Sistema de Casamento - Programação Orientada a Objetos

Aluno: João Paulo Silva Piauhy - Linguagem de Programação II

Introdução

O sistema de casamento apresentado é um exemplo de aplicação baseada na **Programação Orientada a Objetos (POO)**. Ele permite o gerenciamento de um casamento, incluindo o cadastro de convidados, tarefas e presentes. O sistema possui um fluxo de login para diferenciar as permissões entre noivos e convidados.

Conceitos de POO Utilizados

1. **Encapsulamento**: As informações de cada entidade (usuário, casamento, tarefas, etc.) são mantidas dentro de classes específicas.
2. **Herança**: Subclasses derivadas de `Usuario` para noivos e convidados, isso exemplificaria a herança.
3. **Polimorfismo**: Poderia ser aplicado para que métodos como `exibir_dados()` fossem sobrescritos em subclasses.
4. **Abstração**: As classes representam conceitos do mundo real, como `Usuario`, `Tarefa` e `Presente`.
5. **Composição**: A classe `Casamento` agrega `Usuario`, `Tarefa` e `Presente`, demonstrando uma relação de "tem um".

Estrutura do Sistema

1. Classe `SistemaCasamento`

Gerencia o funcionamento geral do sistema, incluindo login, criação do casamento e a adição de convidados, tarefas e presentes.

Principais métodos:

- `mostrar_menu()`: Exibe o menu apropriado com base no estado de login.
- `login(email, senha)`: Permite que usuários façam login no sistema.
- `verificar_permissao_noivo()`: Verifica se o usuário atual tem permissão de noivo.
- `criar_casamento(data, local)`: Cria um novo casamento com data e local definidos.

- `adicionar_convidado(convidado)`: Adiciona um convidado à lista de convidados do casamento.
- `adicionar_tarefa(tarefa)`: Registra uma nova tarefa.
- `adicionar_presente(presente)`: Adiciona um presente à lista de presentes.

2. Classe `Usuario`

Representa um usuário do sistema, que pode ser um noivo ou um convidado.

Atributos:

- `nome`: Nome do usuário.
- `email`: Email do usuário.
- `senha`: Senha para login.
- `eh_noivo`: Define se o usuário é noivo ou convidado.

Métodos:

- `exibir_dados()`: Retorna uma string com os dados do usuário.

3. Classe `Tarefa`

Representa uma tarefa relacionada ao casamento.

Atributos:

- `descricao`: Descrição da tarefa.
- `responsavel`: Pessoa responsável pela execução da tarefa.
- `status`: Status da tarefa (Pendente, Em andamento, Concluída).

Métodos:

- `__init__()`: Inicializa uma nova tarefa com status padrão "Pendente".

4. Classe `Presente`

Gerencia a lista de presentes do casamento.

Atributos:

- `descricao`: Nome ou descrição do presente.
- `loja`: Loja onde o presente pode ser encontrado.
- `preco`: Valor do presente.

5. Classe `Casamento`

Armazena informações sobre o casamento e gerencia tarefas, convidados e presentes.

Atributos:

- `data`: Data do casamento.
- `local`: Local onde ocorrerá o casamento.
- `lista_convidados`: Lista de convidados associados ao casamento.
- `tarefas`: Lista de tarefas relacionadas ao casamento.
- `lista_presentes`: Lista de presentes cadastrados.

Métodos:

- `adicionar_convidado(convidado)`: Adiciona um convidado à lista.
- `adicionar_tarefa(tarefa)`: Adiciona uma nova tarefa ao casamento.
- `adicionar_presente(presente)`: Adiciona um presente à lista.

Funcionamento do Sistema

1. O sistema inicia e exibe um menu.
2. Se um usuário não estiver logado, ele pode fazer login.
3. Se for um noivo:
    - Pode criar um casamento e gerenciar tarefas, convidados e presentes.
    - Pode visualizar e atualizar tarefas.
4. Se for um convidado:
    - Pode visualizar a lista de presentes.
5. O sistema trata erros como tentativas de acesso sem um casamento criado.

O sistema usa um **loop de interação** para permitir que o usuário selecione opções até optar por sair.