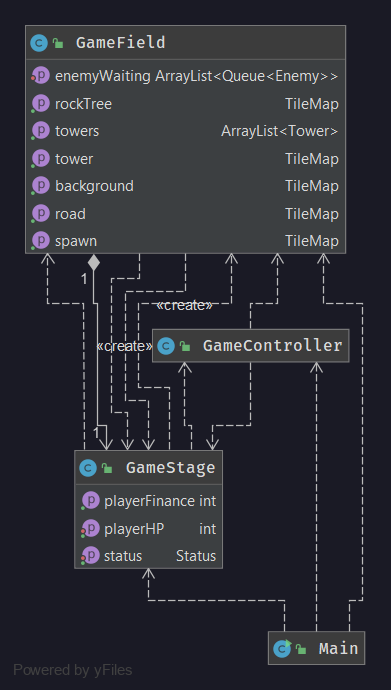
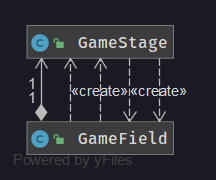
**I. Tổng quát về chương trình**

1. Khi chương trình được bắt đầu, main sẽ gọi đến *GameController*

2. Khi người chơi đã lựa chọn được màn chơi tại menu, một đối tượng *GameField* sẽ được tạo ra và load tất cả thông tin về màn chơi vào GameField (bao gồm thông tin về *Road*, *Tower*, *Spawn*, *Target*, *Enemy* wave, …)

3. Và ở đây *GameField* tạo ra một đối tượng *GameStage* để quản lý màn chơi và người chơi (Điều kiện thắng/thua, trạng thái của trò chơi, tài chính của người chơi, số mạng của người chơi, …)

*Sơ đồ mối quan hệ trong hàm core và quan hệ giữa GameStage và GameField*

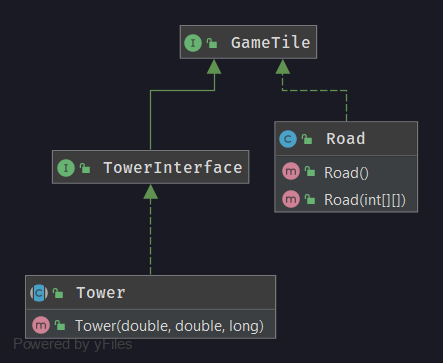
**II. Đối tương của trò chơi**

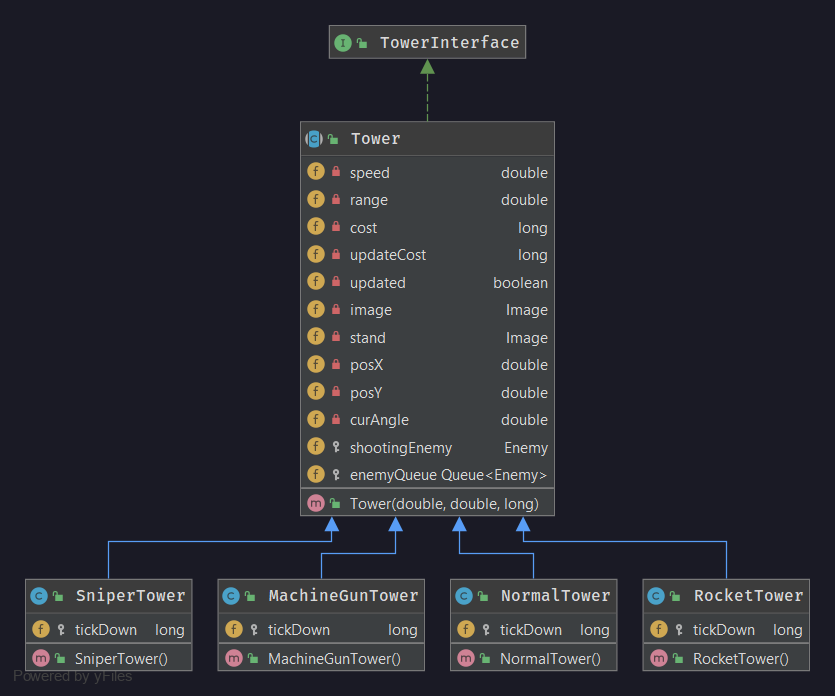
Trong chương trình, các đối tượng của trò chơi được chia ra làm 2 kiểu đối tượng nhỏ, bao gồm

