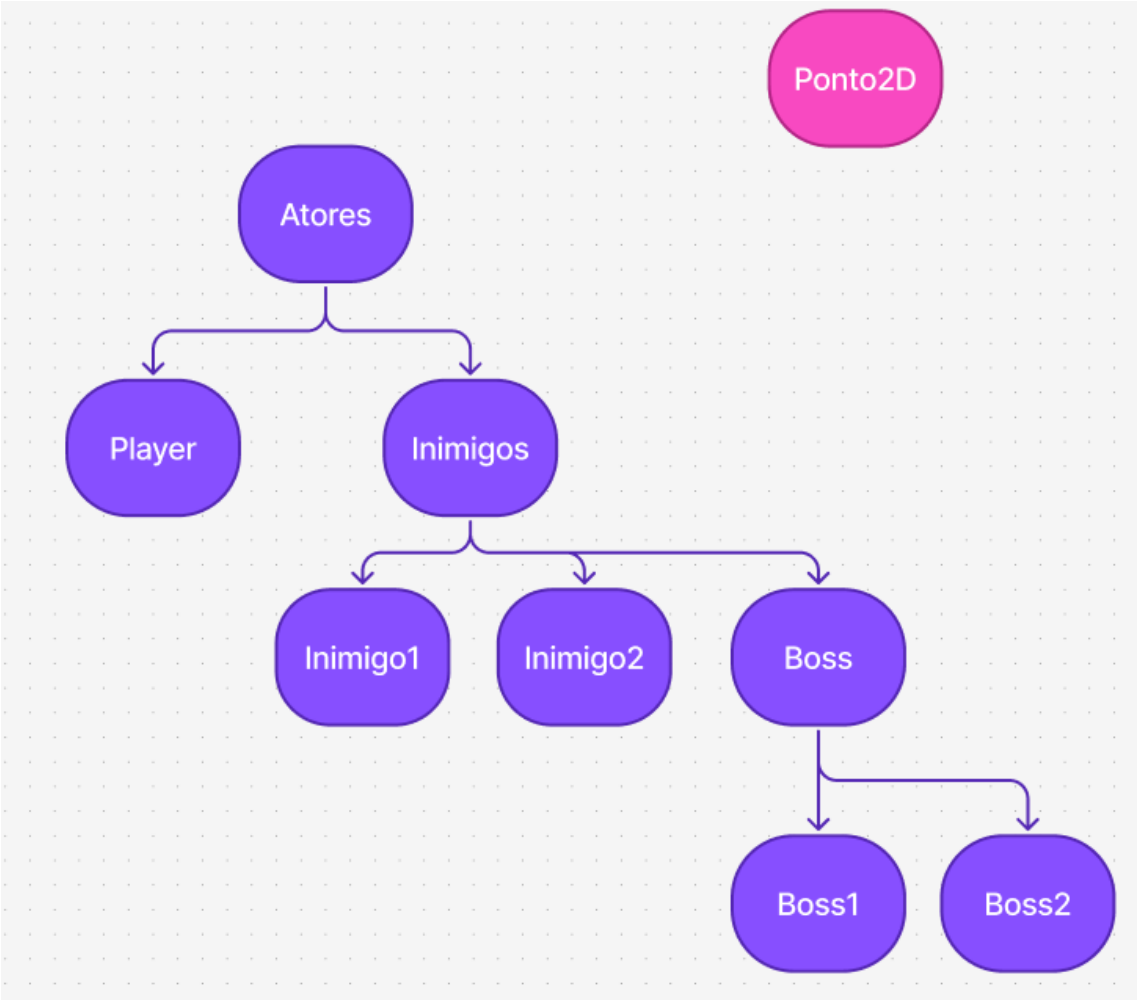
2. Estrutura das classes

Ponto2D

É uma classe pública que possui os atributos Coordenadas X e Y e suas respectivas velocidades, além de possuir métodos Setters e Getters, por si só não é muito útil, porém ela será incorporada em outras classes por meio da composição.

Atores e seus derivados



Atores: Classe pública e abstrata que define as características essenciais do player e inimigos, tendo como atributos:

Ponto2D ponto: posição e velocidade do ator.

