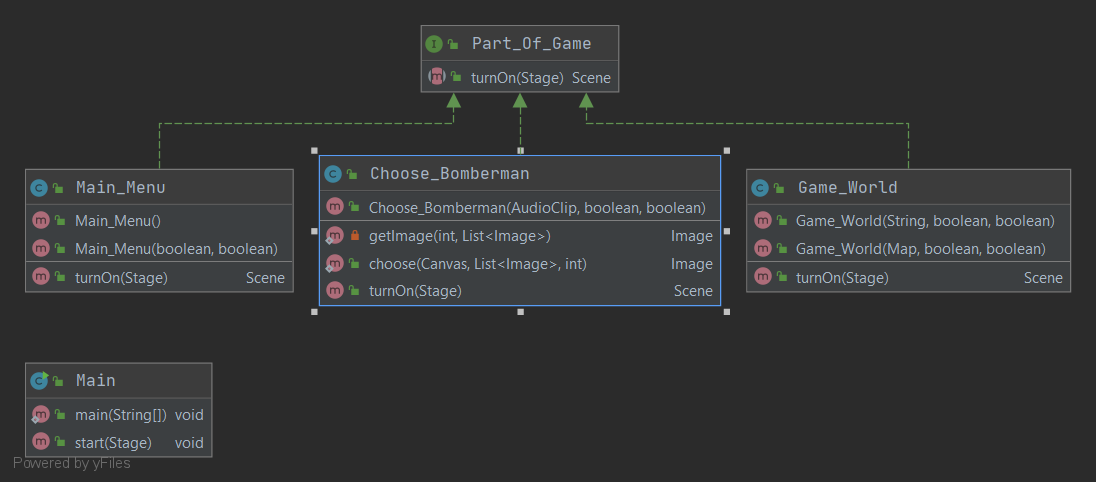
**Sinh viên Nguyễn Minh Quang**

**mssv 19020405**

**lớp INT2204 7**

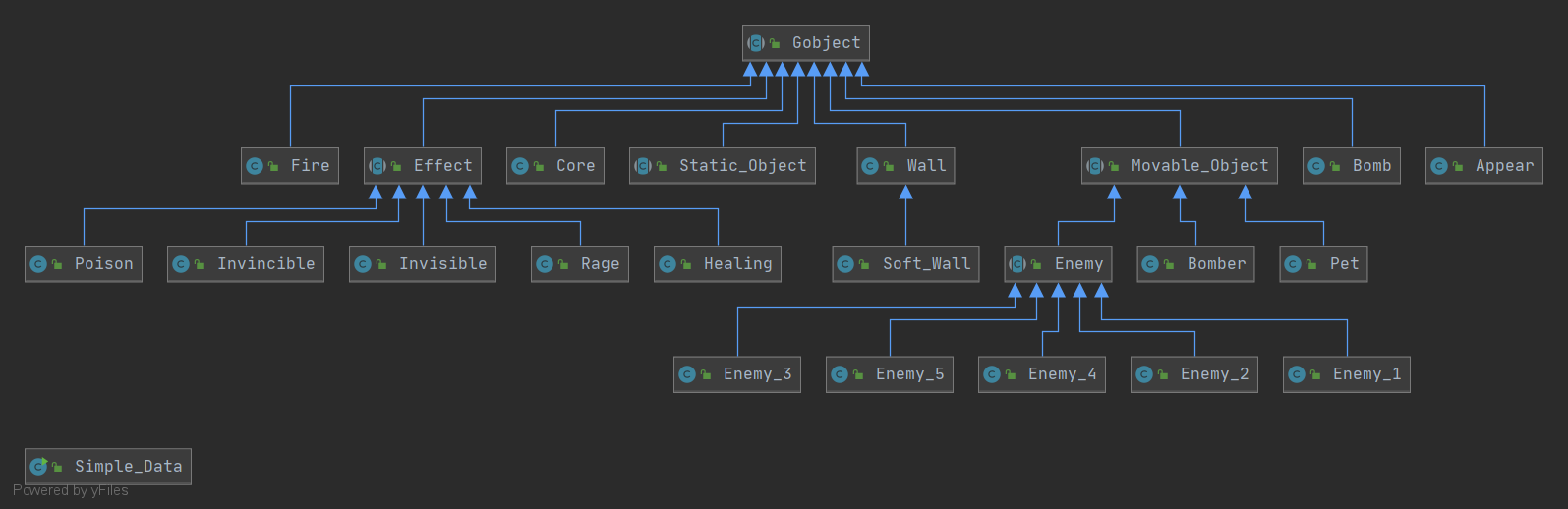
**mô tả bài tập lớn Bomber Man**

1. **Thiết kế đối tượng**
2. **Đối tượng thành phần chương trình**



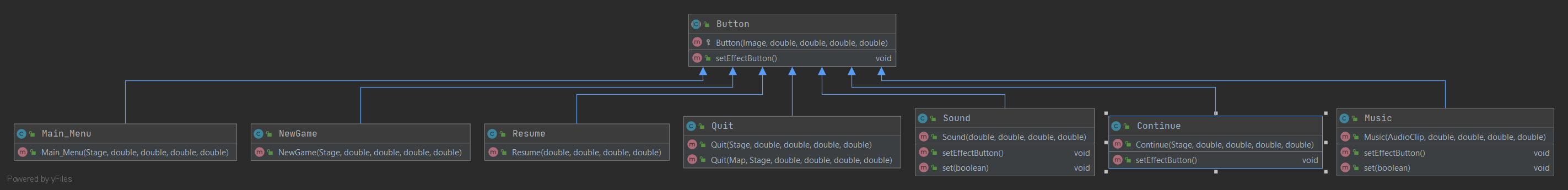
* Về cơ bản, chương trình được chia làm 3 phần chính:
* Main\_menu chứa các tùy chọn chính, là phần đầu tiên được gọi
* Choose\_Bomberman thực hiện lựa chọn nhân vật
* Game\_World chứa vong lặp chính của Game, phục vụ điều khiển game
* Các thành phần trên đều có khả năng gọi được (qua việc khởi tạo và gọi hàm turnOn) do đó cho phép liên kết các phần của chương trình linh hoạt
* Việc chia thành các phần này, tuy về mặt logic chưa thực sự rõ ràng, nhưng khối lượng công việc của chúng lại tương đối mạch lạc và khá tương đồng. trong các bản cập nhập tiếp theo (nếu có), chúng sẽ tiếp tục được chia nhỏ

