SISTEMA ZOID

Sistema de suporte e assistência para jogos de RPG

---------------------------------

**André Luiz Braga, D.Sc.**

+55 21 98854-1509

Pos-Doc Researcher

--------------------------------

Computer Science and Robotics

***IME – Militar Engineering Institute, Brazil***

Doctor in Computer and Systems Engineering

Knowledge, Information and Database Systems

***COPPE – Rio de Janeiro Federal University – Brazil***

[andre.luiz.braga@terra.com.br](mailto:andre.luiz.braga@terra.com.br" \t "_blank)

---------------------------------------------------

Former IBM Senior Certified IT Architect – IBM Brazil/IBM-US

[1 Descrição 4](#_Toc524184803)

[2 MODULOS 5](#_Toc524184804)

[2.1 MODULO ELROS – Sistema geral 5](#_Toc524184805)

[2.2 MODULO SAURON - Jogos e Partidas 5](#_Toc524184806)

[2.2.1 Criar e gerenciar Jogos 5](#_Toc524184807)

[2.2.2 Criar e gerenciar Partidas 6](#_Toc524184808)

[2.2.3 Construção de Grupos 6](#_Toc524184809)

[2.3 Manipulação de AVATARES 7](#_Toc524184810)

[2.3.1 **MODULO ARAGORN** – Gerencia Geral de Avatares e Armazenamento de Avatares 7](#_Toc524184811)

[2.3.2 **MODULO ELROND** - Construção de Avatares Masculinos 8](#_Toc524184812)

[2.3.3 **MODULO GALADRIEL** - Construção de Avatares Femininos 9](#_Toc524184813)

[2.3.4 **MODULO RADAGAST** - Construção de Avatares Neutros 10](#_Toc524184814)

[2.3.5 **MODULO BALROG** - Construção de Avatares Criaturas (Monstros) 11](#_Toc524184815)

[2.4 Animais de Estimação -- 12](#_Toc524184816)

[2.4.1 **MODULO SHADOWFAX** – Criação de Animais de Estimação 12](#_Toc524184817)

[2.4.2 **MODULO PETSHADOW** – Cuidar de Animal de Estimação 13](#_Toc524184818)

[2.5 Modulo Itens de Jogo -- 14](#_Toc524184819)

[2.5.1 **MODULO GANDWISE** 14](#_Toc524184820)

[2.5.1.1 Gerencia e Armazenamento dos Itens 14](#_Toc524184821)

[2.5.1.2 Sacola de Itens 15](#_Toc524184822)

[2.5.2 **MODULO LEGOLAS** – Construção de Itens de Armas 16](#_Toc524184823)

[2.5.3 **MODULO TURIN**– Construção de itens de Valores 17](#_Toc524184824)

[2.5.4 **MODULO DWARF** – Construção de itens de Ferramentas 18](#_Toc524184825)

[2.5.5 **MODULO GANDALF** – Construção de itens Mágicos 19](#_Toc524184826)

[2.6 Localização e Mapas -- 20](#_Toc524184827)

[2.6.1 **MODULO TERRAMEDIA** 20](#_Toc524184828)

[2.6.1.1 Construção de Mapas 20](#_Toc524184829)

[2.6.1.2 Construção de Locais de Mapas – CidadeMapa 21](#_Toc524184830)

[2.6.1.3 Construção de Locais de Mapas –CavernaMapa 21](#_Toc524184831)

[2.6.1.4 Construção de Locais de Mapas – LagoMapa 22](#_Toc524184832)

[2.6.1.5 Construção de Locais de Mapas – RioMapa 22](#_Toc524184833)

[2.6.1.6 Construção de Locais de Mapas – TavernaMapa 23](#_Toc524184834)

[2.6.2 **MODULO CELEBORN** – Construção de Livros de Mapas 24](#_Toc524184835)

[2.6.3 **MODULO BEORN** – Localização de Avatares 25](#_Toc524184836)

[2.7 Modulo Estórias -- MODULO Bilbo 26](#_Toc524184837)

[2.7.1 Construção de Estórias 26](#_Toc524184838)

[2.7.2 Construção de Frases de Estórias 27](#_Toc524184839)

[2.7.3 Construção de Palavras de Frases 28](#_Toc524184840)

[2.7.4 Algoritmos Auxiliares de palavras 29](#_Toc524184841)

[2.7.5 Construção de Palavras de Frases 29](#_Toc524184842)

[2.7.6 Construção e gerencia de Livros de Estórias 30](#_Toc524184843)

[2.8 MODULO EOMER - Módulo Missões 31](#_Toc524184844)

[2.8.1 Construção de Missões e Armazenamento 31](#_Toc524184845)

[2.8.2 Construção e Armazenamento do Livro de Missões 32](#_Toc524184846)

[2.9 Módulo Encontros e Combates 33](#_Toc524184847)

[2.9.1 **MODULO BOROMIR** – Construção de Conversa 33](#_Toc524184848)

[2.9.2 **MODULO FARAMIR** – Construção de Combates 34](#_Toc524184849)

[3 NOTAS e GUIAS GERAIS 36](#_Toc524184850)

# Descrição

Este sistema, composto de vários módulos tem por finalidade a assistência e manipulação de jogos de RPG (Role Playing Games). Os módulos serão desenvolvidos por times separados mantendo a integração através dos paradigmas de polimorfismo e herança.

