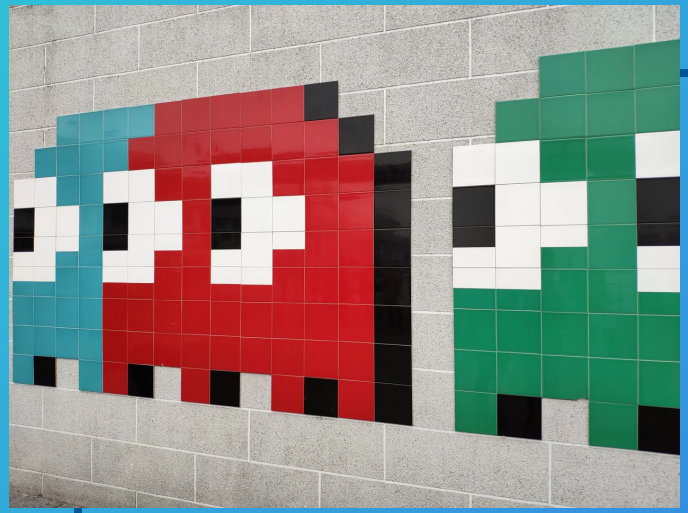
#AVALIAÇÃO PACMAN

programação orientada a objetos

João Carlos Lima
joao.clsilva@sp.senac.br

INTRODUÇÃO

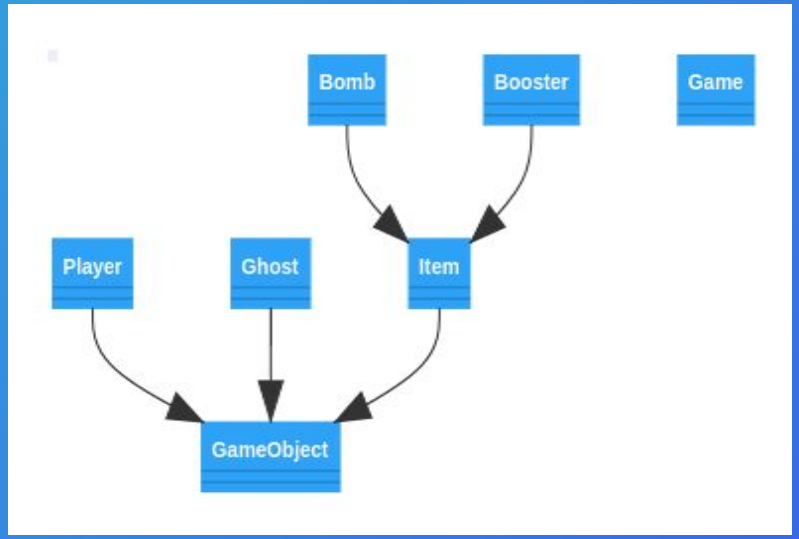


O objetivo deste projeto é criar a estrutura inicial de classes para o jogo **Pac Man**

DIAGRAMA DE CLASSES

02 PONTOS

O diagrama ao lado representa as classes e relacionamento entre elas. Crie o código java que representa o diagrama a seguir.



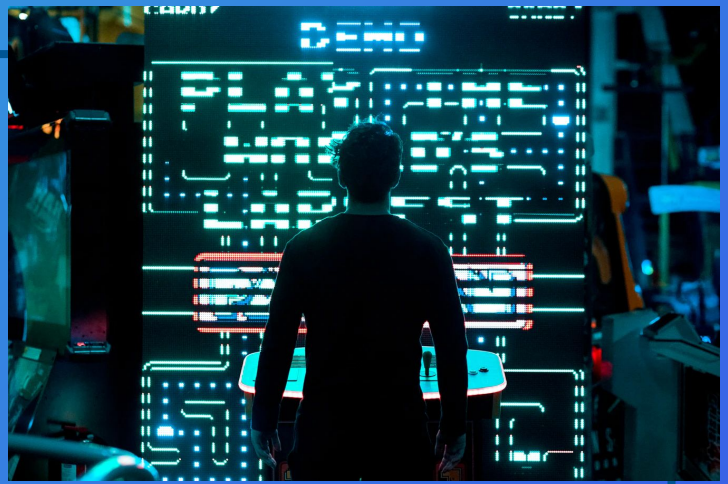
CLASSE GAMEOBJECT

02 PONTOS

Essa classe representa todos os elementos que são exibidos no jogo.

Os atributos desta classe são:

- A posição x do elemento, representada por um número inteiro positivo.
- A posição y do elemento, representada por um número inteiro positivo.
- O tamanho da tela em pixels, representada por um número inteiro positivo.



Essa classe deve ter um construtor padrão e um construtor que recebe a posição x e y do elemento.

