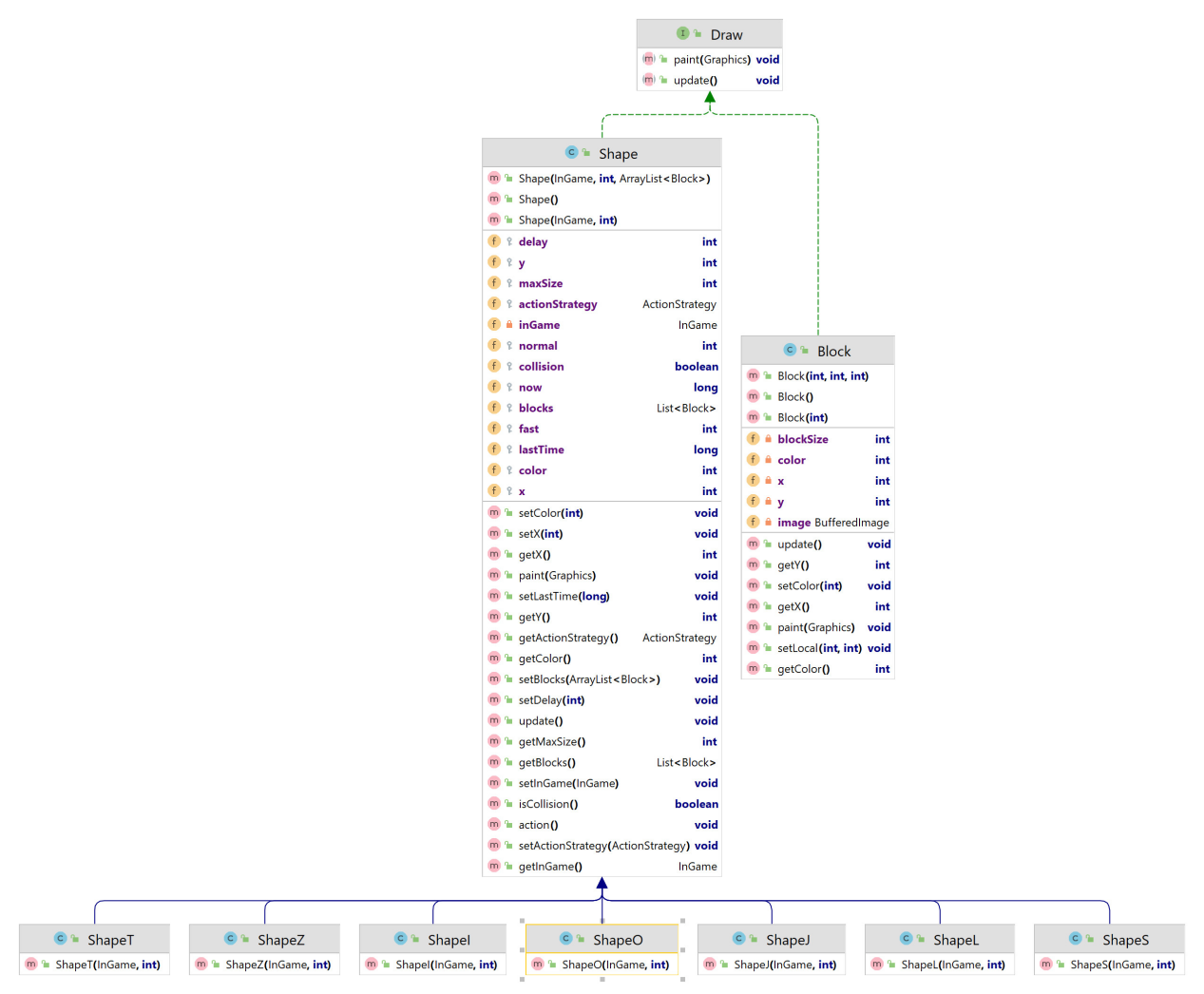
* 1. **Trách nhiệm từng class của Pattern**

**+**Shape : là thành phần tổ hợp (composite) lưu trữ danh sách thành phần con block, quản lí vị trí, màu sắc, và kích thước của hình dạng. Thực hiện phương thức paint(Graphic g) để vẽ những block thành phần lên giao diện. Thực hiện phương thức update() để cập nhật trạng thái bao gồm di chuyển các khối thành phần và kiểm tra va chạm.

+Block : là thành phần con (leaf) lưu trữ thông tin về màu sắc, vị trí, hình ảnh của một khối. Thực hiện phương thức paint() để vẽ giao diện. Phương thức update() để cập nhật trạng thái của khối.