O sistema “Zoid” é composto dos seguintes módulos e funcionalidades

1. Gerencia geral do Sistema
2. Controle de Avatares
3. Controle de Combates/Encontros entre Avatares
4. Estórias que darão contexto ao jogo
5. Grupos de Avatares
6. Itens, como ferramentas, Armas, etc.
7. Controle dos Jogos e partidas
8. Mapas e Localização de Avatares
9. Missões
10. Animais de Estimação

Cada um desses subsistemas tem vários outros conjuntos de manipulação.

# MODULOS

## **MODULO ELROS** – Sistema geral

Este é o módulo principal do sistema que irá chamar a criação do painel principal de boas-vindas e iniciar a criação e execução dos jogos. Novas versões de GerenteZoid (GerenteZoidV1, GerenteZoidV2, etc..) poderão ser criadas.

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

public class GerenteZoidV1 extends GerenteZoid{

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public abstract class GerenteZoid {

MODIFICÁVEIS:

public class Zoid {

## **MODULO SAURON** - Jogos e Partidas

### Criar e gerenciar Jogos

Este módulo irá criar os jogos operados pelo sistema e que poderão se constituir de várias partidas, as quais serão criadas em outro módulo. Criar uma interface visual para criação inicial de um jogo entrando com as informações básicas e além disso:

1. Possibilitar inserir livros de missões, estórias e partidas.
   1. OBS: Futuramente (por outra equipe de projeto) o botão que insere, abrirá um outro painel para criação com detalhes.
2. Completar os detalhes da classe de Jogo e JogoArquivo
3. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas e vá chamando todos os métodos de escrita e leitura dos objetos do mapa (Veja nota geral a parte)

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

public class Partida {

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public abstract class Jogo {

public class JogoArquivo extends Jogo{

### Criar e gerenciar Partidas

Este módulo irá criar e gerenciar as partidas dos jogos operados pelo sistema. Criar uma interface visual para criação inicial da partida entrando com as informações básicas e além disso:

1. Possibilitar inserir os atributos da partida, como grupo de jogadores, missão, etc....
2. Completar os detalhes da classe Partida se necessário
3. Acrescentar métodos que julgue necessários para operar a partida.
4. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas e vá chamando todos os métodos de escrita e leitura dos objetos do mapa (Veja nota geral a parte)

NÃO MODIFICÁVEIS:

public abstract class Jogo {

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public class Partida {

### Construção de Grupos

Este módulo irá somente criar Grupos de Avatares através da escolha de jogadores na Fábrica de Avatares. Criar uma interface visual com a lista de jogadores para a escolha.

1. Criar novos atributos específicos que pertençam a todos os objetos dessa categoria se necessário.
2. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas.
3. Acrescente uma chamada para salvar e ler todos os jogadores que estiverem na lista do grupo.

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

public abstract class Avatar {

public abstract class FabricaAvatar {

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public class GrupoJogo {

## Manipulação de AVATARES

### **MODULO ARAGORN** – Gerencia Geral de Avatares e Armazenamento de Avatares

Este módulo irá criar um “Gerente Central” que irá armazenar todos os Avatares que serão futuramente escolhidos pelos construtores dos avatares e jogadores.

Criar uma interface visual para construção da Fábrica de Avatares entrando com as informações básicas e além disso:

1. Possibilitar inserir os vários Avatares disponíveis e Armazenados criando os objetos das classes herança não abstratas
   1. OBS: Futuramente (por outra equipe de projeto) o botão que insere o Avatar abrirá um outro painel para criação do Avatar com detalhes.
2. Completar os detalhes da classe de FabricaAvatar, se necessário
3. Completar o código da classe FabricaAvatarArquivo para escrever e ler todos os dados dos Avatares em um arquivo.
4. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas e vá chamando todos os métodos de escrita e leitura dos outros objetos que compõe o livro (Veja nota geral a parte)

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

public abstract class Avatar **e todas as suas heranças**

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public abstract class FabricaAvatar {

public class FabricaAvatarArquivo extends FabricaAvatar{

### **MODULO ELROND** - Construção de Avatares Masculinos

Este módulo irá somente criar “Modelos” de avatares que serão futuramente configurados por jogadores. Você deve colocar atributos disponíveis a serem preenchidos depois pelo jogador, como por exemplo, “Estória de vida”, Esposa, filhos.... etc... e outras coisas que você achar que o jogador irá preencher especificamente para o seu personagem. Se limite a coisas que façam sentido a categoria de avatares masculinos.

1. Criar uma interface visual para construção de avatares masculinos, entrando com as informações básicas dos avatares assim como as informações extras referentes a Avatares do tipo Masculino
2. Criar novas classes de avatares masculinos, por herança e que já tenham alguns dos parâmetros preenchidos no construtor, por exemplo, um Elfo poderia já ter uma quantidade de pontos de Força e Magia pré-estabelecidos.
3. Novos atributos específicos que pertençam a todos os objetos dessa categoria podem e devem ser acrescentados na classe abstrata
4. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas.
5. Ao terminar a criação, o Avatar deverá ser inserido na “Fabrica de Avatares” na seção de avatares “Disponíveis” para os jogadores e chamar os métodos de “Salvar” para guardar os avatares. Assuma a Fábrica de Avatares já feita por outro grupo.

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

public abstract class Avatar {

public abstract class FabricaAvatar {

public class FabricaAvatarArquivo extends FabricaAvatar{

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public abstract class AvatarMasculino extends Avatar {

### **MODULO GALADRIEL** - Construção de Avatares Femininos

Este módulo irá somente criar “Modelos” de avatares que serão futuramente configurados por jogadores. Você deve colocar atributos disponíveis a serem preenchidos depois pelo jogador, como por exemplo, “Estória de vida”, filhos.... etc... e outras coisas que vc achar que o jogador irá preencher especificamente para o seu personagem. Se limite a coisas que façam sentido a categoria de avatares femininos.

