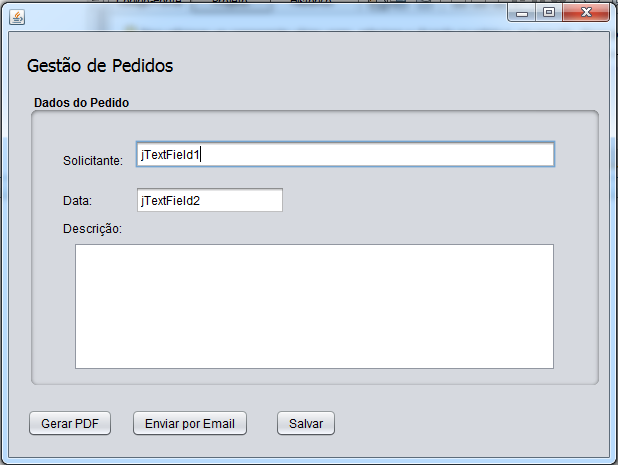
**Padrão de Projetos Estruturais**

**Composite**

Compor objetos em estrutura de árvore para representarem hierarquias todo-parte. Composite permite tratar de maneira uniforme objetos individuais e composições de objetos.

Tipicamente modelos Composite trabalham com sistemas de nós que são chamados de árvore. Em uma árvore de objetos podemos ter nós do tipo galho, que poderão conter outros nós, e nós do tipo folha, que não podem conter outros nós.

Exemplo:



**JFrame**

**Painel**

**jtxtSolicitante**

**jtxtData**

**jtxtDescricao**

**btnSalvar**

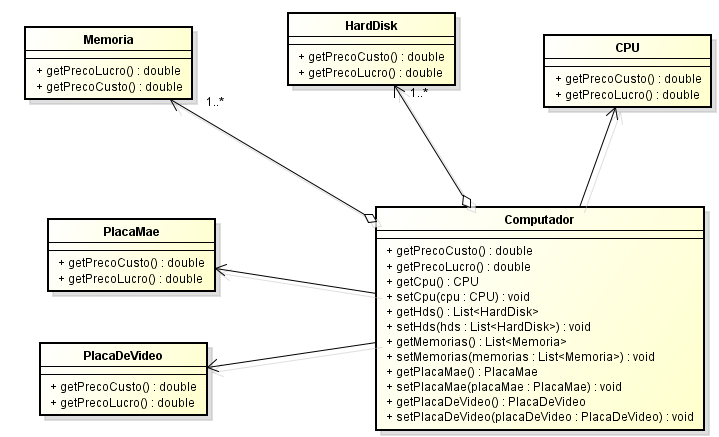
**btnGerarPDF**

**btnEnviarPorEmail**

Vamos imaginar que estamos montando um sistema de vendas de computadores que são configurados conforme as necessidades do cliente. Neste caso, teríamos os componentes comuns de um computador: placa-mãe,memória, hard-disk, CPU e placa de vídeo.

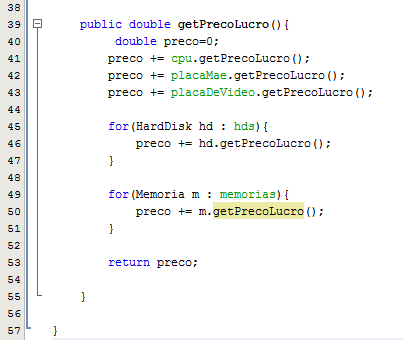
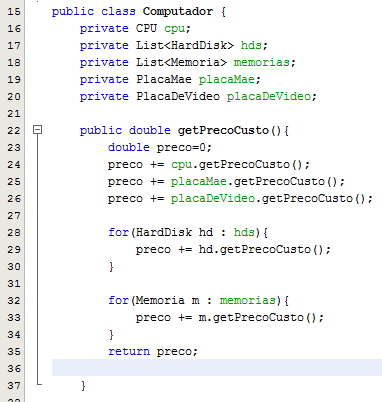
Neste caso, sem aplicar o design pattern composite e sem grandes critérios técnicos, podemos imaginar o seguinte modelo como anti-pattern do Composite:

**Anti-Padrão**



O modelo apresenta uma classe Computador que esta diretamente associada com todos seus componentes. Para adicionar um novo componente no computador precisaremos alterar a interface da classe Computador,adicionando a associação e também teremos que mudar a regra de cálculo de preços para que considere este novo componente.

