**Requisitos de implementação para C++**

**A explicação do que cada classe faz está no documento “Explicação das classes” nesta mesma pasta.**

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

Hierarquia Item:

Item(abstract) >> Kuraudo\_Mirrors(Concreta)

Item (abstract) >>Equipamento (abstract) >> Arma (abstract) >> Espada (Concreta)

Item (abstract) >>Equipamento (abstract) >> Armadura (abstract) >> Roupas\_Leves (Concreta)

Item (abstract) >>Equipamento (abstract) >> Acessorio (abstract) >> Anel\_De\_Forca (Concreta)





















Hierarquia Heroi:

Heroi (abstract) >> Mago >> (concreta) >> Guerreiro (concreta) >> Ladino(concreta)









Hierarquia Monstro:

Monstro (abstract) >> Camtasia (concreta) >> Kuraudo(concreta)







Hierarquia Cidade:

Cidade (abstract) >> Sepolopolis (concreta)





Hierarquia Dungeon:

Dungeons (Abstract) >> Caverna\_Monstro(concreta)





Hierarquia Grupo:

Grupo\_Geral (abstract) >> Grupo(concreta)





Hierarquia Magia:

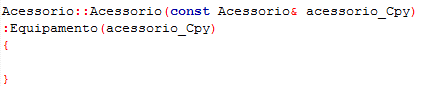
Magia (abstract) >> Trovao( concreta)





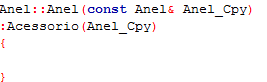
1. Em todas as classes: **construtor de cópia**, **operatores<< e +=,** e **construtor default**. Fazer o máximo de reaproveitamento de código usando **static\_cast**