1. Criar uma interface visual para construção destes avatares, entrando com as informações básicas dos avatares assim como as informações extras referentes a Avatares deste tipo
2. Criar novas classes de avatares deste tipo, por herança e que já tenham alguns dos parâmetros preenchidos no construtor, por exemplo, um Elfo poderia já ter uma quantidade de pontos de Força e Magia pré-estabelecidos.
3. Novos atributos específicos que pertençam a todos os objetos dessa categoria podem e devem ser acrescentados na classe abstrata
4. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas.
5. Ao terminar a criação, o Avatar deverá ser inserido na “Fabrica de Avatares” na seção de avatares “Disponíveis” para os jogadores e chamar os métodos de “Salvar” para guardar os avatares. Assuma a Fábrica de Avatares já feita por outro grupo.

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

public abstract class Avatar {

public abstract class FabricaAvatar {

public class FabricaAvatarArquivo extends FabricaAvatar{

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public abstract class AvatarFeminino extends Avatar {

### **MODULO RADAGAST** - Construção de Avatares Neutros

Este módulo irá somente criar “Modelos” de avatares que serão futuramente configurados por jogadores. Você deve colocar atributos disponíveis a serem preenchidos depois pelo jogador, como por exemplo, “Estória de vida”, filhos.... etc... e outras coisas que vc achar que o jogador irá preencher especificamente para o seu personagem. Se limite a coisas que façam sentido a categoria de avatares “Neutros”

1. Criar uma interface visual para construção destes avatares, entrando com as informações básicas dos avatares assim como as informações extras referentes a Avatares deste tipo
2. Criar novas classes de avatares deste tipo, por herança e que já tenham alguns dos parâmetros preenchidos no construtor, por exemplo, um Elfo poderia já ter uma quantidade de pontos de Força e Magia pré-estabelecidos.
3. Novos atributos específicos que pertençam a todos os objetos dessa categoria podem e devem ser acrescentados na classe abstrata
4. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas.
5. Ao terminar a criação, o Avatar deverá ser inserido na “Fabrica de Avatares” na seção de avatares “Disponíveis” para os jogadores e chamar os métodos de “Salvar” para guardar os avatares. Assuma a Fábrica de Avatares já feita por outro grupo.

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

public abstract class Avatar {

public abstract class FabricaAvatar {

public class FabricaAvatarArquivo extends FabricaAvatar{

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public abstract class AvatarNeutro extends Avatar {

### **MODULO BALROG** - Construção de Avatares Criaturas (Monstros)

Este módulo irá somente criar “Modelos” de avatares que serão futuramente escolhidos pelos construtores dos jogos para serem personagens usados na estória. Você deve colocar atributos disponíveis a serem preenchidos depois pelos construtores, como por exemplo, de onde ele veio, o que ele come, o que o irrita, porque ele ataca ou do que tem medo e outras coisas que vc achar que será preenchido especificamente para um certo monstro daquela categoria. Se limite a coisas que façam sentido a categoria.

1. Criar uma interface visual para construção destes avatares, entrando com as informações básicas dos avatares assim como as informações extras referentes a Avatares deste tipo
2. Criar novas classes de avatares deste tipo, por herança e que já tenham alguns dos parâmetros preenchidos no construtor, por exemplo, um Orco poderia já ter uma quantidade de pontos de Força e Defesa pré-estabelecidos.
3. Novos atributos específicos que pertençam a todos os objetos dessa categoria podem e devem ser acrescentados na classe abstrata
4. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas.
5. Ao terminar a criação, o Avatar deverá ser inserido na “Fabrica de Avatares” na seção de avatares “Disponíveis” para os criadores de jogos e chamar os métodos de “Salvar” para guardar os avatares. Assuma a Fábrica de Avatares já feita por outro grupo.

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

public abstract class Avatar {

public abstract class FabricaAvatar {

public class FabricaAvatarArquivo extends FabricaAvatar{

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public abstract class AvatarMonstro extends Avatar {

## Animais de Estimação --

### **MODULO SHADOWFAX** – Criação de Animais de Estimação

Este módulo irá somente criar “Modelos” de avatares que serão futuramente escolhidos pelos construtores dos jogos para serem usados pelos personagens na estória. Você deve colocar atributos disponíveis a serem preenchidos depois pelos construtores, como por exemplo, de onde ele veio, o que ele come, o que o irrita, porque ele ataca ou do que tem medo e outras coisas que vc achar que será preenchido especificamente para um certo monstro daquela categoria. Se limite a coisas que façam sentido a categoria.

1. Criar uma interface visual para construção destes avatares, entrando com as informações básicas dos avatares assim como as informações extras referentes a Avatares deste tipo
2. Criar novas classes de avatares deste tipo, por herança e que já tenham alguns dos parâmetros preenchidos no construtor, por exemplo, um Águia, Lobo, etc.
3. Novos atributos específicos que pertençam a todos os objetos dessa categoria podem e devem ser acrescentados na classe abstrata
4. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas.
5. Ao terminar a criação, o Avatar deverá ser inserido na “Fabrica de Avatares” na seção de avatares “Disponíveis” para os criadores de jogos e chamar os métodos de “Salvar” para guardar os avatares. Assuma a Fábrica de Avatares já feita por outro grupo.