1. **Đối tượng game**



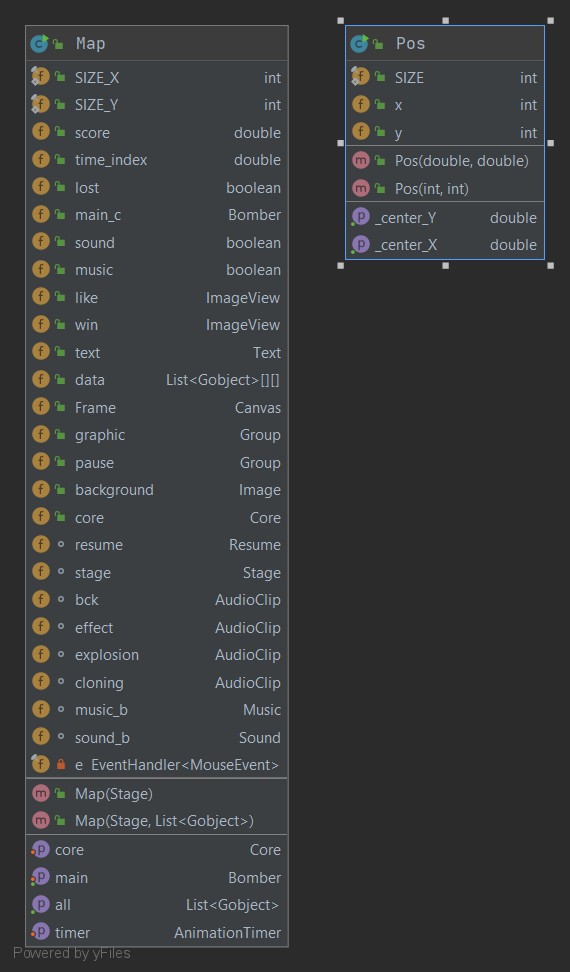
* Tất cả đối tượng xuất hiện trong vòng lặp chính của game đều là Gobject
* Trong lớp con của Gobject, có một số lớp quan trọng:
* Movable\_Object: các đối tượng sống, có khả năng di chuyển (do đó sẽ có các hoạt ảnh đặc biệt)
* Effect: chứa các hiệu ứng (vật phẩm) được sinh ra
* Core: lõi, mục tiêu phá hủy của Bomber, có khả năng tạo ra Enemy
* Appear: tiền đề quyết định tạo ra loại Enemy nào
* Fire: tạo ra bởi Bomb, sát thương đối tượng
* Static\_Object và Pet: phục vụ update
* Bomb: bom, luôn luôn tạo ra 5 ngọn lửa xung quanh khi nổ
* Enemy: kẻ địch, có khả năng di chuyển là tấn công độc lập, với mức độ “thông minh” tăng dần (theo số hiệu)
* Bomber: nhân vật chính, được điều khiển
* Simple\_Data: một lớp cài đặt interface Serializable, chứa những thông tin quan trọng nhất của một Gobject, phục vụ cho việc lưu trữ thông tin lâu dài và khôi phục Gobject nhanh chóng (Gobject và Simple\_Data luôn có constructor phục vụ việc này). Do vậy trong game này KHÔNG có sự khởi tạo theo bản đồ như đã yêu cầu, bởi vì lưu trữ như vậy là không tối ưu khi lượng object lớn và giới hạn khả năng lưu trữ của game

1. **Một số đối tượng phụ trợ**
2. **Button**

****

* Button thừa kế lớp ImageView, với các hiệu ứng đặc biệt để biến nó trở thành button
* Mỗi button có một mục đích riêng, và thường là gọi đến một phần của chương trình (hoặc điều khiển)

1. **Support Data**



* **Map:**
* Bản đồ quản lý Gobject, nhận điều khiển từ thành phần Game\_World, trong đó có hàm update để cập nhập trạng thái từng Gobject theo thời gian thực
* Trong map có các hàm đảm bảo việc xung đột không xảy ra
* Mỗi Gobject đều chứa một tham chiếu đến bản đồ mà nó đang thuộc về, do đó việc **XỬ LÍ VA CHẠM** trở nên đồng bộ do tất cả đối tượng được cập nhập đồng thời, và khi một đối tượng muốn di chuyển phải tham chiếu đến vị trí các vật thể xung quanh nó trên bản đồ
* **Tuy không thể khởi tạo theo cách đề bài yêu cầu, map có thể được khởi tạo từ dữ liệu sẵn có, miễn là nó hợp lệ.** Cụ thể hơn là từ tập dữ liệu của Simple\_Data để khởi tạo, từ đó cho phép xây dựng nhiều vật thể hơn, trên cùng vị trí. Tuy vậy điều đó cũng có nghĩa không thể can thiệp (trừ khi xóa dữ liệu đi) vào dữ liệu bản đồ, vì chỉ chương trình này mới có thể ghi và đọc dữ liệu.
* Xử lí các sự kiện chuột và bàn phím (khi tạm dừng)
* **Pos:**
* Đối tượng vị trí, có các hàm cơ bản để xử lí vị trí
* Có hàm tìm đường phục vụ update (có thể sử dụng, nhưng chưa áp dụng)