LinkedList<Projetil> listaProjeteis: lista onde são armazenados os projéteis que ele dispara.

boolean explodindo: indica se o ator está no estado de explosão.

double inicioExplosao e double fimExplosao: tempos que controlam o início e fim da explosão.

double raio: raio do ator.

long proxTiro: tempo mínimo entre tiros.

Está classe também possui os métodos: colision e o abstract dispara() que são responsáveis por implementar as colisões, e cada classe diferente criar a sua própria maneira de disparar,

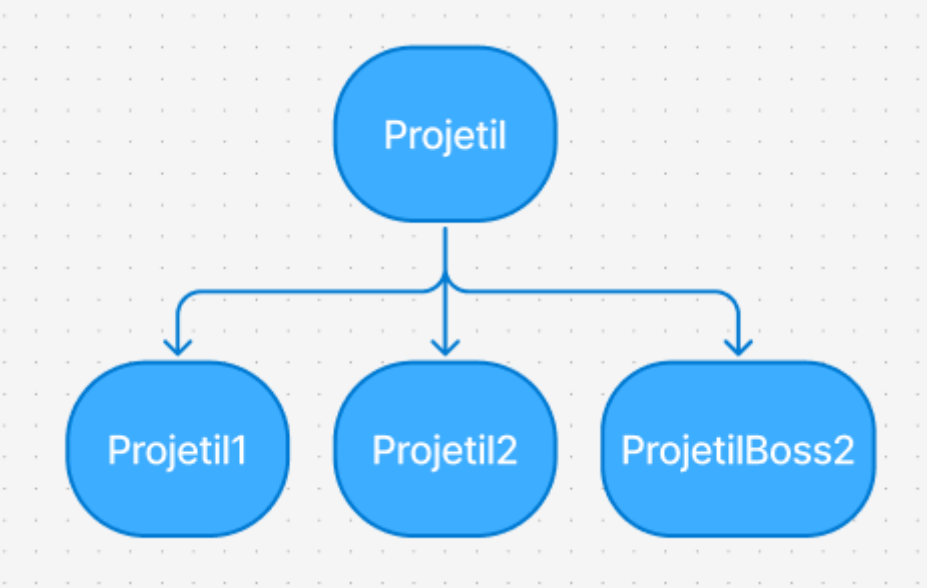
Player e Inimigos: São classes públicas que estendem atores, sendo que inimigos é uma classe abstrata, adicionando os sus próprios atributos.

Inimigo 1 e 2: São classes públicas que estendem Inimigos, implementando a sua própria movimentação e métodos de dispara.

Boss: É uma classe que realiza a extensão de Inimigos, definindo os atributos de vida e vidainicial.

Boss 1 e 2 As duas classes são extensões de Boss, mas implementam os seus próprios dispara() e atualizaestado(), fazendo com que os chefes se diferenciem.

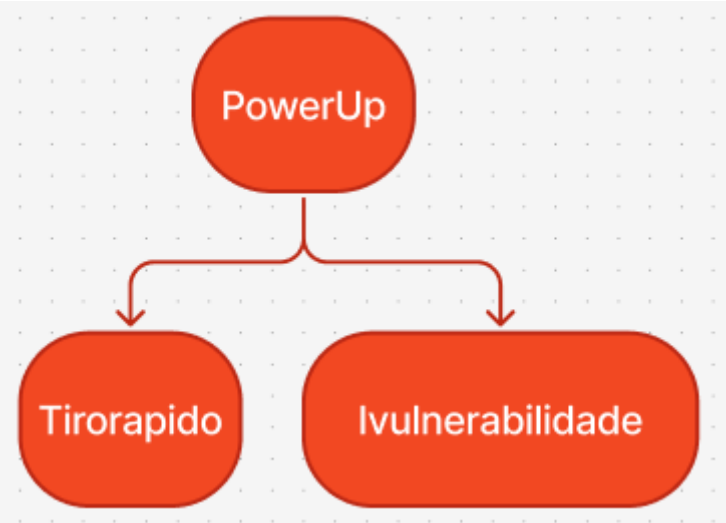
Projetil e seus derivados



Projetil: É uma classe pública e abstrata que serve como base para futuros projéteis de inimigos.