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

public abstract class Avatar {

public abstract class FabricaAvatar {

public class FabricaAvatarArquivo extends FabricaAvatar{

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public abstract class AvatarPet extends Avatar {

### **MODULO PETSHADOW** – Cuidar de Animal de Estimação

Este módulo tem por finalidade “tratar” de um Avatar de animal de estimação diretamente, alimentando, interagindo com ele. A classe “PetCare” será responsável por abrigar e receber os comandos que vão interagir com o tipo de Pet que estiver nela.

Criar uma interface visual para interação com o PET agindo com ele e vendo suas reações e estado atual. Acrescente:

1. Novos atributos específicos que pertençam a todos os objetos dessa categoria podem e devem ser acrescentados na classe abstrata
2. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas.
3. Ao terminar a criação, o Avatar deverá ser inserido na “Fabrica de Avatares” na seção de avatares “Disponíveis” para os criadores de jogos e chamar os métodos de “Salvar” para guardar os avatares. Assuma a Fábrica de Avatares já feita por outro grupo.

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

public abstract class Avatar {

public abstract class FabricaAvatar {

public class FabricaAvatarArquivo extends FabricaAvatar{

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public abstract class AvatarPet extends Avatar {

public class PetCare {

## Modulo Itens de Jogo --

### **MODULO GANDWISE**

#### Gerencia e Armazenamento dos Itens

Este módulo irá criar um “Gerente Central” que irá armazenar todos os Itens que serão futuramente escolhidos pelos construtores dos Itens.

Criar uma interface visual para construção da Fábrica de Itens entrando com as informações básicas e além disso:

1. Possibilitar inserir os vários Itens disponíveis criando os objetos das classes herança não abstratas
   1. OBS: Futuramente (por outra equipe de projeto) o botão que insere o Item abrirá um outro painel para criação do Avatar com detalhes.
2. Completar os detalhes da classe de FabricaItens, se necessário
3. Completar o código da classe FabricaItensArquivo para escrever e ler todos os dados dos Itens em um arquivo.
4. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas e vá chamando todos os métodos de escrita e leitura dos outros objetos que compõe o livro (Veja nota geral a parte)

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

public abstract class ItemJogo **e todas as suas heranças**

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public abstract class FabricaItens {

public class FabricaItensArquivo extends FabricaItens{

#### Sacola de Itens

Este módulo irá criar coleções de itens que serão usados pelos jogadores nas missões dos jogos.

Criar uma interface visual para construção da Sacola entrando com as informações básicas e além disso:

1. Possibilitar inserir os vários tipos de itens criando objetos das classes herança de ItemJogo.
   1. OBS: Futuramente (por outra equipe de projeto) o botão que insere um tipo de item, abrirá um outro painel para criação com detalhes.
2. Completar os detalhes da classe de SacolaItens e SacolaItensArquivo
3. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas e vá chamando todos os métodos de escrita e leitura dos objetos do mapa (Veja nota geral a parte)

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

public abstract class ItemJogo {

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public abstract class SacolaItens {

public class SacolaItensArquivo extends SacolaItens{

### **MODULO LEGOLAS** – Construção de Itens de Armas

Este módulo irá somente criar “Modelos” de itens que serão futuramente escolhidos por jogadores.

Criar uma interface visual para construção das armas, entrando com as informações básicas dos itens de jogo assim como as informações extras referentes a “Armas” e eventualmente ao tipo de classe de arma especifica, por exemplo, “Espada” teria um atributo Comprimento.

1. Criar novas classes de **armas** por herança e que já tenham alguns dos parâmetros preenchidos no construtor, por exemplo, uma espada já pode ter o número de pontos de dano.
2. Novos atributos específicos que pertençam a todos os objetos dessa categoria podem e devem ser acrescentados na classe abstrata
3. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas.
4. Ao terminar a criação, o item deverá ser inserido na “Fabrica” na seção de “Disponíveis” para os jogadores e chamar os métodos de “Salvar” para salvar os itens. Assuma a fábrica já feita por outro grupo.

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

abstract class ItemJogo

abstract class FabricaItens {

public class FabricaItensArquivo extends FabricaItens{

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public abstract class Arma extends ItemJogo{

### **MODULO TURIN**– Construção de itens de Valores

Este módulo irá somente criar “Modelos” de itens que serão futuramente escolhidos por jogadores.

Criar uma interface visual para construção dos tipos de valores, entrando com as informações básicas dos itens de jogo assim como as informações extras referentes a “Valores” e eventualmente ao tipo de classe de arma especifica, por exemplo, “Diamantes” teria um atributo Peso.

1. Criar novas classes de **Valores** por herança e que já tenham alguns dos parâmetros preenchidos no construtor, se necessário
2. Novos atributos específicos que pertençam a todos os objetos dessa categoria podem e devem ser acrescentados na classe abstrata
3. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas.
4. Ao terminar a criação, o item deverá ser inserido na “Fabrica” na seção de “Disponíveis” para os jogadores e chamar os métodos de “Salvar” para salvar os itens. Assuma a fábrica já feita por outro grupo.

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

abstract class ItemJogo

abstract class FabricaItens {

public class FabricaItensArquivo extends FabricaItens{

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public abstract class Valores extends ItemJogo{

### **MODULO DWARF** – Construção de itens de Ferramentas

Este módulo irá somente criar “Modelos” de itens que serão futuramente escolhidos por jogadores.

