Projeto Final: Criando seu Próprio Sistema de Personagens de RPG

Objetivo: Construir um sistema simples em Java para criar e gerenciar personagens de um jogo de RPG. Você aprenderá a usar Classes, Objetos, Herança e Polimorfismo para criar um código organizado e reutilizável.

Ferramentas:

- Um ambiente de desenvolvimento Java (Recomendado: Eclipse, VS Code com Java ou Replit).

Etapa 1: A Base de Tudo - A Classe Personagem

Todo personagem, seja ele um guerreiro ou um mago, compartilha características básicas. Vamos criar uma classe "molde" para todos eles.

Conceitos: Classe, Atributos e Construtor.

1. **Crie a Classe Personagem:**
    - Crie um novo arquivo chamado `Personagem.java`.

Atributo	Tipo de Dado	Descrição
`nome`	`String`	O nome do personagem.
`nivel`	`int`	O nível de experiência.
`forca`	`int`	O atributo de força física.

2. **Crie o Construtor e os Métodos:**

   No arquivo Personagem.java:

```
public class Personagem {
    String nome;
    int nivel;
    int forca;

    public Personagem(String nome, int forca) {
        this.nome = nome;
        this.nivel = 1;
```

```
        this.forca = forca;
    }

    public void mostrarStatus() {
        System.out.println("Nome: " + this.nome);
        System.out.println("Nível: " + this.nivel);
        System.out.println("Força: " + this.forca);
    }

    public void atacar() {
        System.out.println(this.nome + " fez um ataque básico!");
    }
}
```

Etapa 2: Especialização - As Classes `Guerreiro` e `Mago`

Agora vamos usar **Herança** para criar tipos específicos de personagens. Eles vão "herdar" tudo da classe `Personagem` e adicionar suas próprias características.

Conceitos: Herança (`extends`), Subclasse, Sobrescrita (`@Override`).

1. **Crie a Classe `Guerreiro`:**

   No arquivo `Guerreiro.java`:

```
public class Guerreiro extends Personagem {
    int habilidadeComEspada;

    public Guerreiro(String nome, int forca, int habilidadeComEspada) {
        super(nome, forca); // Chama o construtor da classe pai
        this.habilidadeComEspada = habilidadeComEspada;
    }

    @Override // Sobrescrevendo o método atacar
    public void atacar() {
        System.out.println(this.nome + " atacou com sua espada!");
    }
}
```

2. **Crie a Classe `Mago`:**

   No arquivo `Mago.java`:

```
public class Mago extends Personagem {
    int poderMagico;

    public Mago(String nome, int forca, int poderMagico) {
        super(nome, forca); // Chama o construtor da classe pai
        this.poderMagico = poderMagico;
    }

    @Override // Sobrescrevendo o método atacar
    public void atacar() {
        System.out.println(this.nome + " lançou uma bola de fogo!");
    }
}
```

Etapa 3: Juntando Tudo - A Classe Principal `Jogo`

Agora vamos criar a classe principal que irá instanciar (criar os objetos) e colocar nossos personagens em ação.

Conceitos: Instanciação de Objetos, Método `main`.

1. **Crie a Classe `Jogo`:**

   No arquivo `Jogo.java`:

```
public class Jogo {
    public static void main(String[] args) {
        Guerreiro garen = new Guerreiro("Garen", 10, 8);
        Mago ryze = new Mago("Ryze", 4, 10);

        System.out.println("--- Início do Jogo ---");
        garen.atacar();
        ryze.atacar();
    }
}
```

Desafio Final: Expanda o Universo do Jogo!

Agora que você já criou a base do nosso sistema, é hora de testar seus conhecimentos. O desafio é adicionar novas funcionalidades ao projeto, aplicando os conceitos que aprendeu.

Parte 1: Crie uma Nova Classe - O `Arqueiro`

Siga o mesmo padrão que usamos para `Guerreiro` e `Mago` para criar uma nova classe de personagem.

1. **Crie a classe `Arqueiro`** que herda de `Personagem`.
2. Adicione um atributo específico para ele: `int precisao`.
3. Crie um construtor para `Arqueiro` que receba `nome`, `forca` e `precisao`. Lembre-se de usar `super()` para enviar os dados para a classe `Personagem`.
4. Sobrescreva (`@Override`) o método `atacar()` na classe `Arqueiro` para que ele exiba a mensagem: `[Nome do Arqueiro] disparou uma flecha!`.

Parte 2: Habilidades Especiais

Vamos deixar o sistema mais interessante. Cada classe de personagem terá uma habilidade especial única.

1. **Na classe `Personagem`**, adicione um novo método chamado `usarHabilidadeEspecial()`. A implementação dele na classe `Personagem` pode ser uma mensagem genérica como: `Usando habilidade especial...`.
2. **Sobrescreva o método `usarHabilidadeEspecial()` em cada uma das classes filhas** (`Guerreiro`, `Mago` e no seu novo `Arqueiro`) com uma mensagem personalizada:
    - **Guerreiro:** `Garen usou Fúria de Batalha!`
    - **Mago:** `Ryze canalizou uma Tempestade Arcana!`
    - **Arqueiro:** `Ashe usou Flecha de Cristal Encantada!`

Parte 3: A Prova Final

Modifique sua classe `Jogo` para incluir o novo personagem e testar as habilidades especiais.

1. Crie uma instância da sua nova classe `Arqueiro`.
2. Faça com que todos os seus personagens (`Guerreiro`, `Mago` e `Arqueiro`) usem o método `atacar()`.
3. Depois, faça com que todos eles usem o método `usarHabilidadeEspecial()`.

Dicas para o Sucesso:

- **Herança:** Lembre-se da palavra-chave `extends` para fazer uma classe herdar de outra.
- **Construtor da Subclasse:** A primeira linha dentro do construtor de uma subclasse deve ser a chamada `super(...)` para inicializar os atributos da classe pai.
- **Polimorfismo:** Use a anotação `@Override` sempre que for substituir um método da classe pai. Isso ajuda a evitar erros e deixa o código mais claro.

Desafio Extra (Para quem quer ir além):

Se você terminou tudo e quer um desafio maior, tente implementar um sistema de **pontos de vida (PV)**:

1. Adicione um atributo `pontosDeVida` na classe `Personagem`.
2. Modifique o método `atacar()` para que ele retorne um valor de dano (por exemplo, `return this.forca * 2;`).
3. Crie um novo método `receberDano(int dano)` na classe `Personagem` que subtraia o dano dos pontos de vida.
4. No `Jogo`, simule uma batalha: faça um personagem atacar o outro