Projetil1 Projetil2 e ProjetilBoss2: São classes públicas utilizadas para criar os projéteis do inimigo1, inimigo2 e o projétil especial do Boss2.

PowerUp e seus derivados



PowerUp: Classe pública e abstrata que irá definir as principais características dos powerups.

Tirorapido e Ivulnerabilidade: Classes públicas que são extensões de PowerUp e implementam a funcionalidade dos poderes.

5. Boss2

A classe Boss 2 é uma extensão de Boss (que Boss em si é uma extensão de inimigos, e inimigos uma extensão de atores), portanto o Boss2 possui diversos atributos como os de atores que são: “Ponto2D ponto” para representar a sua posição no jogo, uma lista ligada de projéteis chamada “LinkedList<Projetil> listaProjeteis” para criar e remover os tiros da tela, “double inicioExplosão, boolean explodindo e double fimExplosão” que são atributos para controlar a explosão que o Boss irá gerar quando ele morrer. Além de seu “raio” demonstrando o seu tamanho e o “proxTiro” utilizado para criar um intervalo de tempo entre os disparos do chefe. Já seus atributos de inimigos são “double angulo” que representa a direção em que o chefe está olhando, e o “double vr” que é a velocidade de rotação do Boss. Seus atributos de Boss são “int vida” e “double vidainicial” utilizadas para representar a quantidade de tiros que faltam para o chefe ser morto e a atualização da sua barra de vida. Boss2 não possui nenhum atributo exclusivo da sua classe, apenas métodos.

public void Boss2 (double x, double y, double vx, double vy, double angulo, double vR, LinkedList<Projetil> listaProjeteis, int vida)

Para chamar o Boss2, basta instancia-lo com seu construtor que irá atribuir os valores passados como parâmetros aos seus atributos por meio do método super, o raio recebe um valor fixo de 27 e o atributo vidainicial recebe o valor da vida que foi passada como parâmetro.

Public void@Override desenha(long currentTime)

Tem como parâmetro o currentTime que é utilizado para determinar o tempo da explosão ao matar o chefe, esse método é o responsável por gerar todas as imagens relacionadas ao chefe na tela, então ela desenha o formato do Boss2, muda a sua cor ao receber dano para informar ao player que os seus projéteis são efetivos, e chama o método desenhabarra() para aparecer a sua barra de vida na tela, e também desenha a explosão quando o chefe for morrer.

@Override public void desenhabarra()

É desenhado um retângulo verde representando a vida do Boss2, e a largura do retângulo diminui baseada na vida atual do chefe em comparação com a vida inicial.

@Override public boolean Colision(Linkedlist<Projetil> projeteis, long currentTime, double c)

Recebe como parâmetros uma lista ligada de projeteis que irá ser os projéteis do player na Main, o currentTime é usado para determinar o tempo das explosões, e o double c é responsável por manipular o tamanho da “hitbox” da colisão por meio de seu valor.

Este método também tem a propriedade de diminuir a vida do chefe quando entrar em contato com o projétil do player inimigo, e caso sua vida chegue em 0, inicia uma explosão retornando o estado explodindo.

@Override public void dispara(long currentTime, double PlayerY)

É o método que possibilita o chefe atirar os seus próprios projeteis ao detectar que é hora de realizar o próximo tiro, quando o momento chegar, o chefe irá escolher entre dois ataques se o player estiver em uma coordenada Y menor que a do Boss2.

O primeiro ataque tem 90% de chance de ocorrer que dispara 3 projéteis vermelhos semelhantes ao do Inimigo1 em ângulos diferentes, já o segundo tiro que dispara dois projéteis laranjas com uma propriedade especial de “rebater” nos cantos da tela, essa propriedade é gerada pela inversão das suas velocidades X ou Y ao atingir uma borda.

@Override public boolean atualizaEstado(long DeltaTime, long currentTime, double playerY, LinkedList<Projetil> ProjetilInimigo)

É o método principal que chama os outros métodos além de desenha, além de implementar a movimentação horizontal do Boss, que tem a sua velocidade horizontal invertida toda vez que “bate” em uma das bordas da tela, este método também chama o colision(),se o chefe não estiver com o estado explodindo e também chama o dispara(), caso o Boss seja explodido retorna false para futuramente ser removido do jogo, caso contrário retorna true.