Criar uma interface visual para construção das ferramentas, entrando com as informações básicas dos itens de jogo assim como as informações extras referentes a “Ferramentas” e eventualmente ao tipo de classe de ferramenta especifica, por exemplo, “Corda” teria um atributo Comprimento.

1. Criar novas classes de **Ferramenta** por herança e que já tenham alguns dos parâmetros preenchidos no construtor, por exemplo, uma espada já pode ter o número de pontos de dano.
2. Novos atributos específicos que pertençam a todos os objetos dessa categoria podem e devem ser acrescentados na classe abstrata
3. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas.
4. Ao terminar a criação, o item deverá ser inserido na “Fabrica” na seção de “Disponíveis” para os jogadores e chamar os métodos de “Salvar” para salvar os itens. Assuma a fábrica já feita por outro grupo.

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

abstract class ItemJogo

abstract class FabricaItens {

public class FabricaItensArquivo extends FabricaItens{

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public abstract class Ferramenta extends ItemJogo{

### **MODULO GANDALF** – Construção de itens Mágicos

Este módulo irá somente criar “Modelos” de itens que serão futuramente escolhidos por jogadores.

Criar uma interface visual para construção dos Itens Mágicos, entrando com as informações básicas dos itens de jogo assim como as informações extras referentes a “Itens Mágicos” e eventualmente ao tipo de classe especifica, por exemplo, “Tocha” teria um atributo distancia de iluminação.

1. Criar novas classes de **ItemMagico** por herança e que já tenham alguns dos parâmetros preenchidos no construtor, por exemplo, uma varinha já pode ter o número de pontos de dano.
2. Novos atributos específicos que pertençam a todos os objetos dessa categoria podem e devem ser acrescentados na classe abstrata
3. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas.
4. Ao terminar a criação, o item deverá ser inserido na “Fabrica” na seção de “Disponíveis” para os jogadores e chamar os métodos de “Salvar” para salvar os itens. Assuma a fábrica já feita por outro grupo.

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

abstract class ItemJogo

abstract class FabricaItens {

public class FabricaItensArquivo extends FabricaItens{

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public abstract class ItemMagico extends ItemJogo{

## Localização e Mapas --

### **MODULO TERRAMEDIA**

#### Construção de Mapas

Este módulo irá criar os mapas que serão futuramente escolhidos pelos construtores das missões dos jogos.

Criar uma interface visual para construção do Mapa entrando com as informações básicas do Mapa e além disso:

1. Possibilitar inserir os vários tipos de locais no mapa criando objetos das classes de LocalMapa da lista abaixo.
   1. OBS: Futuramente (por outra equipe de projeto) o botão que insere um tipo de local, abrirá um outro painel para criação do local com detalhes.
2. Criar e Inserir objetos trajetos – Completar essa classe, se necessário e prover a criação de trajetos.
3. Completar os detalhes da classe de MapaJogo
4. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas e vá chamando todos os métodos de escrita e leitura dos objetos do mapa (Veja nota geral a parte)

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

public abstract class LocalJogo {

public abstract class LocalMapa extends LocalJogo{

Criar opções no mapa para inserção de todos os tipos de locais abaixo

* public class CavernaMapa extends LocalMapa{
* public class CidadeMapa extends LocalMapa{
* public class LagoMapa extends LocalMapa{
* public class RioMapa extends LocalMapa{
* public class TavernaMapa extends LocalMapa{

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public class MapaJogo {

public class Trajeto {

#### Construção de Locais de Mapas – CidadeMapa

Este módulo irá somente criar os locais de mapa que serão futuramente escolhidos pelos construtores de mapas.

Criar uma interface visual para construção do local entrando com as informações básicas do local assim como as informações extras referentes a este tipo de local e eventualmente ao tipo de classe de um local especifico como “Taverna do Bilbo”, por exemplo, teria um atributo nome do dono.

1. Completar os detalhes das classes de local
2. Novos atributos específicos que pertençam a todos os objetos dessa categoria podem e devem ser acrescentados na classe abstrata **LocalMapa**
3. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas.

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

public abstract class LocalJogo {

public abstract class LocalMapa extends LocalJogo{

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public class CidadeMapa extends LocalMapa{

#### Construção de Locais de Mapas –CavernaMapa

Este módulo irá somente criar os locais de mapa que serão futuramente escolhidos pelos construtores de mapas.

Criar uma interface visual para construção do local entrando com as informações básicas do local assim como as informações extras referentes a este tipo de local e eventualmente ao tipo de classe de um local especifico como “Caverna de Smaug”, por exemplo, teria um atributo nome do monstro.

1. Completar os detalhes das classes de local
2. Novos atributos específicos que pertençam a todos os objetos dessa categoria podem e devem ser acrescentados na classe abstrata **LocalMapa**
3. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas.

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

public abstract class LocalJogo {

public abstract class LocalMapa extends LocalJogo{

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public class CavernaMapa extends LocalMapa{

#### Construção de Locais de Mapas – LagoMapa

Este módulo irá somente criar os locais de mapa que serão futuramente escolhidos pelos construtores de mapas.

Criar uma interface visual para construção do local entrando com as informações básicas do local assim como as informações extras referentes a este tipo de local e eventualmente ao tipo de classe de um local especifico.

1. Completar os detalhes das classes de local
2. Novos atributos específicos que pertençam a todos os objetos dessa categoria podem e devem ser acrescentados na classe abstrata **LocalMapa**
3. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas.

