***Importante***: Para cada item abaixo deve ser copiado trechos do código que cumprem o requisito e explicado, se não for aparente, o porquê o requisito é cumprido. Sejam bem explícitos. Deve ser indicado também o arquivo da classe em que está o trecho do código. Eu avaliarei o código do Github a partir desse documento para confirmá-lo e também para detectar possíveis erros. **Quem não seguir o que está indicado aqui, não terá o projeto avaliado e perderá a atividade.**

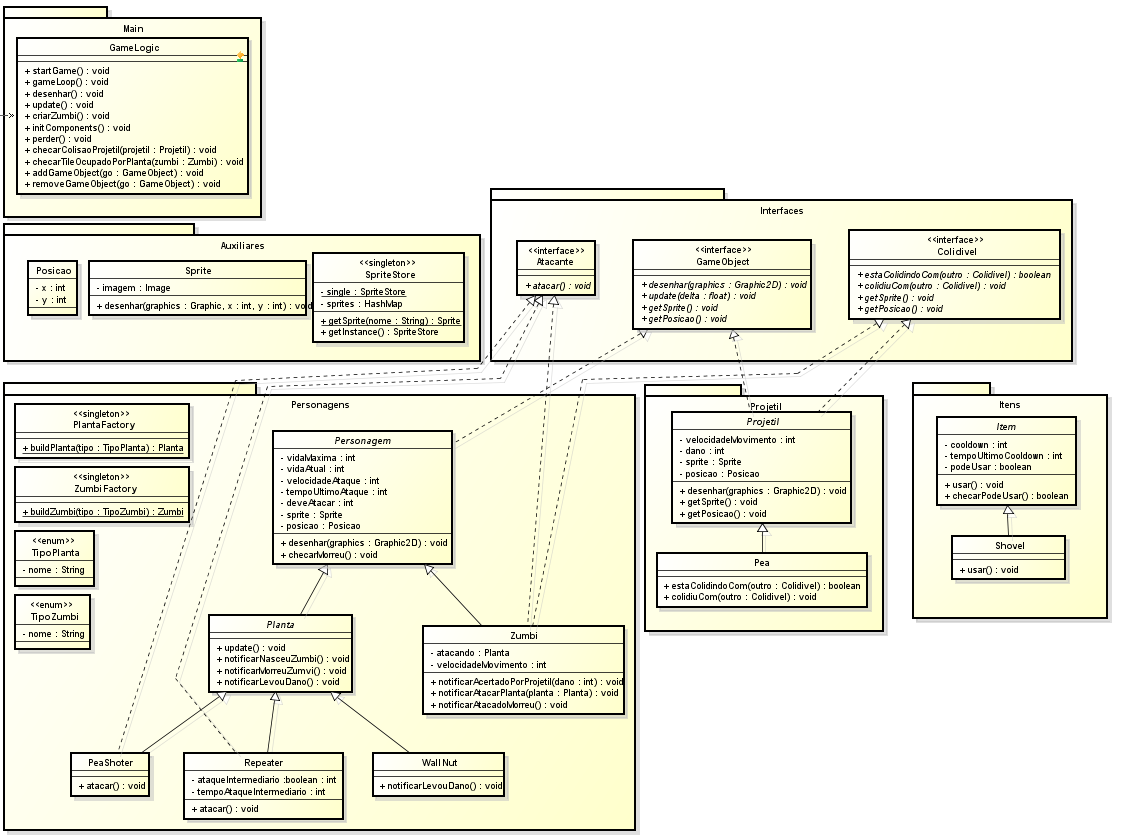
**Usar um novo repositório!**

**Entrega final: 26 de novembro via GitHub. Prova oral sobre o código ainda a ser definida.**

**Entrega parcial: 19 de novembro via GitHub. Apenas Diagrama de Classes, interfaces e classes abstratas no branch general.**

**Requisitos Gerais**

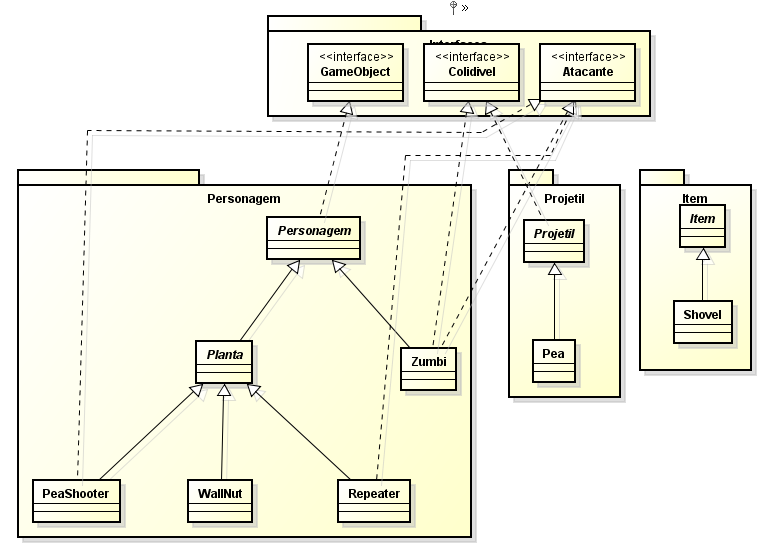
***Todos os atributos e funções membros devem estar relacionados a classe***