+Draw: là interface cài đặt phương thức cho Shape và Block

* 1. **Design Class Diagram**



1. **Pattern MVC**
   1. **Phân tích chức năng**

**-** Tổ chức thành phần lớp và package thành những phần có chung logic theo model chỉ chứa dữ liệu, view chỉ hiển thị các dữ liệu lên và controller để xử lý những điều hướng và tác động từ người dùng vào view

* 1. **Trách nhiệm từng class của Pattern**

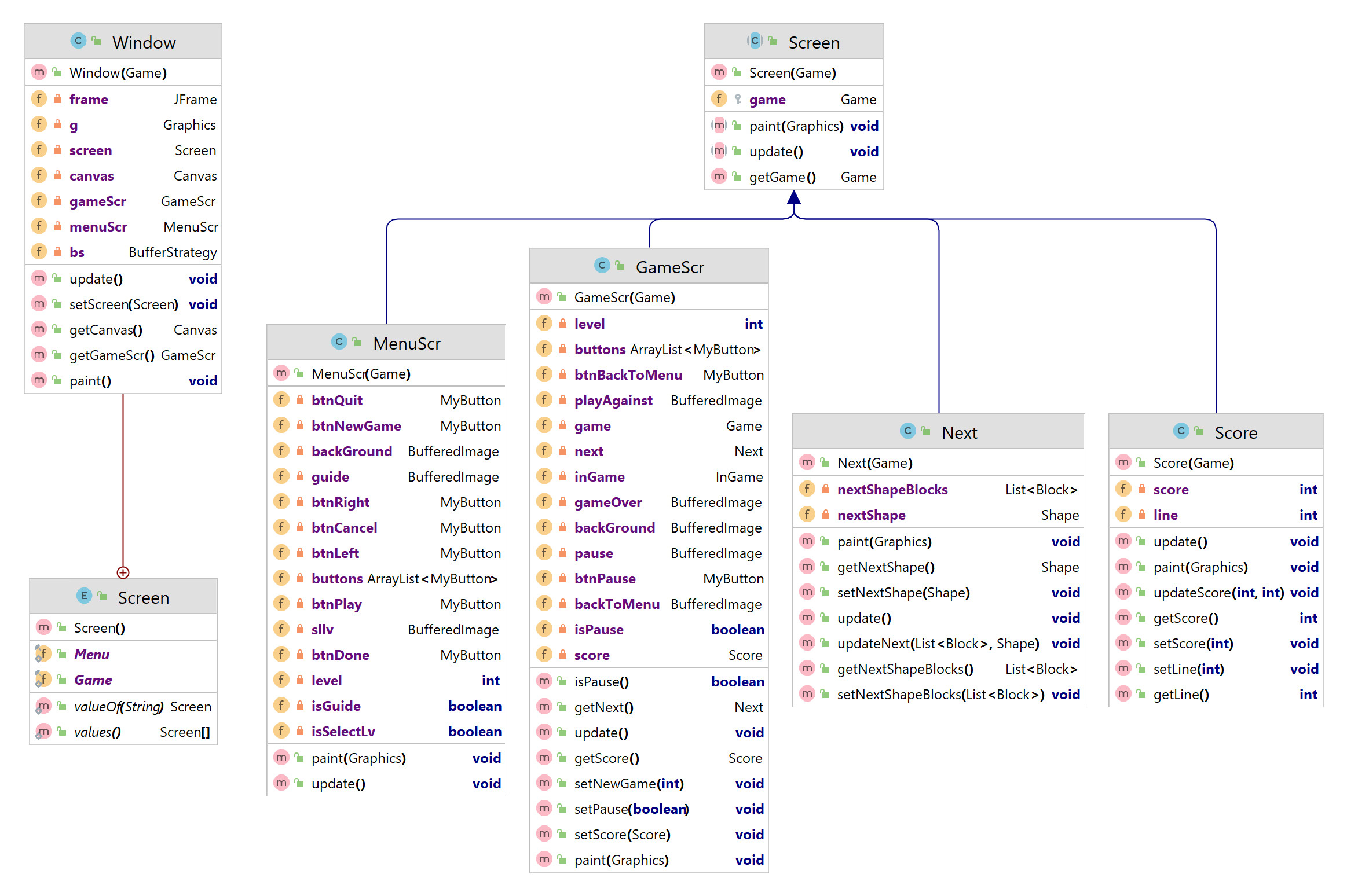
+ model: chứa dữ liệu

+ view: hiển thị lên màn hình

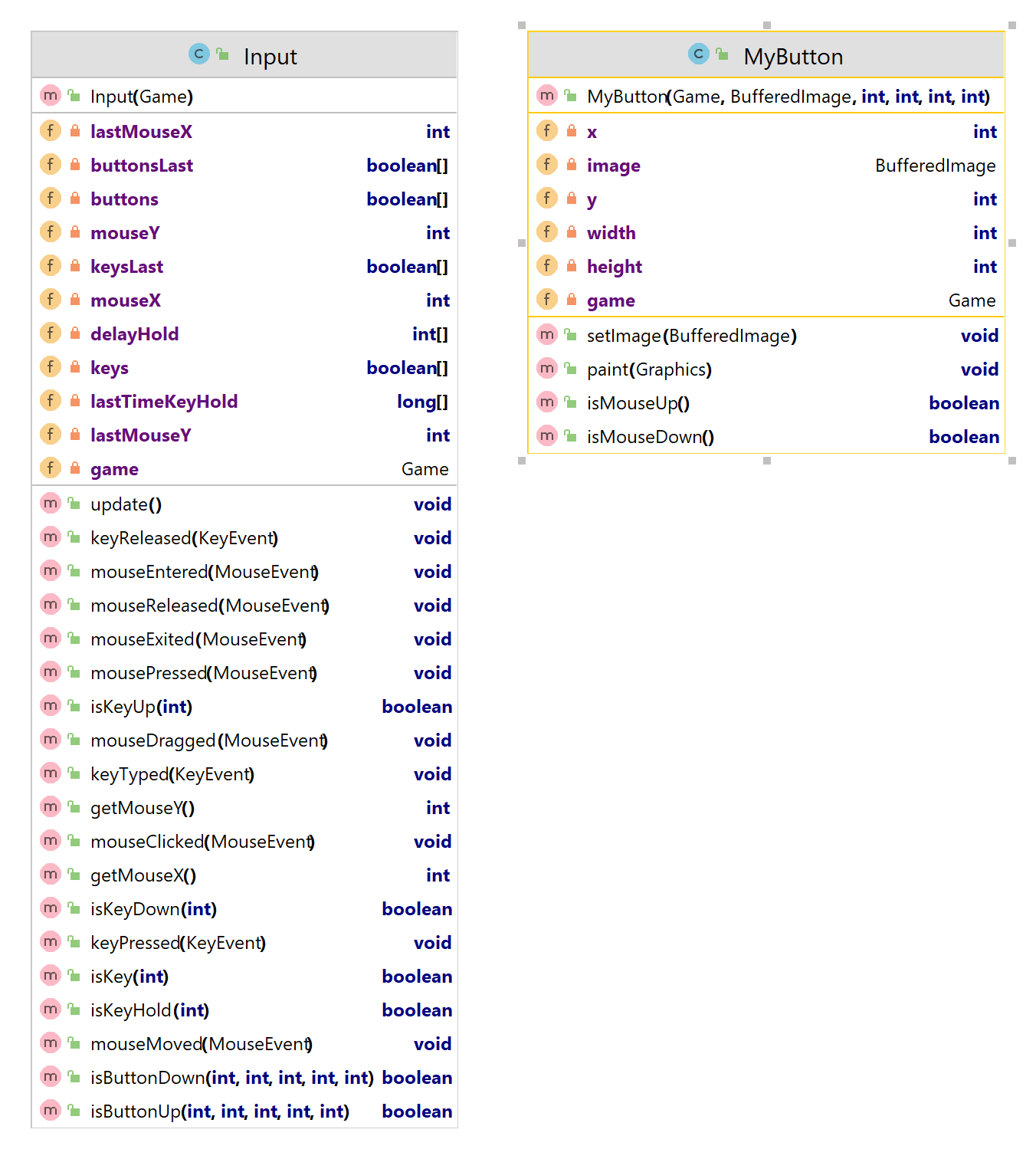
+ controller: điều hướng xử lý

* 1. **Design Class Diagram**

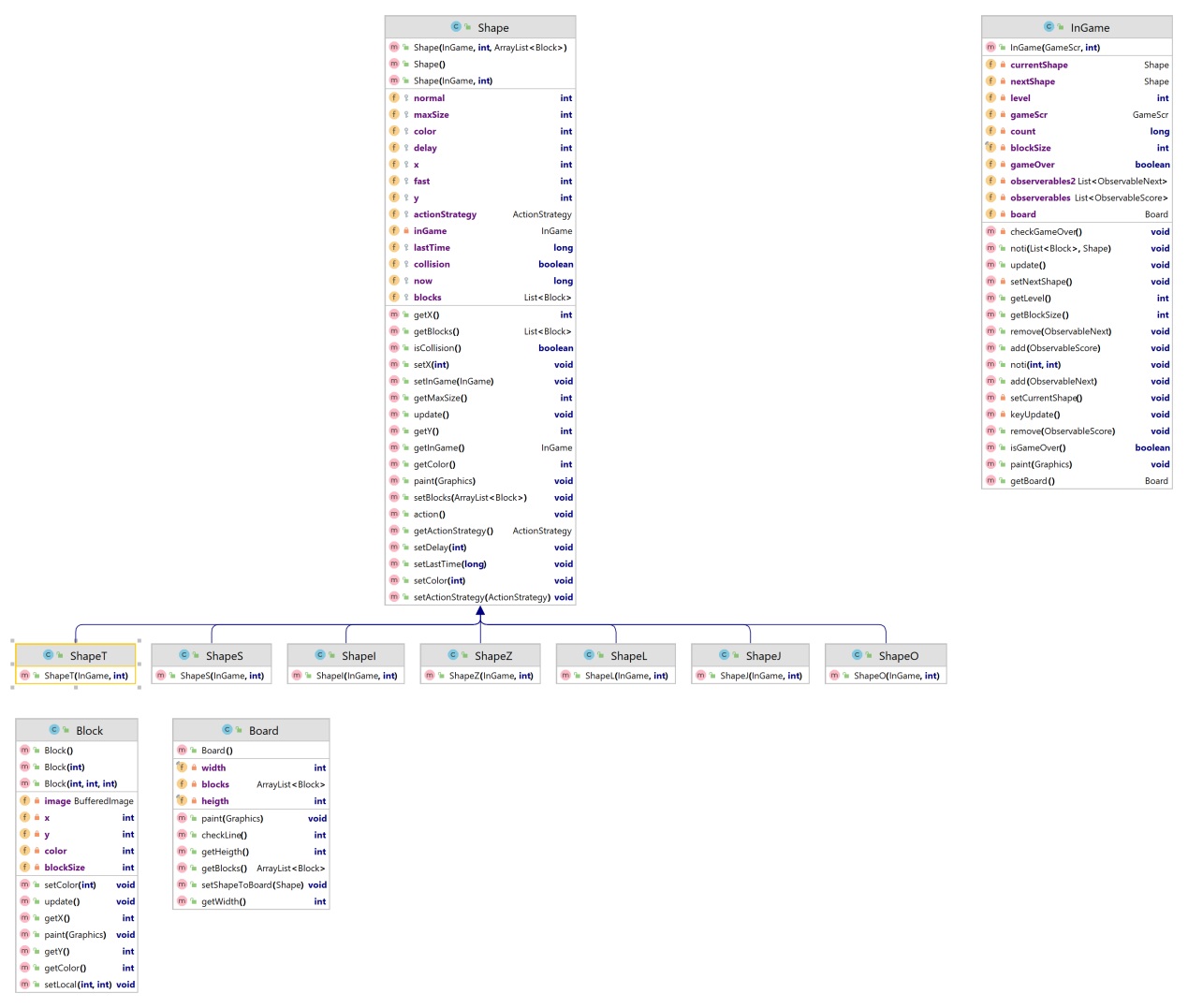
**view**



controller



model