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

public abstract class LocalJogo {

public abstract class LocalMapa extends LocalJogo{

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public class LagoMapa extends LocalMapa{

#### Construção de Locais de Mapas – RioMapa

Este módulo irá somente criar os locais de mapa que serão futuramente escolhidos pelos construtores de mapas.

Criar uma interface visual para construção do local entrando com as informações básicas do local assim como as informações extras referentes a este tipo de local e eventualmente ao tipo de classe de um local especifico como “Rio de Rivendel”, por exemplo, teria um atributo povo que vive lá.

1. Completar os detalhes das classes de local
2. Novos atributos específicos que pertençam a todos os objetos dessa categoria podem e devem ser acrescentados na classe abstrata **LocalMapa**
3. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas.

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

public abstract class LocalJogo {

public abstract class LocalMapa extends LocalJogo{

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public class RioMapa extends LocalMapa{

#### Construção de Locais de Mapas – TavernaMapa

Este módulo irá somente criar os locais de mapa que serão futuramente escolhidos pelos construtores de mapas.

Criar uma interface visual para construção do local entrando com as informações básicas do local assim como as informações extras referentes a este tipo de local e eventualmente ao tipo de classe de um local especifico como “Taverna do Bilbo”, por exemplo, teria um atributo nome do dono.

1. Completar os detalhes das classes de local
2. Novos atributos específicos que pertençam a todos os objetos dessa categoria podem e devem ser acrescentados na classe abstrata **LocalMapa**
3. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas.

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

public abstract class LocalJogo {

public abstract class LocalMapa extends LocalJogo{

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public class TavernaMapa extends LocalMapa{

### **MODULO CELEBORN** – Construção de Livros de Mapas

Este módulo irá criar um “Gerente Central” que irá armazenar todos os mapas que serão futuramente escolhidos pelos construtores das missões dos jogos.

Criar uma interface visual para construção do Livro de Mapas Mapa entrando com as informações básicas do Livro de Mapa e além disso:

1. Possibilitar inserir os vários mapas criando objetos das classes de MapaJogo
   1. OBS: Futuramente (por outra equipe de projeto) o botão que insere o Mapa abrirá um outro painel para criação do Mapa com detalhes.
2. Completar os detalhes da classe de LivroMapa, se necessário
3. Completar o código da classe LivroMapasArquivo para escrever e ler todos os dados dos Mapas em um arquivo.
4. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas e vá chamando todos os métodos de escrita e leitura dos outros objetos que compõe o livro (Veja nota geral a parte)

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

public class MapaJogo {

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public abstract class LivroMapas {

public class LivroMapasArquivo extends LivroMapas{

### **MODULO BEORN** – Localização de Avatares

Este módulo irá somente criar os locais aonde os Avatares estarão localizados.

Criar uma interface visual para construção do local entrando com as informações básicas do local

1. Completar os detalhes da classe, se necessário
2. Crie um método que receba outro LocalAvatar como parâmetro e retorne a distância entre estes avatares
3. Obter e inserir o LocalAvatar nos em objetos de herança da classe Avatar
4. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas.

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

public abstract class LocalJogo {

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public abstract class Avatar {

public class LocalAvatar extends LocalJogo{

## Modulo Estórias -- **MODULO Bilbo**

### Construção de Estórias

Irá gerenciar a construção das estórias a partir das frases que estariam sendo construídas no outro módulo. Para este módulo, pode se considerar objetos criados apenas para teste.

Deverá criar uma interface com o usuário para ler todas as frases da estória e escrevendo elas na interface assim como em um arquivo. A interface deverá possibilitar “Regerar” a estória aleatoriamente através de novos métodos na classe “FraseEstoria” que sorteie novas palavras para a frase em questão....

1. Completar os detalhes das classes modificáveis abaixo.
2. Carregar o objeto estória
3. Construir o método “retornarFraseAleatória” lendo os identificadores e substituindo pelas frases dos objetos frase de acordo.
4. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas.
5. Criar os métodos para salvar a estória em um formato de leitura assim como na tela

**Classes utilizadas:**

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public class EstoriaJogo {

public class FrasesEstoria {

public class PalavrasFrase {

### Construção de Frases de Estórias

Irá criar frases “Modificáveis” que serão utilizadas para criar estórias dinamicamente. Cada objeto “FraseEstoria” terá uma String aonde um identificador da forma: “#ZOIDP\_01” , “#ZOIDP\_02”, “#ZOIDP\_03”, .... irá indicar uma possível lista de palavras que estará num objeto “PalavrasFrase” que estará associado a esse identificador. Por exemplo, se interface cria uma frase assim:

A ***#ZOIDP\_01*** que estava no ***#ZOIDP\_02 #ZOIDP\_03*** desde #***ZOIDP\_04***

E ela poderia gerar as múltiplas frases assim, aleatoriamente:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A | #ZOIDP\_01 | que estava no | #ZOIDP\_02 | #ZOIDP\_03 | desde | #ZOIDP\_04 |
|  | pedra |  | chão | Fugindo |  | ontem |
|  | vassoura |  | Reino | Bebendo |  | ano passado |
|  | vaca |  | castelo | escondida |  | a morte |
|  | arvore |  | pescoço | Brilhando |  | a chuva |

Criar uma interface visual para construção das frases, aonde o usuário vai digitar os identificadores e o programa deve ler e procurar os identificadores na frase e adicionar objetos “PalavraFrase” para cada identificador. Os objetos “PalavrasFrase” podem ser criados diretamente pois haverá outra equipe para construir esses objetos. Um botão futuramente abrirá um segundo painel para criação de PalavrasFrase. Este módulo deverá:

1. Completar os detalhes das classes modificáveis abaixo.
2. Inserir as Frases prontas no objeto “Estória” e mandar salvá-lo
3. Carregar o objeto estória antes de acrescentar mais frases.
4. Construir o método “retornarFrase” lendo os identificadores e substituindo pelas frases dos objetos frase de acordo.
5. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas.

