**Classe abstrata Item (Item.h e Item.cpp)**

Todo item, tanto equipamento e itens de cura quanto itens chaves, são derivados desta classe abstrata. Todos os itens possuem:

* Uma **string para** **nome**, que será a sua forma de identificação no programa.
* Uma **string para** **descrição**, que seria uma breve explicação sobre o que faz e/ou sua importância para a estória.
* Um **inteiro para a durabilidade**, que seria quantas vezes o item seria utilizado até ser quebrado. Esta variável é mais voltada para os objetos da classe derivada “**Equipamento**” do que para os itens em si
* Uma **variável booleana para verificar se é consumível.** Caso sim, ao ser utilizado, o tem faz seu efeito e saí do inventario da classe “**Grupo**”. Caso não, o item pode ser utilizado diversas vezes (Ou até nem ser utilizado pelo jogador, só servindo como objeto chave para se ativar algum evento).

Este método possui os seguintes construtores:

* **Construtor default**, com valores padrão para o nome e a descrição.
* **Construtor de cópia**, para copiar os valores de outro item. Não é necessário validar os dados.
* **Construtor inicializando e validando os valores** “nome”, “descrição”, “durabilidade” e “consumibilidade”.
* **Destrutor virtual**, para chamar os destrutores das classes derivadas.

Além dos métodos “get()”, “set()” e operadores += e << para as variáveis acima, temos os métodos:

* **virtual void Diminuir\_Durabilidade()**, responsável por diminuir a durabilidade dos itens.
* **virtual void Efeito() = 0**, método virtual puro indicando qual o efeito( Ex: Causar dano, adicionar atributos positivos a um herói, etc). Cada item terá seu próprio efeito, por isso ela é uma função virtual.

**Classe concreta Kuraudo\_Mirrors (Kuraudo\_Mirrors.h e Kuraudo\_Mirrors.cpp).**

**Classe base: Item**

Esta classe não tem muita utilidade ao programa, servindo apenas como um item-chave (que tem a função de ativar algum evento, ou de simbolizar algum grande feito do jogador).

Kuraudo\_Mirrors possui:

* Atributos da classe **Item**( Nome, descrição, etc.).

Este método possui os seguintes construtores. Todos eles chamam o construtor da classe **Item** para realizar o reaproveitamento de código, com exceção do destrutor:

* **Construtor vazio**, com valores pré-determinados.
* **Construtor de cópia**, para construir um objeto usando outro objeto desta mesma classe como referência.
* **Destrutor virtual**, para chamar os destrutores das classes derivadas.

Além dos métodos “get()”, “set()” e operadores += e << para as variáveis acima, temos os métodos:

* **virtual void Efeito() = 0**, método virtual puro indicando qual o efeito( Ex: Causar dano, adicionar atributos positivos a um herói, etc). Cada item terá seu próprio efeito, por isso ela é uma função virtual.
* **void Diminuir\_Durabilidade() :** Diminui a durabilidade do item. Esta função sobrescreve a mesma da classe base.

**Classe abstrata Equipamento (Equipamento.h e Equipamento.cpp). Classe base: Item**

**A Classe Equipamento** é a classe abstrata que corresponde a todos e qualquer equipamento (Arma, Armadura e Acessório). Ela é derivada da classe **Item** e é base para as classes abstratas **“Arma”, “Armadura”e “Acessórios”.**

Equipamento contém:

* Atributos da classe **Item**( Nome, descrição, etc.).
* **Inteiros para causar um bônus nos atributos ataque, defesa magia e velocidade**, que se somará ao valor total dos atributos do herói.

Este método possui os seguintes construtores. Todos eles chamam o construtor da classe **Item** para realizar o reaproveitamento de código, com exceção do destrutor:

* **Construtor default**, com valores padrão para o nome e a descrição. **Construtor de cópia**, para copiar os valores de outro item. Não é necessário validar os dados.
* **Construtor default**, com valores padrão para um único bônus.
* **Construtor default**, com valores padrão para dois bônus.
* **Construtor de cópia**, para copiar os valores de outro Equipamento. Não é necessário validar os dados.
* **Construtor inicializando e validando os valores** “nome”, “descrição”, “durabilidade” e “consumibilidade”, “bonus de ataque”, “bonus de defesa”, “bonus de magia”, “bonus de velocidade”.
* **Destrutor virtual**, para chamar os destrutores das classes derivadas.

Além dos métodos “get()”, “set()” e operadores += e << para as variáveis acima, temos os métodos:

* **virtual void Add\_Bonus(),** que Adiciona um bônus com valor maior ou igual a 0 a um equipamento.
* **virtual void Efeito() = 0**, método virtual puro indicando qual o efeito( Ex: Causar dano, adicionar atributos positivos a um herói, etc). Cada item terá seu próprio efeito, por isso ela é uma função virtual.