Essa classe deve ter os métodos *get* e *set*.

O construtor e os métodos *sets* devem validar os parâmetros

CLASSE PLAYER

01
PONTO

Essa classe representa o personagem que é controlado pelo jogador. Player é um GameObject.

Player deve conter a direção (em graus) em que o personagem está se movimentando. Player tem quantidade de vidas.
O Player pode estar ou não invencível.

Essa classe deve ter um método que verifica se o player pode se mover. Esse método deve considerar a posição e a direção do jogador e retornar verdadeiro, caso a nova posição esteja dentro da tela ou falso, caso esteja fora da tela. A nova posição a ser analisada é a posição atual, acrescida de 10 pixels na direção que o player aponta.

Essa classe deve conter um método que move o player, verificando antes se ele pode se mover.

Essa classe deve ter um construtor padrão e um construtor que recebe a posição x, y e a direção do player. Essa classe deve ter os métodos *get* e *set*. O construtor e os métodos *sets* devem validar os parâmetros.

CLASSE GHOST

01 PONTO

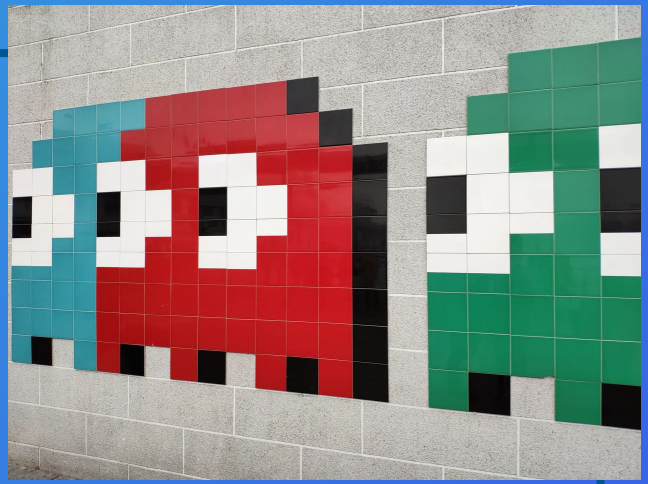
Essa classe representa o inimigo do jogo. Ghost é um GameObject.

A exemplo da classe Player, a classe Ghost deve ter os métodos para verificar se ele pode se mover e o método que o movimenta.

O método que movimenta o ghost deve mudar a direção aleatoriamente.

Essa classe deve ter um construtor padrão e um construtor que recebe a posição x, y e a direção do Ghost.

Essa classe deve ter os métodos *get* e *set*. O construtor e os métodos *sets* devem validar os parâmetros.



CLASSE ITEM

01
PONTO

Essa classe representa o item do jogo. Item é um GameObject.

O Item pode estar visível ou não.

Essa classe deve ter um construtor padrão e um construtor que recebe a posição x, y.

Essa classe deve ter os métodos *get* e *set*. O construtor e os métodos *sets* devem validar os parâmetros.

CLASSE BOOSTER

01 PONTO

Essa classe representa o item Booster do jogo. Booster é um Item.

O Booster faz com que o Player fique invencível por um determinado tempo em turnos. Esse tempo é variável.

Essa classe deve ter um construtor padrão e um construtor que recebe a posição x, y e a duração.

Essa classe deve ter os métodos *get* e *set*. O construtor e os métodos *sets* devem validar os parâmetros.

CLASSE BOMB

01 PONTO

Essa classe representa o item Bomb do jogo. Bomb é um Item.

A bomba remove uma vida do player.

Essa classe deve ter um construtor padrão e um construtor que recebe a posição x, y.

Essa classe deve ter os métodos *get* e *set*. O construtor e os métodos *sets* devem validar os parâmetros.

CLASSE GAME

02 PONTOS

Essa classe deve conter o método main para testar as demais classes.

No método main, crie um Player e 4 Ghosts.

Crie alguns Itens dos diversos tipos para testar seu funcionamento.

Mova o Player e os Ghosts até o final do jogo (término das vidas).

Ao mover os elementos, verifique a colisão e aplique as regras:

- Se Player colide com o Ghost perde uma vida.
- Se Player colide com Bomb perde uma vida.
- Se Player colide com Booster fica invencível, isto é, pode colidir sem perder vida durante a duração do Booster.