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

public class EstoriaJogo {

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public class FrasesEstoria {

public class PalavrasFrase {

### Construção de Palavras de Frases

Parte do que irá criar frases “Modificáveis” que serão utilizadas para criar estórias dinamicamente. Cada objeto “FraseEstoria” terá uma String aonde um identificador da forma: “#ZOIDP\_01”, “#ZOIDP\_02”, “#ZOIDP\_03”, .... irá indicar uma possível lista de palavras que estará num objeto “PalavrasFrase” que estará associado a esse identificador. Por exemplo, se a interface cria uma frase assim:

A ***#ZOIDP\_01*** que estava no ***#ZOIDP\_02 #ZOIDP\_03*** desde #***ZOIDP\_04***

E ela poderia gerar as múltiplas frases assim, aleatoriamente:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A | #ZOIDP\_01 | que estava no | #ZOIDP\_02 | #ZOIDP\_03 | desde | #ZOIDP\_04 |
|  | pedra |  | chão | Fugindo |  | ontem |
|  | vassoura |  | Reino | Bebendo |  | ano passado |
|  | vaca |  | castelo | escondida |  | a morte |
|  | arvore |  | pescoço | Brilhando |  | a chuva |

Este modulo deverá ser responsável pela criação das colunas de possíveis palavras para os identificadores (como no modelo acima)

Criar uma interface visual para construção das frases, aonde o usuário vai digitar os identificadores e depois a lista de palavras alternativas criando o objeto PalavraFrase que deve ser inserido em um objeto frase.

Deverá:

1. Completar os detalhes das classes modificáveis abaixo.
2. Inserir as Palavras no objeto “FraseEstoria” e mandar salvá-lo
3. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas.

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

public class EstoriaJogo {

public class FrasesEstoria {

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public class PalavrasFrase {

### Algoritmos Auxiliares de palavras

Crie um sistema que primeiro leia uma lista de palavras no formato “#ZOIDP\_01”, “#ZOIDP\_02”, “#ZOIDP\_03”, e armazene em um HashMap. A seguir, abra um arquivo e leia todas as linhas, uma a uma e calcule a quantidade de cada um dos identificadores inseridos no HashMap.

Consulte Módulos Anteriores para mais informações

### Construção de Palavras de Frases

Parte do que irá criar frases “Modificáveis” que serão utilizadas para criar estórias dinamicamente. Cada objeto “FraseEstoria” terá uma String aonde um identificador da forma: “#ZOIDP\_01”, “#ZOIDP\_02”, “#ZOIDP\_03”, .... irá indicar uma possível lista de palavras que estará num objeto “PalavrasFrase” que estará associado a esse identificador. Por exemplo, devemos ler do teclado uma frase assim:

A ***#ZOIDP\_01*** que estava no ***#ZOIDP\_02 #ZOIDP\_03*** desde #***ZOIDP\_04***

E ela poderia gerar as múltiplas frases assim, aleatoriamente:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A | #ZOIDP\_01 | que estava no | #ZOIDP\_02 | #ZOIDP\_03 | desde | #ZOIDP\_04 |
|  | pedra |  | chão | Fugindo |  | ontem |
|  | vassoura |  | Reino | Bebendo |  | ano passado |
|  | vaca |  | castelo | Escondida |  | a morte |
|  | arvore |  | pescoço | Brilhando |  | a chuva |

A lista de palavras, para cada identificador pode ser criada diretamente no programa principal.

Criar entrada de dados aonde o usuário vai digitar a frase criando o objeto FraseEstoria onde deve ser inserido em um objeto PalavraFrase criado no programa.

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

public class FrasesEstoria {

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public class PalavrasFrase {

### Construção e gerencia de Livros de Estórias

Este módulo irá criar um “Gerente Central” que irá armazenar todas as Estórias que serão futuramente escolhidos pelos construtores dos jogos

Criar uma interface visual para construção do Livro de Estórias entrando com as informações básicas e além disso:

1. Possibilitar inserir os várias Estórias criando objetos das classes de EstóriaJogo
   1. OBS: Futuramente (por outra equipe de projeto) o botão que insere abrirá um outro painel para criação com detalhes.
2. Completar os detalhes da classe de LivroEstoria, se necessário
3. Completar o código da classe LivroEstoriaArquivo para escrever e ler todos os dados em um arquivo.
4. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas e vá chamando todos os métodos de escrita e leitura dos outros objetos que compõe o livro (Veja nota geral a parte)

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

public class EstoriaJogo {

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public abstract class LivroEstorias {

public class LivroEstoriasArquivo extends LivroEstorias{

## **MODULO EOMER** - Módulo Missões

### Construção de Missões e Armazenamento

Este módulo tem por finalidade construir uma missão para posterior armazenamento no livro de missões a serem usados pelos jogadores.

1. Criar uma interface visual para a criação da missão e preencher as informações básicas dos atributos.
2. Completar os detalhes das classes modificáveis se necessário.
3. Criar objetos que compõe a classe missão diretamente. Futuramente eles serão criados independentemente por outra equipe:
4. EstoriaJogo: Este objeto será obtido de um objeto “LivroEstorias”
5. MapaJogo: Este objeto será obtido de um objeto LivroMapas
6. SacolaItems: Criado a partir de uma interface a parte. Pode ser criado como um objeto de teste diretamente aqui.
7. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas.