1. Pelo menos 4 atributos
2. Pelo menos 4 funções membros sem incluir get e set
3. Diagrama UML completo (obrigatório salvar também o png do diagrama no gitHub)

**Requisitos de implementação**

1. Todas as classes concretas devem vir de interfaces ou classes abstratas. Pelo menos três hierarquias de classes. Uma das hierarquias deve ter três níveis. Exemplo: Personagem >> Ciborgue >> Robocop; Class Arma (interface) >> Beretta93R

Diagrama com principais hierarquias. Hierarquias Personagem, Projetil e Item. Personagens possui três níveis.



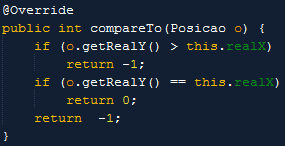
1. Ao menos três interfaces. A terceira interface deve ser uma interface que liga duas hierarquias como no exemplo da interface **corredor** (Figura 1).

Ver diagrama com principais hierarquias. Interfaces GameObject, Colidivel, Atacante. GameObject e Colidivel unem duas hierarquias (ambos unem Personagem a Projetil).

1. Usar a interface **Comparable** e sobrescrever o método **compareTo** em pelo menos uma hierarquia

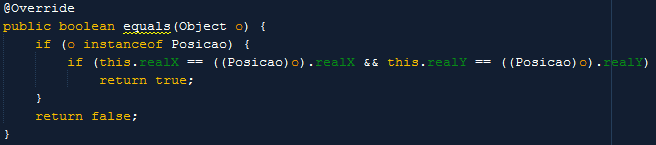
Override do método compareTo na classe auxiliar Posição, que compara se a altura entre duas posições.

C:\Users\Felipe\Desktop\Requesitos\Comparable.PNG



1. Sobrescrever **equals** para de Object

Todas as classes possuem o método equals. Exemplo em Posição:

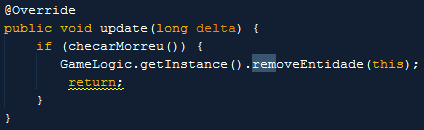
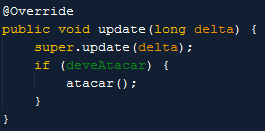


1. Todas as hierarquias devem ter classes Concretas, e em uma das hierarquias, três classes Concretas relacionadas: Exemplo Servico >> ServicoStream >> (Netflix, HBOStream, AmazonPrime, NowTv). Em uma pesquisa de 10 segundos: <http://www.tomsguide.com/us/pictures-story/620-top-online-streaming-video.html>

Hierarquia Personagens possui três classes concretas derivadas de planta (PeaShooter, WallNut, Repeater) e a classe Zumbi.

1. Sempre usar o super para o máximo de reaproveitamento de código

Vários lugares do código. Exemplo em Peashooter que chama no update o super.update de Planta que por não estar implementado chama o update de Personagem, se a planta morreu não continua o update especifico e retira ela da lista de entidades.



1. Atributos static e const static

Exemplo de static em ZumbiFactory.

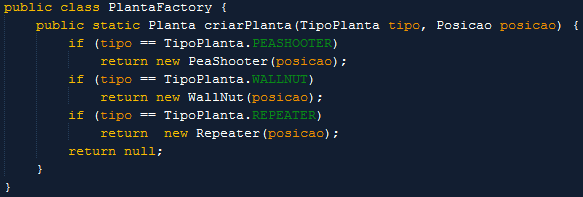
C:\Users\Felipe\Desktop\Requesitos\StaticZumbisCriados.PNG

Exemplo de const static em WallNut.