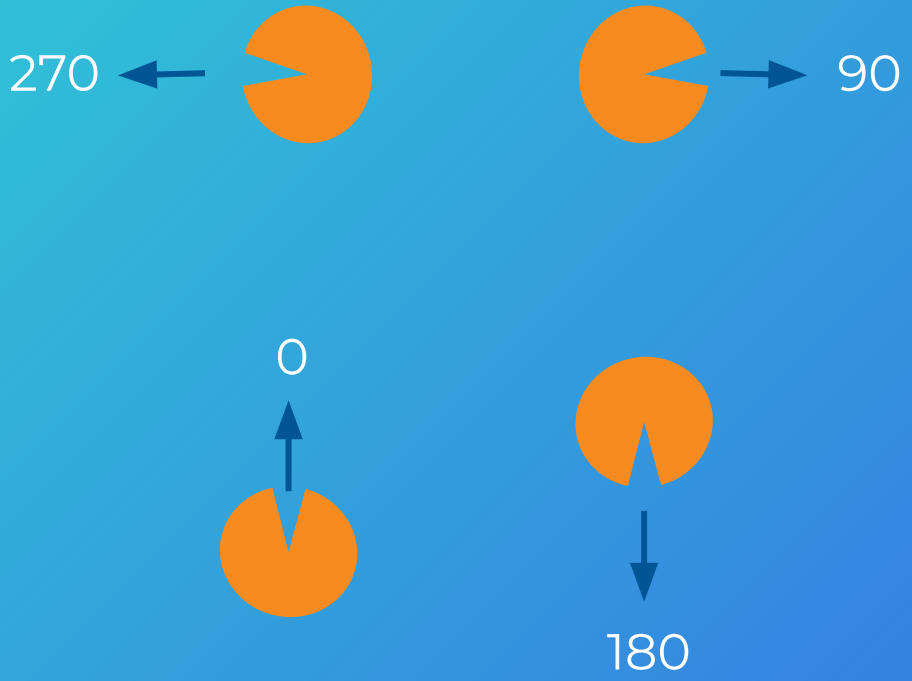
INSTRUÇÕES FINAIS

Você é livre para criar outras classes e métodos, mas fique atento para manter a coesão.

Você deve apresentar a aplicação funcionando até o término da aula.

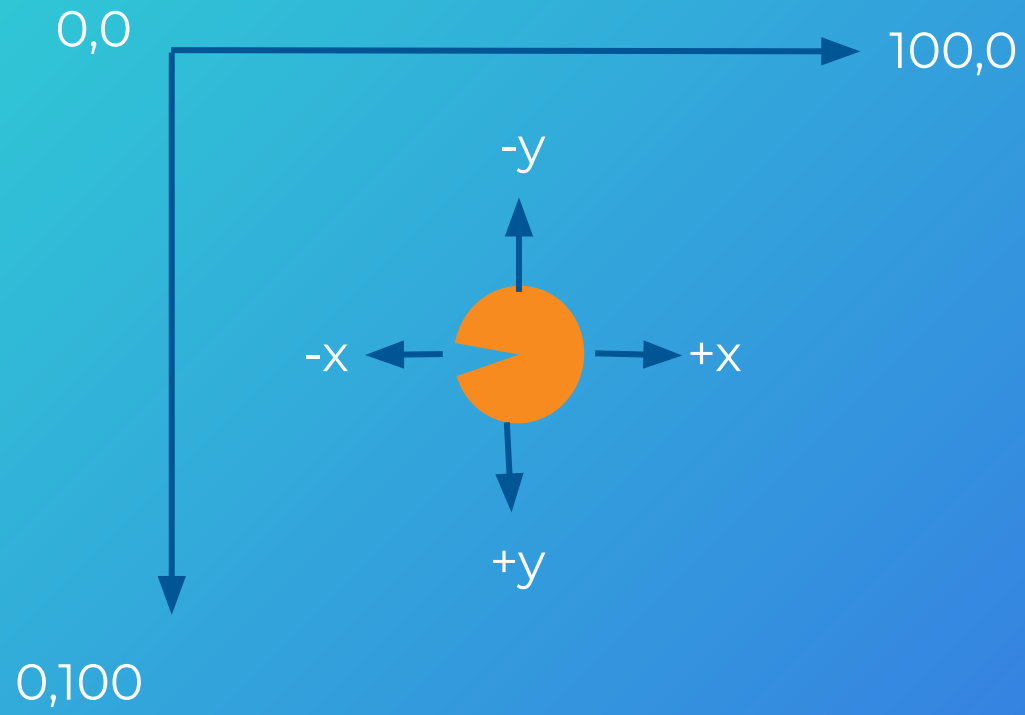
Existe uma classe que representa a interface gráfica do jogo. Você pode utilizar para visualizar o comportamento dos objetos.
github.com/joaocarloslima/pacman

INSTRUÇÕES FINAIS



O Player pode se mover apenas em 4 direções, de acordo com o valor do ângulo armazenado no atributo direção.

INSTRUÇÕES FINAIS



O plano cartesiano de telas funciona um pouco diferente do plano cartesiano matemático.

OBRIGADO

João Carlos Lima
joao.clsilva@sp.senac.br