* Đối tượng Tĩnh (*Immoveable*)
* Đối tượng Động (*Moveable*)

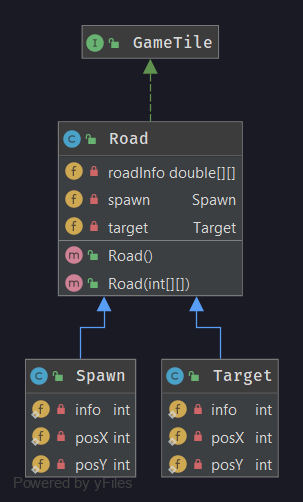
Cụ thể, trong lớp đối tượng Tĩnh (có interface là *GameTile*):

* ***Tower***
* ***Road***



*a, Lớp Tower Trừu Tượng*

* Lớp *Tower* trừu tượng chứa những function cơ bản của một *Tower*, và là hàm cha của các loại *Tower* con cụ thể.
* Các lớp *NormalTower*, *MachineGunTower*, *RocketTower*, *SniperTower* là hàm con được kế thừa các thuộc tính và function của lớp *Tower* Trừu Tượng

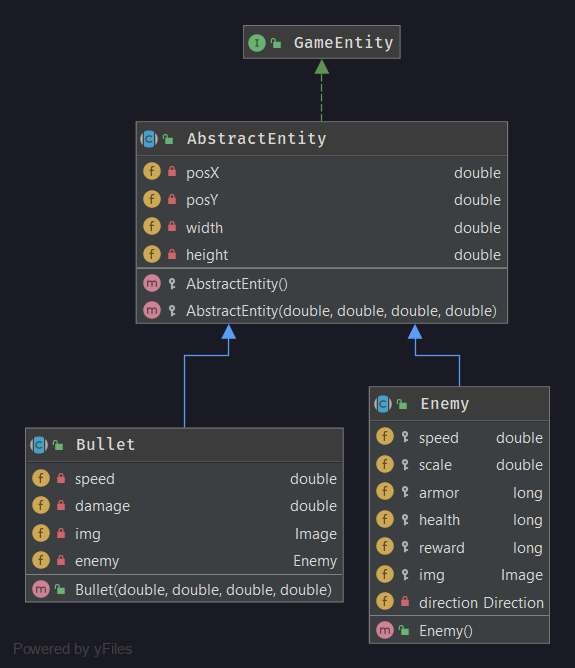
*b. Lớp Road*

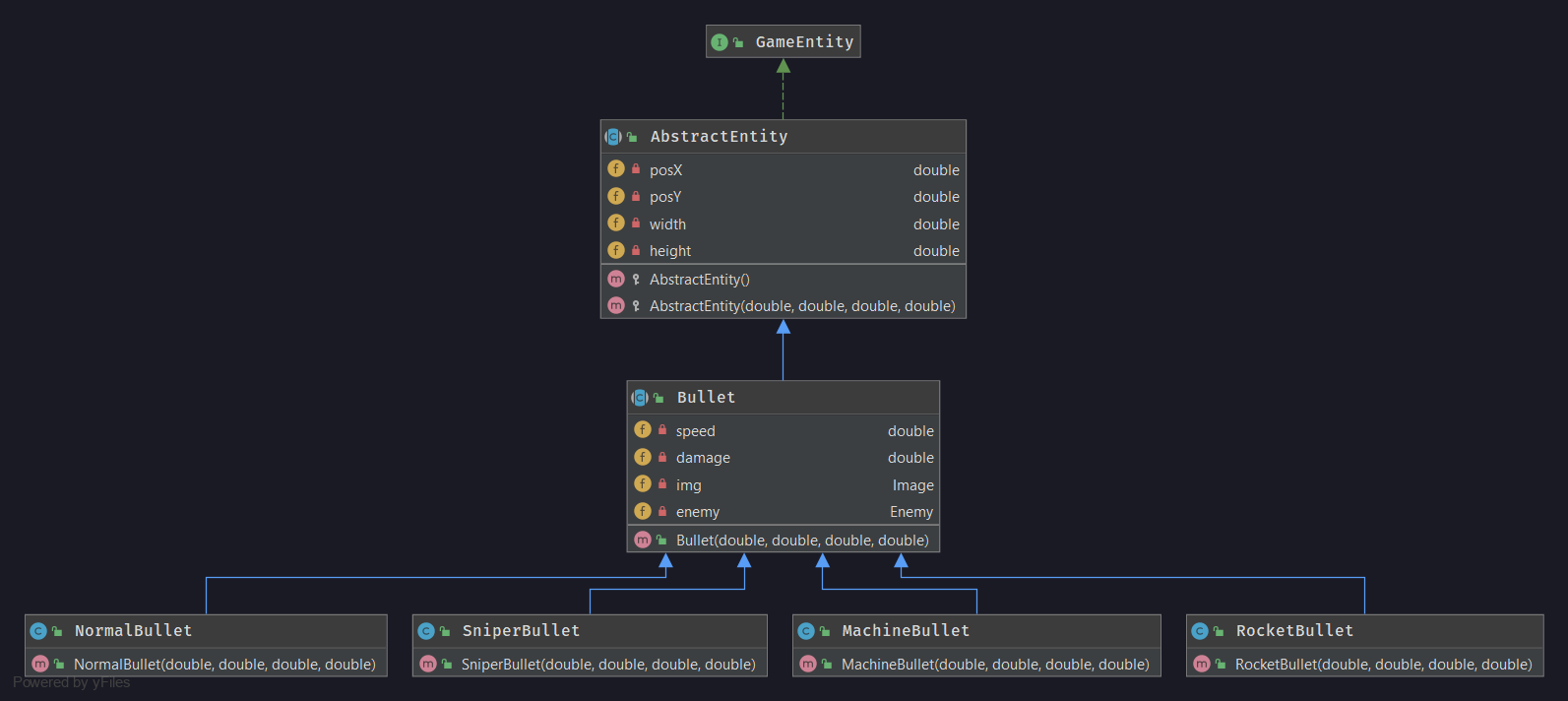
- Lớp Road được thừa kế interface *GameTile*, chứa những thông tin cơ bản về đường đi trong màn chơi (gồm *Spawn*, *Target* và thông tin của đường (*roadInfo*))