1. **Tính năng của game**
2. **Tiếp tục màn chơi**

* Từ dữ liệu như trên đã mô tả, ta có thể khôi phục quá trình chơi mà không xảy ra xung đột (có một điều bắt buộc là chương trình phải được thoát đúng cách <button:quit>, chứ không phải close từ stage)
* Các chức năng giống với tạo màn chơi mới, và mức khó được khởi động lại (tuy nhiên sẽ không hiển thị)

1. **Tạo màn chơi mới**

* Khởi tạo màn chơi mới, với dữ liệu ban đầu ngẫu nhiên, và mức khó từ đầu

1. **Điều khiển**

* Như video demo đã mô tả, các hoạt động click vào button, điều chỉnh hướng di chuyển của nhân vật chính, đặt bom, khóa chọn nhân vật …

1. **Xử lí va chạm, bom nổ**

* Như đã nói, các va chạm được xử lí đồng thời quan hàm update của map, căn cứ theo vị trí. Khi một vật thể muốn di chuyển phải xem xét tính hợp lệ của nó thông qua map.
* Xử lí đặt bom: có 2 tiêu chí

+ vị trí hiện tại luôn hợp lệ

+ vị trí tiếp theo phải hợp lệ

* Do đó bom và bomber có thể đứng cùng 1 ô, nhưng khi Bomber ra khỏi ô đó thì Bom là vật cản
* Bom nổ:

+ Bom tạo ra lửa

+ Bom sẽ nổ sau 1 khoảng thời gian

+ lửa không phải là vật cản, là thành phần gây sát thương tại chỗ

+ vị trí của lửa miễn là trong bản đồ thì luôn hợp lệ

+ lửa sẽ biến mất sau 1 khoảng thời gian

1. **Quản lí hiệu ứng (vật phẩm)**

* Hiệu ứng không phải vật cản nên trên lý thuyến, nó gần giống lửa
* Khi tại vị trí hiệu ứng xuất hiện 1 đối tượng sống, thì đối tượng đó sẽ được hưởng hiệu ứng và chỉ đối tượng đó, bằng cách thay đổi các chỉ số của đối tượng đó, sau đó hiệu ứng biến mất
* Hiệu ứng có thời gian sống, sau 1 khoảng thời gian sẽ biến mất

1. **Hiệu ứng âm thanh**

* Cho phép bật nhạc nên và hiệu ứng âm thanh
* Luôn có khả năng điều chỉnh

1. **Khả năng tự động của Enemy**

* Enemy có khả năng tự di chuyển, tổng quát như sau:

+ enemy\_1: có khả năng rẽ, hạn chế đi vào lối cũ

+ enemy\_2: có khả năng của enemy\_1, có thêm khả năng di chuyển đến Bomber trong tầm nhìn

+ enemy\_3: tối ưu khả năng rẽ của enemy\_2, tầm nhìn xa hơn, tự di chuyển đến hiệu ứng, triệu hồi Bomb tại vị trí Bomber

+ enemy\_4: di chuyển gần giống enemy\_3 ( có thay đổi tuần suất rẽ), triệu hồi Enemy\_1

+ enemy\_5: có khả năng của enemy\_4, khả năng triệu hồi enemy\_2

* Enemy có khả năng hấp thu hiệu ứng, được sinh ra khi khởi tạo Map và gián tiếp từ Core

1. **Mức chơi và độ khó**

* Các map được sinh ra ngẫu nhiên
* Càng chơi càng khó (do damage của enemy tăng và tần suất triệu hồi enemy tăng)
* Hạn chế can thiệp vào sự tải của Map (do chỉ chương trình này ghi được)
* Điểm số: chỉ được hiển thị khi thua, tính trên thời gian chơi

1. **Khả năng tiếp tục phát triển**

* Mỗi Bomber có khả năng được điều khiển độc lập từ nhiều luồng
* Dữ liệu của Gobject (cơ bản) là khá nhẹ
* Phục vụ giao tiếp mạng
* Hàm tìm đường có sẵn, nguồn dữ liệu ảnh phong phú
* Thêm nhân vật, cách điều khiển tự động mới