Ý tưởng game

* Ý tưởng là chạy một vòng lặp vô hạn để liên tục xử lý các thuật toán của game và hiển thị thông tin đã xử lý bằng giao diện đồ họa.
* Gói engine: khởi tạo game và bắt đầu vòng lặp của game.
  + Lớp Game: Khởi tạo và bắt đầu vòng lặp game.
  + Lớp InPut: Nhận diện các sự kiện.
  + Lớp FileLoader: đọc các tệp dữ liệu game.
* Gói thuật toán xử lý: Xử lý các use case.
  + Lớp InGame: Tạo ra các hình, bảng và tính toán các điều kiện để đưa ra các trạng thái của game.
  + Lớp Shape: Các hình dạng của hình.
  + Board: Bảng lưới các hình di chuyển trong đó.
* Gói giao diện đồ họa: Hiển thị game.
  + Lớp MyButton: Các nút trong giao diện.
  + Lớp Window: Nền của các màn hình game.
  + Lớp MenuScr: Màn hình menu.
  + Lớp GameScr: Màn hình in game.

# Các pattern sử dụng trong bài:

* Observer pattern : để thông báo đến pannel chứa khối xuất hiện tiếp theo cập nhật và thay đổi
* Factory pattern: để tạo ra các khối ngẫu nhiên khác nhau
* Strategy pattern: để thực hiện các chiến lược di chuyển của 1 khối
* Composite pattern: để hình thành 1 khối lớn từ các khối nhỏ
* MVC pattern: để chia package class và quản lý logic được rõ ràng hơn

TÀI LIỆU THAM KHẢO

* Ý tưởng về vòng lặp game: <https://www.youtube.com/watch?v=4iPEjFUZNsw&list=PL7dwpoQd3a8j6C9p5LqHzYFSkii6iWPZF>
* Tham khảo thuật toán: <https://github.com/janbodnar/Java-Tetris-Game>