- Các lớp *Spawn* và *Target* là những hàm con được kế thừa các function và thuộc tính của Road

Trong lớp đối tượng Động (có interface là *GameEntity*) bao gồm:

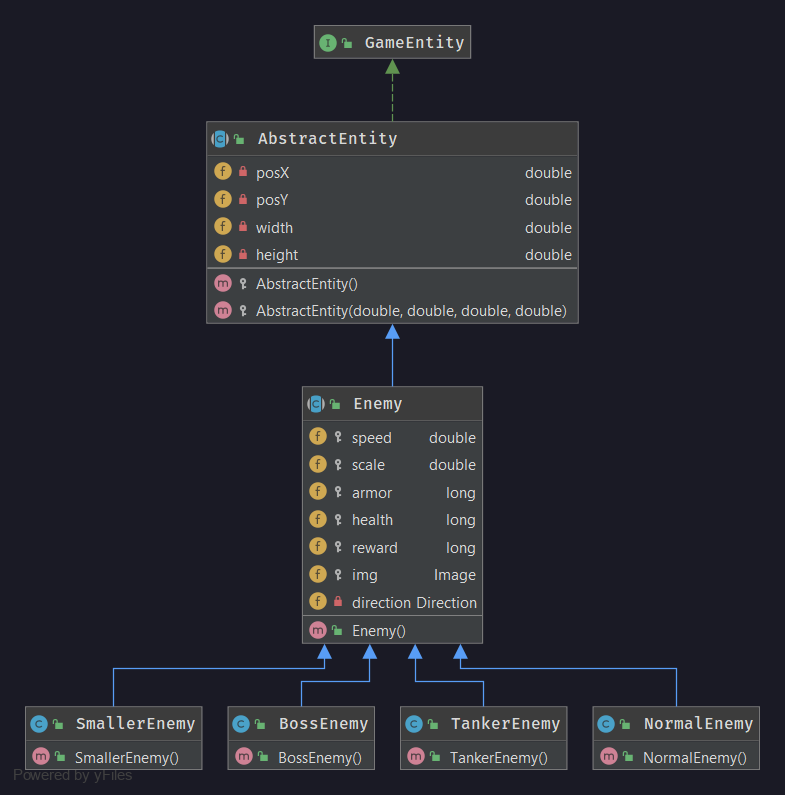
* ***Bullet***
* ***Enemy***





a*. Lớp Bullet trừu trượng*

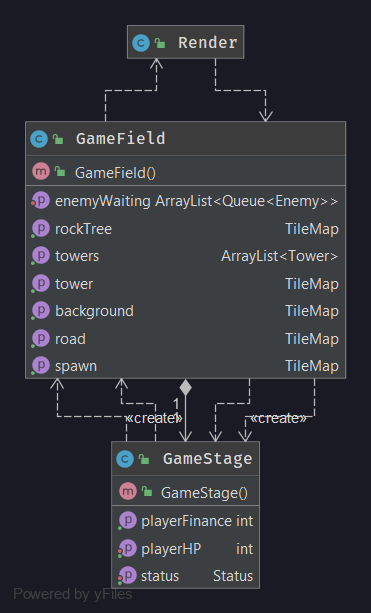
* Lớp Bullet trừu tượng được kế thừa thuộc tính từ Lớp (trừu tượng) AbstractEntity. Và là lớp chứa những thuộc tính và các function cơ bản của loại hình bullet.
* Kế thừa từ Bullet là các lớp con NormalBullet, SniperB ullet, RocketBullet, MachineBullet cùng chung hầu hết các chức năng, nhưng có thuộc tính khác nhau.

*b. Lớp Enemy trừu tượng*