**Código da classe Computador**

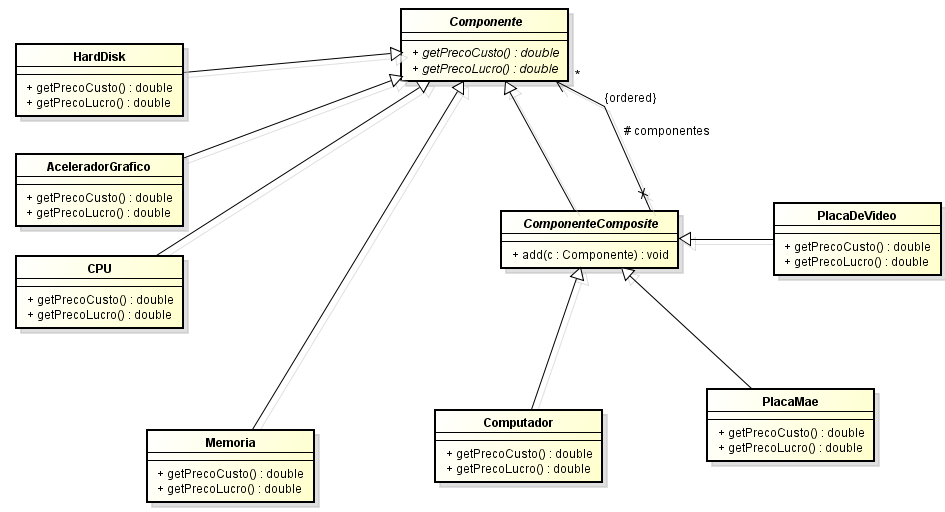


**Aplicando o Padrão**

Quando aplicamos o design pattern Composite, criamos uma estrutura de classes e sub-classes que serão divididas em:

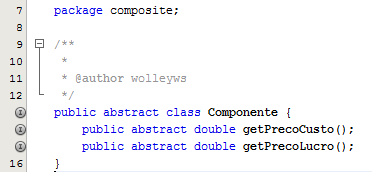
* **Classes composite (nó Galho):** que poderão conter outros componentes;
* **Classes leaf (nó Folha):** que serão componentes finais na hierarquia;

Após o refatoramento do exemplo do anti-pattern, teríamos a seguinte hierarquia de classes:

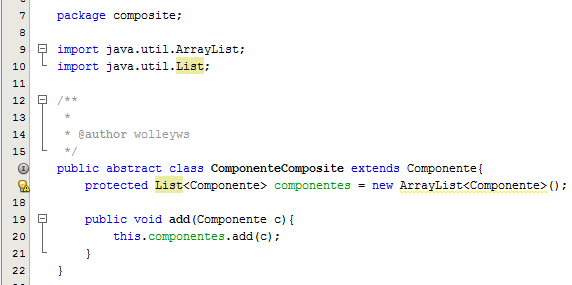


Observe que criamos uma classe Componente e uma classe ComponenteComposite, desta forma se estendermos Componente, criaremos um objeto **Leaf (folha)**, se estendermos a classe ComponenteComposite, criaremos um componente que será container de outros componentes, compostos ou não. Esse modelo permite maior facilidade de extensão e facilidades para aplicar comportamento dinâmico em combinação com outros design patterns.

