As classes criadas estão organizadas em subpacotes, seguindo uma arquitetura modular. Vou explicar como cada conjunto de classes interage e como contribuem para o funcionamento do sistema.

**1. Subpacote modelos**

Este subpacote contém as classes principais do sistema que representam os elementos do domínio do problema, como **frações**, **apartamentos**, **lojas**, **garagens**, **arrecadações** e o próprio **condomínio**.

**Relações Internas**

* **Fracao**:
  + É uma classe abstrata que define as propriedades e comportamentos básicos de todas as frações (e.g., identificador, área, percentagem).
  + As subclasses (**Apartamento**, **Loja**, **Garagem**, **Arrecadacao**) implementam o método abstrato calcularQuotaMensal de forma específica, considerando as regras de cálculo de cada tipo de fração.
* **Condominio**:
  + Gerencia as frações (List<Fracao>), permitindo adicionar, remover e listar frações.
  + Executa cálculos globais, como a soma das quotas mensais ou a verificação das percentagens das frações.
  + Interage com as subclasses de Fracao ao calcular as quotas e ao atualizar percentagens com base na área total.

**Exemplo de Interação**

1. O sistema cria um Condominio.
2. Frações específicas, como Apartamento ou Loja, são instanciadas e adicionadas ao condomínio.
3. O método calcularSomaQuotasMensais do condomínio acessa cada fração para calcular sua contribuição total com base nos valores globais (despesaGeral e despesaElevadores).

**2. Subpacote util**

O subpacote util fornece funções auxiliares que tornam o sistema mais robusto e organizado, facilitando validações, formatações e persistência de dados.

**Relações com Outros Subpacotes**

* **Validacao**:
  + É usada pelas classes de modelos (e.g., Fracao, Condominio) para garantir que os dados fornecidos estão corretos. Por exemplo:
    - Verifica se os textos não estão vazios.
    - Valida se percentagens estão no intervalo de 0 a 100%.
* **Formatacao**:
  + É usada para exibir números (e.g., percentagens e quotas) com 3 casas decimais de forma padronizada.
  + Pode ser chamada em métodos toString das classes para melhorar a apresentação dos dados.
* **Ficheiros**:
  + Gerencia a persistência dos dados no sistema. As classes como Condominio utilizam os métodos salvarDados e carregarDados para armazenar e recuperar frações ou outros objetos do sistema.

**Exemplo de Interação**

1. Ao criar uma nova Fracao, a classe Validacao pode verificar se os dados inseridos (como área ou percentagem) são válidos.
2. Ao salvar ou carregar frações em ficheiros, o método Ficheiros.salvarDados será chamado pelo condomínio, passando a lista de frações.

**Exemplo Prático do Fluxo Completo**

**1. Inicialização**

* Ao iniciar o sistema, o método Ficheiros.carregarDados é chamado para carregar as frações salvas em execuções anteriores.
* Se não houver dados salvos, o sistema inicia com um condomínio vazio.

**2. Adicionar uma Fração**

* O usuário insere os dados de uma nova fração.
* **Validacao** verifica os dados (e.g., texto válido, percentagem entre 0 e 100).
* A fração é instanciada (e.g., um novo Apartamento) e adicionada ao condomínio.

**3. Atualização de Percentagens**

* Sempre que uma fração é adicionada ou removida, o método Condominio.recalcularPercentagens é chamado para ajustar as percentagens de todas as frações com base na nova área total.

**4. Calcular Quotas**

* O sistema usa o método Condominio.calcularSomaQuotasMensais, que chama calcularQuotaMensal de cada fração para determinar a contribuição de cada uma.

**5. Exibição de Dados**

* Ao listar as frações ou exibir quotas, os números são formatados pela classe Formatacao.

**6. Salvamento de Dados**

* Ao encerrar o programa, o método Ficheiros.salvarDados é chamado, garantindo que os dados persistam para a próxima execução.

**Resumo do Funcionamento**

1. **Modelos (modelos)**: Representam os dados e a lógica central do sistema.
2. **Utilitários (util)**: Fornecem funções auxiliares para validação, formatação e persistência.
3. **Interações**:
   * O Condominio utiliza as frações (e suas subclasses) para gerenciar dados e executar cálculos.
   * As classes de util garantem que os dados manipulados nos modelos sejam válidos, bem formatados e persistam adequadamente.

Na **classe principal** (geralmente chamada de Main ou algo similar), todas essas classes são usadas para criar um fluxo lógico do sistema, integrando as funcionalidades de gerenciamento do condomínio, validações, cálculos e persistência. Aqui está um exemplo de como organizar e usar as classes na classe principal:

**Fluxo do Sistema**

1. **Carregamento Inicial**:
   * Os dados do condomínio são carregados do ficheiro.
   * Se o ficheiro não existir, cria-se um novo condomínio vazio.
2. **Menu Principal**:
   * Oferece opções para gerenciar o condomínio:
     + Adicionar frações.
     + Listar frações.
     + Calcular quotas mensais.
     + Verificar percentagens.
     + Atualizar despesas.
     + Salvar e sair.
3. **Interações**:
   * As classes Condominio, Fracao (e subclasses) realizam a lógica principal.
   * As classes de util validam, formatam e salvam/carregam dados.

Se precisar de mais detalhes ou ajustes, é só pedir!