* Lớp trừu tượng *Enemy* cũng kế thừa các thuộc tính của đối tượng tĩnh từ *GameEntity* và *AbstractEnity*
* Đây là lớp chứa những chức năng cơ bản của các loại hình enemy khác nhau.
* Các lớp con *NormalEnemy*, *TankerEnemy*, *SmallerEnemy*, *BossEnemy* đều được kế thừa các chức năng từ lớp trừu tượng *Enemy*, song các thuộc tính thì khác nhau.

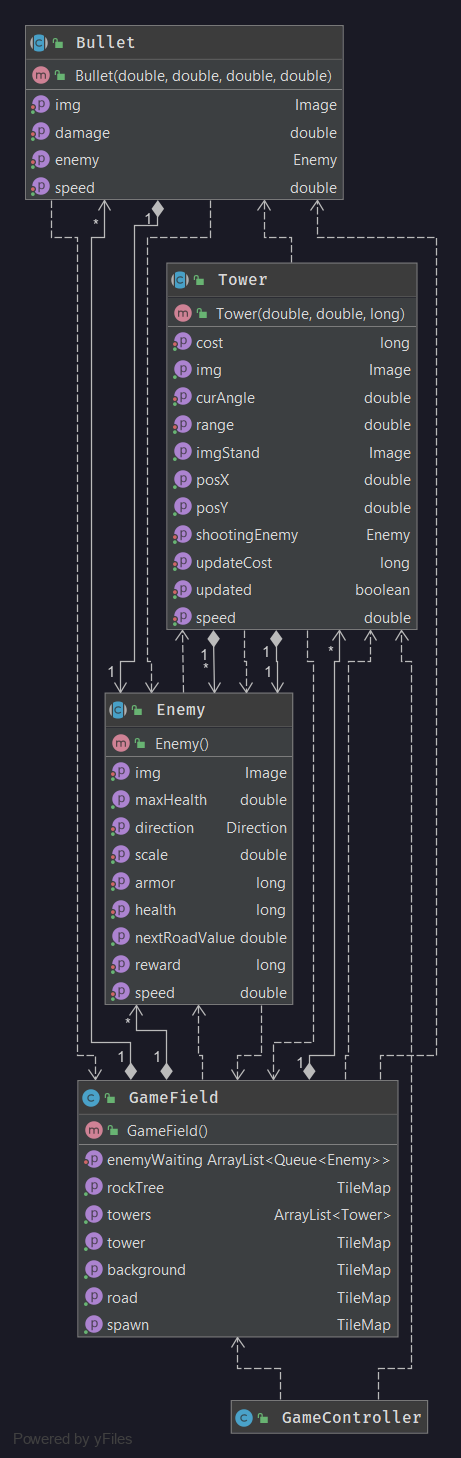
**III. Các hàm chức năng quan trọng**

***1. Render map***

****- Khi đối tượng *GameField* đã được nạp dữ liệu của màn chơi.

- Hàm *draw* trong *GameField* sẽ có nhiệm vụ render ra màn chơi đó (*render* các đối tượng *enemy*, *tower*, *bullet*, các layer của map như: *background*, *road*, …)