Classe: Acessorio





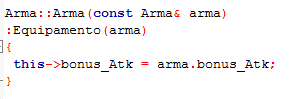
Classe: Anel\_De\_Forca





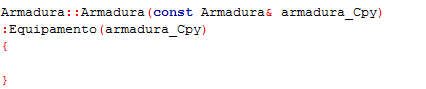


Classe: Arma



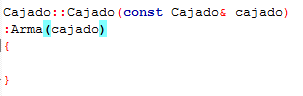


Classe: Armadura



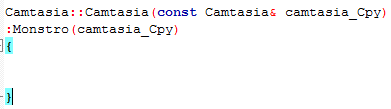


Classe: Cajado



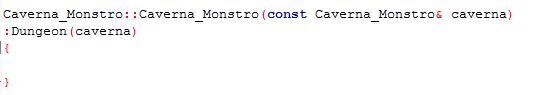


Classe: Camtasia



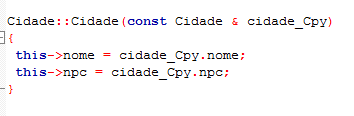


Classe: Caverna\_Monstro



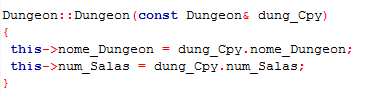


Classe: Cidade



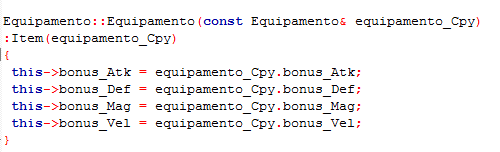


Classe: Dungeon



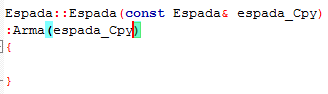


Classe: Equipamento



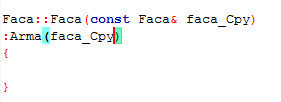


Classe: Espada



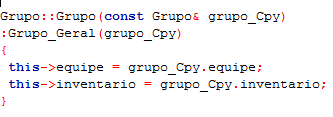


Classe: Faca



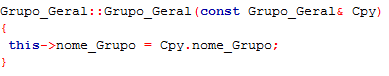


Classe: Grupo



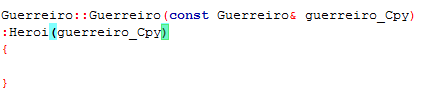


Classe: Grupo\_Geral



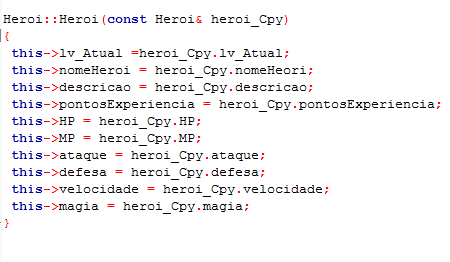


Classe: Guerreiro



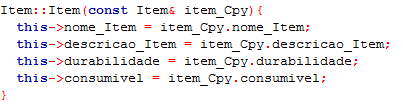


Classe: Heroi



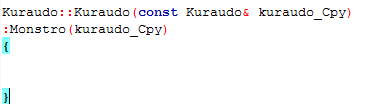


Classe: Item





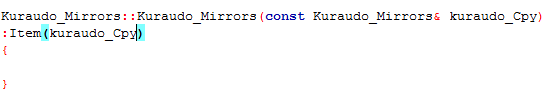
Classe: Kuraudo





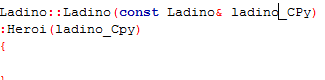


Classe: Kuraudo\_Mirrors



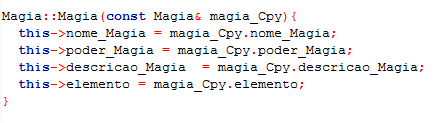


Classe: Ladino



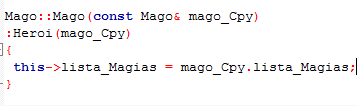


Classe: Magia



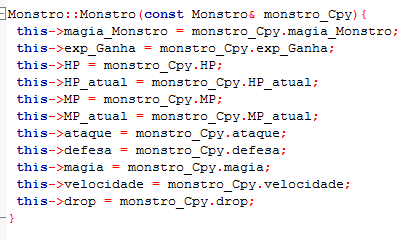


Classe: Mago



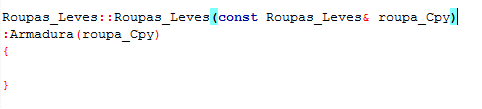


Classe: Monstro



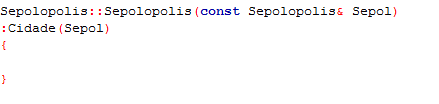


Classe: Roupas\_Leves



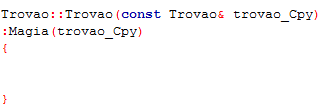


Classe: Sepolopolis





Classe: Trovao





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

Hierarquia Item:

Item(abstract) >> Kuraudo\_Mirrors(Concreta)

Item (abstract) >>Equipamento (abstract) >> Arma (abstract) >> Espada (Concreta)

Item (abstract) >>Equipamento (abstract) >> Armadura (abstract) >> Roupas\_Leves (Concreta)

Item (abstract) >>Equipamento (abstract) >> Acessorio (abstract) >> Anel\_De\_Forca (Concreta)





















Hierarquia Heroi:

Heroi (abstract) >> Mago >> (concreta) >> Guerreiro (concreta) >> Ladino(concreta)









Hierarquia Monstro:

Monstro (abstract) >> Camtasia (concreta) >> Kuraudo(concreta)







Hierarquia Cidade:

Cidade (abstract) >> Sepolopolis (concreta)





Hierarquia Dungeon:

Dungeons (Abstract) >> Caverna\_Monstro(concreta)





Hierarquia Grupo:

Grupo\_Geral (abstract) >> Grupo(concreta)





Hierarquia Magia:





1. Atributos static e const static em todas as hierarquias de classe

Hierarquia: Item





Hierarquia: Monstro



Hierarquia: Heroi



Hierarquia: Cidade



Hierarquia: Dungeon



Hierarquia: Magia



1. Método static em todas as hierarquias de classe

Hierarquia: Cidade



Hierarquia: Dungeon



Hierarquia: Heroi

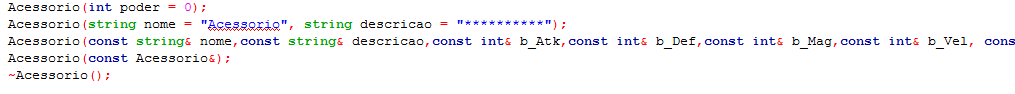


Hierarquia: Item

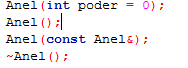


1. Construtores em todas as classes, e três para todas as classes da hierarquia principal. Sempre validar os dados em todas as classes