**Classe abstrata Arma (Arma.h e Arma.cpp). Classe base: Equipamento**

Com poucas exceções (ex: Pokemon), quase todos os personagens jogáveis de RPG possuem uma arma, diferenciando-se na forma de como esta é trabalhada (se ela é possui atributos fixos, se ela pode ser trocada, se ela pode ser fundida com outra).

Arma contém:

* Atributos da classe **Item** e da classe **Equipamento**.

Este método possui os seguintes construtores. Todos eles chamam o construtor da classe **Equipamento** para realizar o reaproveitamento de código, com exceção do destrutor:

* **Construtor default**, com valores padrão para um único bônus.
* **Construtor de cópia**, para copiar os valores de outra Arma. Não é necessário validar os dados.
* **Construtor inicializando e validando todos os valores necessários.**
* **Destrutor virtual**, para chamar os destrutores das classes derivadas.

Além dos métodos “get()”, “set()” e operadores += e << para as variáveis acima, temos os métodos:

* **virtual void Add\_Bonus(),** que Adiciona um bônus com valor maior ou igual a 0 a um equipamento.
* **virtual void Efeito() = 0**, método virtual puro indicando qual o efeito( Ex: Causar dano, adicionar atributos positivos a um herói, etc). Cada item terá seu próprio efeito, por isso ela é uma função virtual.

**Classe abstrata Armadura (Armadura.h e Armadura.cpp). Classe base: Equipamento**

Idem a descrição da classe **Arma**.

Armadura contém:

* Atributos da classe **Item** e da classe **Equipamento**.

Este método possui os seguintes construtores. Todos eles chamam o construtor da classe **Equipamento** para realizar o reaproveitamento de código, com exceção do destrutor:

* **Construtor default**, com valores padrão para um único bônus.
* **Construtor de cópia**, para copiar os valores de outra Armadura. Não é necessário validar os dados.
* **Construtor inicializando e validando todos os valores necessários.**
* **Destrutor virtual**, para chamar os destrutores das classes derivadas.

Além dos métodos “get()”, “set()” e operadores += e << para as variáveis acima, temos os métodos:

* **virtual void Add\_Bonus(),** que Adiciona um bônus com valor maior ou igual a 0 a um equipamento.
* **virtual void Efeito() = 0**, método virtual puro indicando qual o efeito( Ex: Causar dano, adicionar atributos positivos a um herói, etc). Cada item terá seu próprio efeito, por isso ela é uma função virtual.

**Classe abstrata Acessorio (Acessorio.h e Acessorio.cpp). Classe base: Equipamento**

Idem a descrição da classe **Arma e Armadura**, porém é mais comum de ser encontrada em Rpgs.

Acessorio contém:

* Atributos da classe **Item** e da classe **Equipamento**.

Este método possui os seguintes construtores. Todos eles chamam o construtor da classe **Equipamento** para realizar o reaproveitamento de código, com exceção do destrutor:

* **Construtor default**, com valores padrão para um único bônus.
* **Construtor de cópia**, para copiar os valores de outro Acessório. Não é necessário validar os dados.
* **Construtor inicializando e validando todos os valores necessários.**
* **Destrutor virtual**, para chamar os destrutores das classes derivadas.

Além dos métodos “get()”, “set()” e operadores += e << para as variáveis acima, temos os métodos:

* **virtual void Add\_Bonus(),** que Adiciona um bônus com valor maior ou igual a 0 a um equipamento.
* **virtual void Efeito() = 0**, método virtual puro indicando qual o efeito( Ex: Causar dano, adicionar atributos positivos a um herói, etc). Cada item terá seu próprio efeito, por isso ela é uma função virtual.

**Classes concretas Espada, Cajado, Faca, Roupas\_Leves, Anel\_De\_Forca.**

**Classe Base: Arma (Espada, Cajado, Faca), Armadura(Roupas\_Leves), Acessorio(Anel\_De\_Forcao)**

Classes concretas da hierarquia equipamento. Nota-se que elas também podem ser classes abstratas, servindo como base para outras classes (Ex: Espada de Fogo e/ou Armadura de Prata).

Todas estas classes possuem:

* Atributos da classe **Item** e da classe **Equipamento**.

Este método possui os seguintes construtores. Todos eles chamam o construtor da classe **Equipamento** para realizar o reaproveitamento de código, com exceção do destrutor:

* **Construtor default**, com valores padrão para um único bônus.
* **Construtor de vazio,** com valores pré-determinados**.**
* **Destrutor virtual**, para chamar os destrutores das classes derivadas.

Além dos métodos “get()”, “set()” e operadores += e << para as variáveis acima, temos os métodos:

* **void Add\_Bonus(),** que Adiciona um bônus com valor maior ou igual a 0 a um equipamento.
* **void Efeito() = 0**, método virtual puro indicando qual o efeito( Ex: Causar dano, adicionar atributos positivos a um herói, etc). Cada item terá seu próprio efeito, por isso ela é uma função virtual.