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

public abstract class LivroMissoes {

public class EstoriaJogo {

public class MapaJogo {

public abstract class SacolaItens {

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public abstract class Missao {

public class MissaoArquivo extends Missao{

### Construção e Armazenamento do Livro de Missões

Este módulo irá criar um “Gerente Central” que irá armazenar todas as Missões que serão futuramente escolhidos pelos construtores dos jogos

Criar uma interface visual para construção do Livro de Missões entrando com as informações básicas e além disso:

1. Possibilitar inserir os várias Missões criando objetos das classes de MissãoArquivo
   1. OBS: Futuramente (por outra equipe de projeto) o botão que insere abrirá um outro painel para criação com detalhes.
2. Completar os detalhes da classe de LivroMissoes, se necessário
3. Completar o código da classe LivroMissoesArquivo para escrever e ler todos os dados em um arquivo.
4. Preencha os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas e vá chamando todos os métodos de escrita e leitura dos outros objetos que compõe o livro (Veja nota geral a parte)

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

public abstract class Missao {

public class MissaoArquivo extends Missao{

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public abstract class LivroMissoes {

public class LivroMissoesArquivo extends LivroMissoes {

## Módulo Encontros e Combates

### **MODULO BOROMIR** – Construção de Conversa

Este módulo deverá criar uma interface aonde dois jogadores ou Avatares são selecionados e um objeto “Conversa” deve ser criado passando a referência aos dois jogadores. Este objeto deve ser inserido também na lista de encontros de cada jogador assim como no objeto “Partida”:

1. Defina uma estratégia para resultado da conversa entre os dois avatares baseado em suas caraterísticas. No final, o resultado deve apresentar um ganhador com indicações se:
   1. Um Avatar convenceu o outro de algo
   2. Um Avatar enganou o outro
   3. Um Avatar conseguiu extrair as informações do outro.
2. Criar novos atributos específicos que pertençam a todos os objetos dessa categoria se necessário.
3. Crie os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas.

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

public abstract class Avatar {

public abstract class FabricaAvatar {

public abstract class Encontro {

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public class Conversa extends Encontro{

### **MODULO FARAMIR** – Construção de Combates

Este módulo deverá criar uma interface aonde dois jogadores ou Avatares são selecionados e um objeto “Combate” deve ser criado passando a referência aos dois jogadores. Este objeto deve ser inserido também na lista de encontros de cada jogador assim como no objeto “Partida”:

1. Defina uma estratégia para resultado do combate entre os dois avatares baseado em suas caraterísticas. No final, o resultado deve apresentar um ganhador e os respectivos pontos de força, defesa e vida serem atualizados em cada jogador de acordo.
2. Criar novos atributos específicos que pertençam a todos os objetos dessa categoria se necessário.
3. Crie os métodos de escrita e leitura dos dados em arquivo, ou banco de dados criando os métodos apropriados, porem sendo arquivo, assumindo que todas as informações serão escritas em uma só linha separada por vírgulas.

**Classes utilizadas:**

NÃO MODIFICÁVEIS:

public abstract class Avatar {

public abstract class FabricaAvatar {

public abstract class Encontro {

MODIFICAÇÕES ADITIVAS (Somente adicionar atributos, métodos ou código):

public class Combate extends Encontro{

# NOTAS e GUIAS GERAIS

1. Todas as equipes podem criar suas próprias versões do inicializador para teste (GerenteZoidV1, GerenteZoidV2, etc..) e modificar a classe Zoid em seus próprios ambientes.
2. Se as classes forem abstratas então devem criadas heranças dessas classes para que se possam criar objetos delas, por exemplo, “Arma” ... Devem ser criados, objetos “Espada”, “Arco”, etc.
3. Nas classes abstratas “Pai” poderão ser acrescentados atributos que sejam comuns a todas as heranças daquela classe. Por exemplo, você pode acrescentar “tamanhoBarba” a classe abstrata AvatarMasculino.
4. NUNCA remover atributos ou alterar métodos das classes fornecidas
5. Salvar e Ler todas as informações num arquivo
6. Acrescentar atributo e construtor para nome do arquivo se necessário
7. Acrescentar código para escrever todo o objeto com seus objetos internos dentro de um arquivo se necessário. Se o método for “principal”, iniciar a criação dos objetos de leitura e escrita FileWriter ou FileReader e depois passar eles como parâmetro para os outros objetos dentro da classe para que se escrevam ou se leiam.
8. Acrescentar código para escrever e ler os dados em FileWriter e FileReader no seguinte formato:
   1. public void salvar***XYZ***(FileWriter fw) {

<Chamar Salvar da classe pai, se houver>

fw.write(“<Nome da Classe>:” + “,”);

fw.write(“<Atributo 1>” + “,”);

fw.write(“<Atributo 2>” + “,”);

.....

fw.write(“<Atributo n>” + “\r\n”);

<Chamar o salvar para os outros objetos contidos na classe>

}

* 1. public void carregar***XYZ***(BufferedReader br) {

<Chamar leitura da classe pai, se houver>

<Ler cada atributo separado por virgula lembrando que o primeiro é sempre o nome da classe sendo escrita>

<Preencher os atributos da classe com os dados lindos>

<Chamar o leitura para os outros objetos contidos na classe