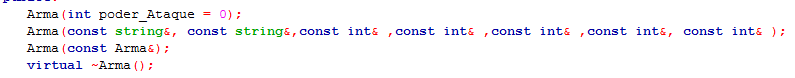
Classe: Acessório



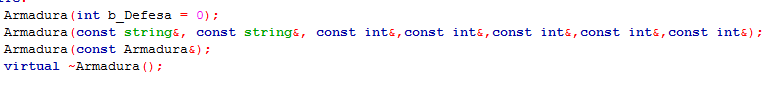
Classe: Anel



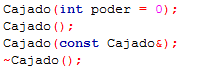
Classe: Arma



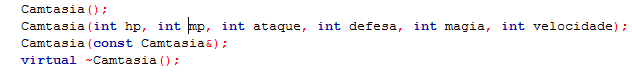
Classe: Armadura



Classe: Cajado



Classe: Camtasia



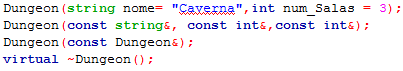
Classe: Caverna\_Monstro



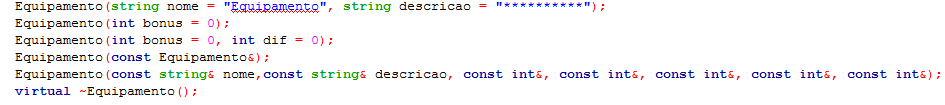
Classe: Cidade



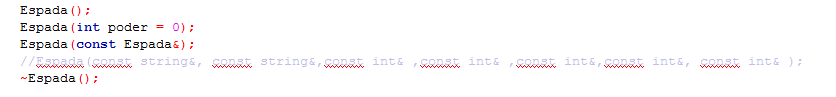
Classe: Dungeon



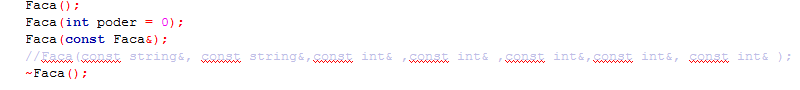
Classe: Equipamento



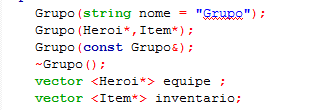
Classe: Espada



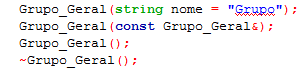
Classe: Faca



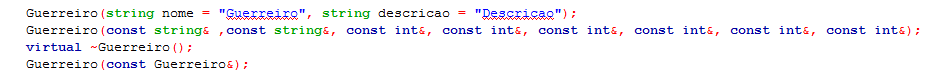
Classe: Grupo



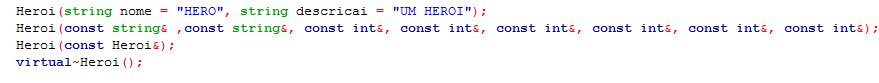
Classe: Grupo\_Geral



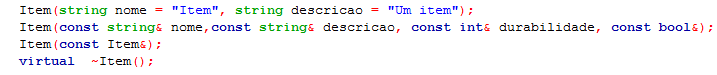
Classe: Guerreiro



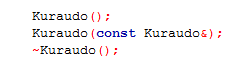
Classe: Heroi



Classe: Item



Classe: Kuraudo



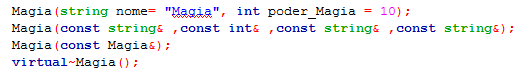
Classe: Kuraudo\_Mirrors



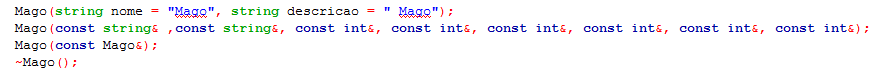
Classe: Ladino



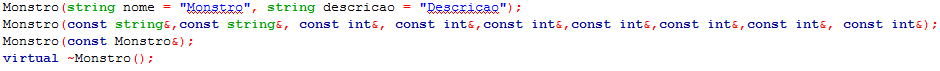
Classe: Magia



Classe: Mago



Classe: Monstro



Classe: Roupas\_Leves



Classe: Sepolópolis



Classe: Trovao



1. Vector em todas em todas as hierarquias de classe

Classe: Mago – Hierarquia Heroi



Classe: Grupo – Hierarquia Grupo



Classe: Equipamento – Hierarquia Item

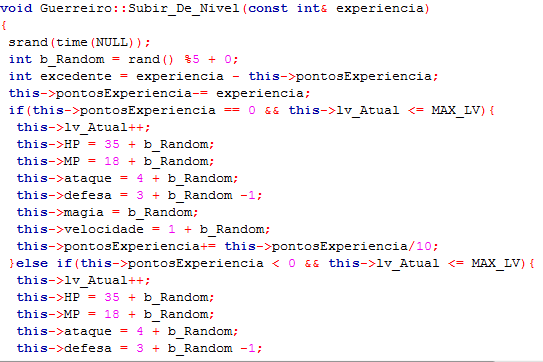


Classe: Cidade– Hierarquia Cidade



1. ENUM na hierarquia principal
2. Usar o **dynamic cast** e **typeid** no main junto com as classes concretas. Para uma da classe concreta identificada, chamar um método dessa classe e fazer uma ação;
3. Usar o rand. 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.http://en.cppreference.com/w/cpp/numeric/random/rand

Classe: Guerreiro.cpp



1. No main o usuário deve fazer entrada via teclado e interagir com a aplicação