***2. Đặt tower***

***//TODO: Chú Hùng miêu tả ở đây nhé***

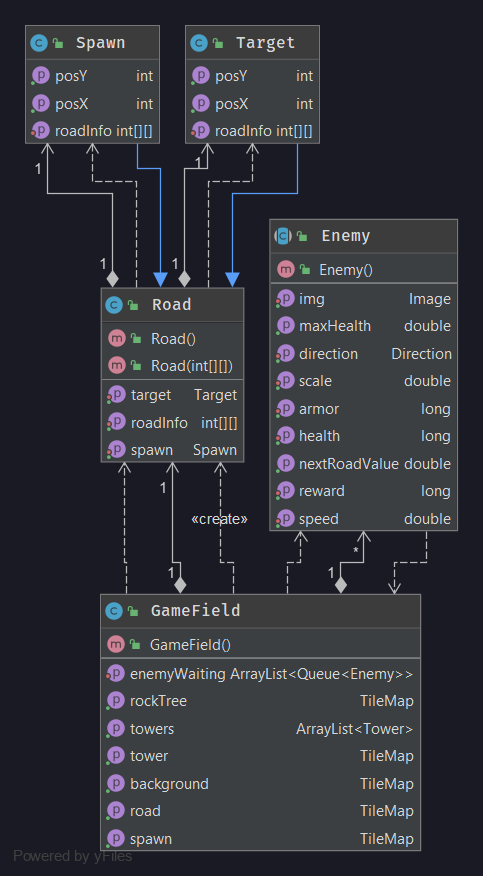
***//****Phía dưới này toàn dấu enter thôi :v*

**3. Hành vi “shoot” của *Tower*** *(Vẫn diagram trên)*

* Trong mỗi đối tượng tower có chức năng tính khoảng cách với các đối tượng Enemy được truyền vào. Ở đây hành vi “shoot” sẽ được diễn ra khi khoảng cách của đôi tượng nằm trong phạm vi tấn công của đối tượng *tower* đang xét.
* Giả sử enemy được truyền vào nằm trong phạm vi tấn công, và *tower* đang gọi tới không tấn công bất cứ enemy nào khác (*shootingEnemy* == null). Thì function *update* của *tower* sẽ gán đối tượng cần tấn công chính là enemy được truyền vào (*shootingEnemy* = *enemy*).
* Khi đấy, chức năng shoot của *tower* sẽ tạo ra một (và nhiều) đối tượng Bullet tương ứng (với *tower*) có mục tiêu là *shootingEnemy*, tức là *enemy* vừa xét.
* Hàm update của *Bullet* đảm bảo sẽ gây sát thương lên *enemy* ngắm tới và đảm bảo *Bullet* sẽ bị huỷ khi chạm tới *enemy.*

**4. Tìm đường của Enemy**

* Ở đây thuật toán tìm đường chỉ dừng lại ở việc thuật toán có trọng số đơn giản. Đối tượng Road cung cấp thông tin của đường đi, khi nạp thông tin vào Road, hàm tạo của đối tượng đã tự gọi hàm *setRoadInfo* với mục đích là tính khoảng cách tại các ô trên bản đồ đến *Target* của màn chơi (đây chính là trọng số của thuật toán)
* Mỗi đối tượng enemy sẽ có 4 hướng để di chuyển (UP, DOWN, RIGHT, LEFT)
* Dựa vào đây, mỗi khi enemy di chuyển, hàm *getNextRoadValue* sẽ liên tục được gọi (với mọi đối tượng Enemy) để trả về thông tin của đường (==0 là không thể di chuyển, !=0 là có thể di chuyển)
* Mỗi khi enemy gặp chướng ngại vật (tức là *getNextRoadValue* == 0, hướng đi phía trước không thể di chuyển tiếp), hàm *findDirection* sẽ được gọi để so sánh 2 trọng số (nằm ở 2 phía còn lại, VD: enemy đang “UP” và gặp chướng ngại vật, *findDirection* sẽ xét trọng số của ô nằm bên hướng “LEFT” và “RIGHT”), nếu trọng số nhỏ hơn, tức là enemy phải di chuyển theo hướng đó (tức là khoảng cách đến *Target* nhỏ hơn).
* Enemy sẽ xuất phát từ Spawn và kết thúc tại *Target*, hàm *hitTarget* đảm bảo khi *enemy* chạm tới *Target* thì người chơi sẽ bị trừ HP và đồng thời xoá đối tượng *enemy* đó.

******