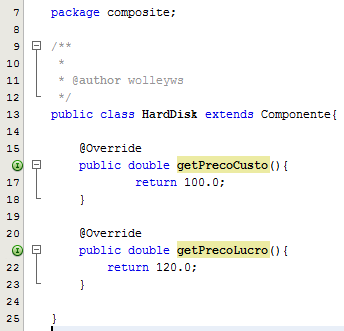
**Classe Componente**



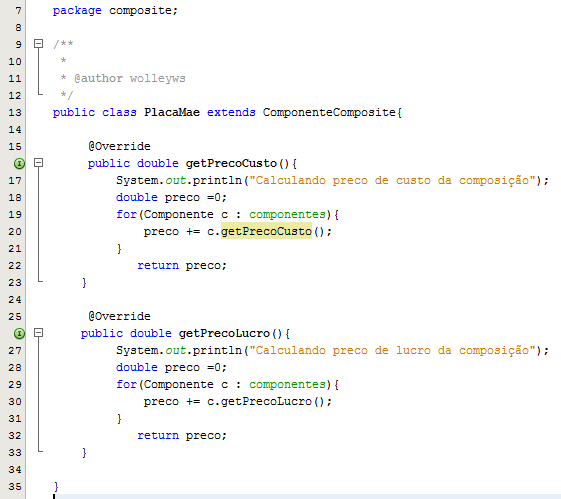
**Classe ComponenteComposite**



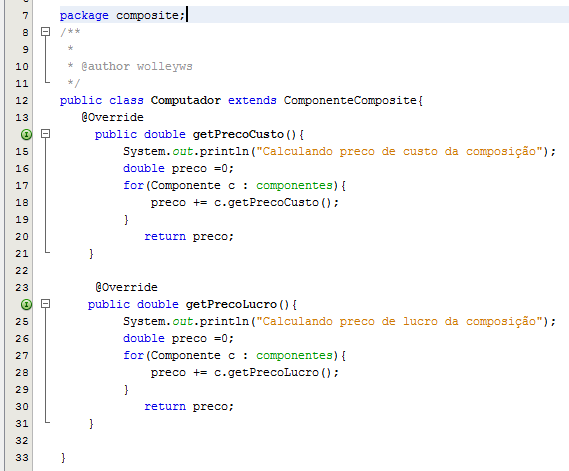
**A sub-classe HardDisk é um exemplo de implementação de objeto leaf (folha):**



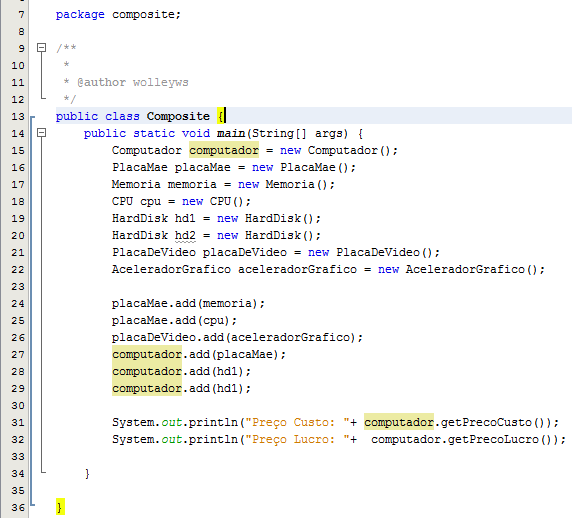
**A classe PlacaMae um exemplo de Composite:**



**O código da classe Computador seria refatorado para:**



**Para finalizar a classe de Exemplo de uso**



Responda as seguintes questões:

1. Quais foram as classes **Leaf** (folha) criadas neste exemplo?
2. Quais foram as classes que herdaram de ComponenteComposite?
3. Quais benefícios foram alcançados com este design de classes?

**Referencias**

Livro: Padrões de Projeto – GoF

Disponível em: http://online.minhabiblioteca.com.br/#/books/9788577800469

Livro: 33 Desing Pattern aplicados com Java – GlobalCode

Disponível em: http://www.slideshare.net/vsenger/33-design-patterns-com-java