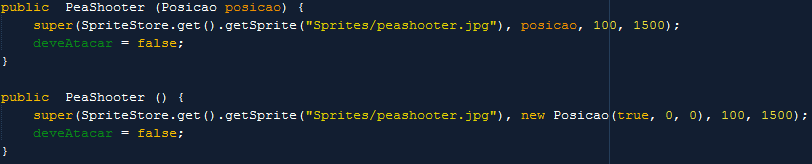
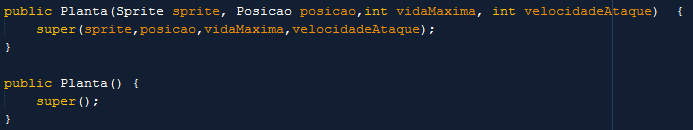
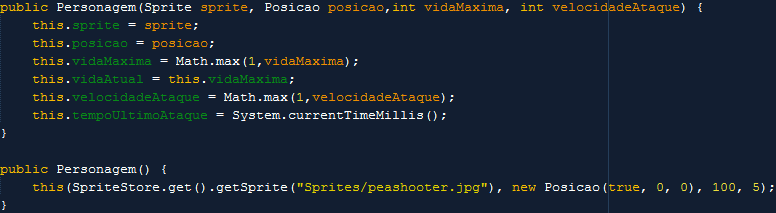
C:\Users\Felipe\Desktop\Requesitos\ConstStaticVidaMaxima.PNG

1. Método static

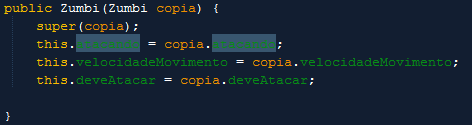
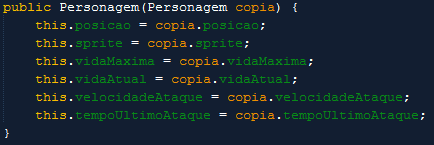
Exemplo do método criarPlanta em PlantaFactory.



1. Construtores em todas as classes, e dois na hierarquia principal. Sempre validar os dados em todas as classes

Dois construtores da hierarquia principal, todas as outras classes tem construtores (menos as Factorys, por não precisarem)

1. Construtor cópia em uma das hierarquias

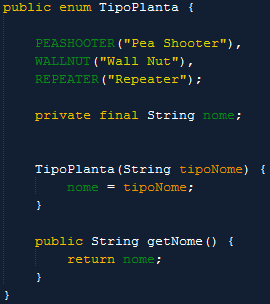
Construtor de copia do Zumbi chamando o construtor de copia do Personagem.

1. ArrayList

Class GameLogic

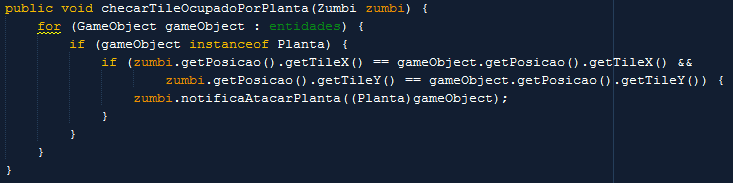
C:\Users\Felipe\Desktop\Requesitos\AL.PNG

1. ENUM

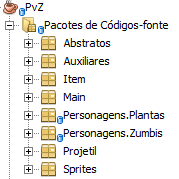


1. Usar o **instanceof** no main junto com as classes concretas. Para uma da classe concreta identificada, chamar um método dessa classe e fazer uma ação;

Na classe GameLogic, que é onde ocorre a logica do jogo. (Classe GameLogic é a única coisa chamada dentro do main)

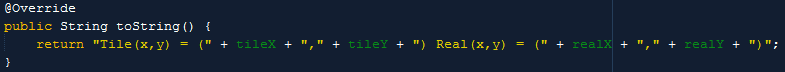


1. Dividir o projeto em pacotes

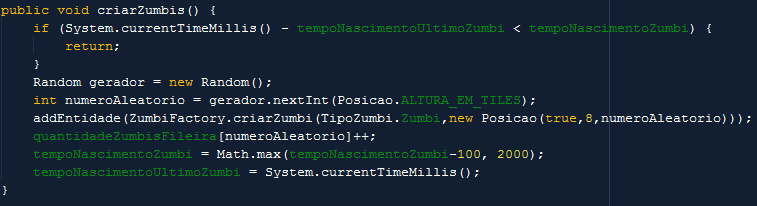


1. Sobrescrever para todas as classes o método toString

Exemplo na classe Posicao.



1. Usar a classe Random do pacote java.util (java.util.Random). Nota: deve ser usado conforme o contexto do projeto. Se for usado em um método genérico sem relação com a classe e apenas para cumpri-lo, esse requisito será desconsiderado.

Na classe GameLogic

1. No main o usuário deve fazer entrada via teclado e interagir com a aplicação. Opcional de bônus: pode ser usada a classe JOptionPane do pacote javax.swing. Vejam: showInputDialog e showMessageDialog.