Subject-Oriented Programming

Programação Orientada a Sujeitos

Augusto César Melo de Oliveira
Sabrina Bet

Introdução

- Orientação a Objetos não era suficiente.
- 1993 - SOP
  - Harrison e Ossher - IBM
- SOP complemento POO

Decomposição

- Quebrar um problemas em vários problemas.
  - Facilidade
  - Distribuição de Responsabilidades
- Recomposição
- SOP = flexibilidade
- Diferentes pontos de vista (visões)

Paradigmas SOP

- Baseado em Característica
- Estágio
- Extensão imprevista
- Composição imprevista

Problemas

- Criar extensões e configurações
  - sem modificação código original
  - encapsular deltas para múltiplas plataformas, versões e características
- Integrar sistemas e componentes

Problemas

- Desenvolvimento de sistemas em diferentes grupos
- Simplificação de Design patterns

Paradigmas SOP

- Reduzir Complexidade
- Melhorar Compreensão
- Limitar impacto das Mudanças
- Reutilização

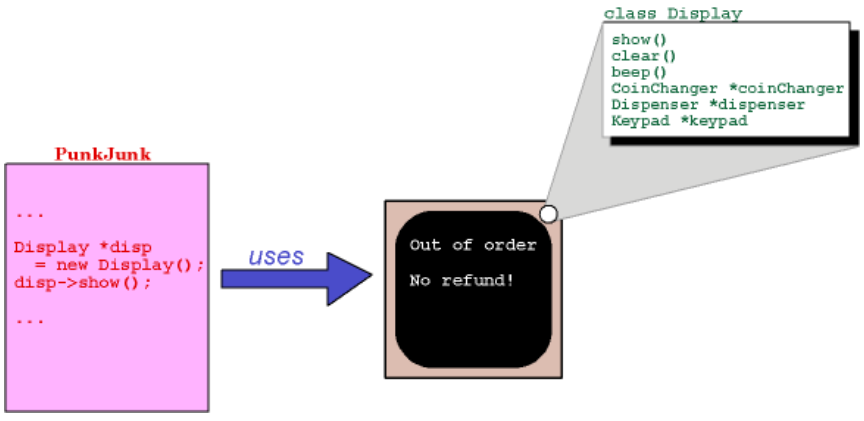
Conceitos

- Tecnologia de composição
- **Sujeito:** modelo de um domínio sob um ponto de vista.
- **Composição de Sujeitos:** combinação de sujeitos produzindo um novo sujeito.
- **Regra de Composição:** especificam como os sujeitos serão compostos.

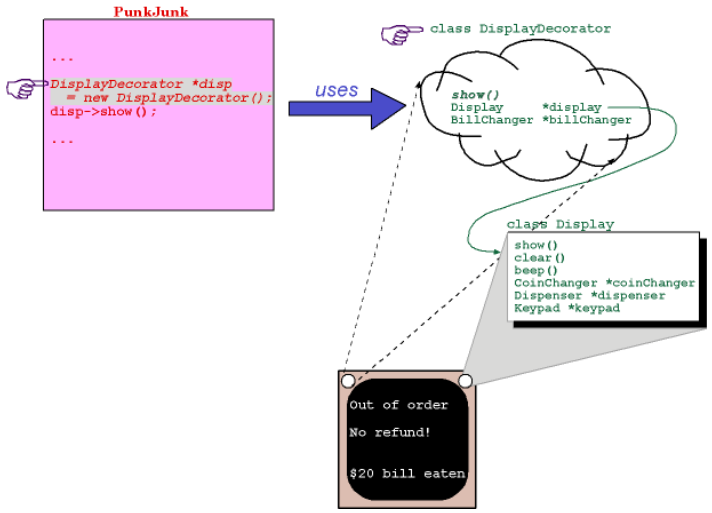
Pattern Decorator

- Exemplo PunkJunk - Dewey.
- Máquina de troco.

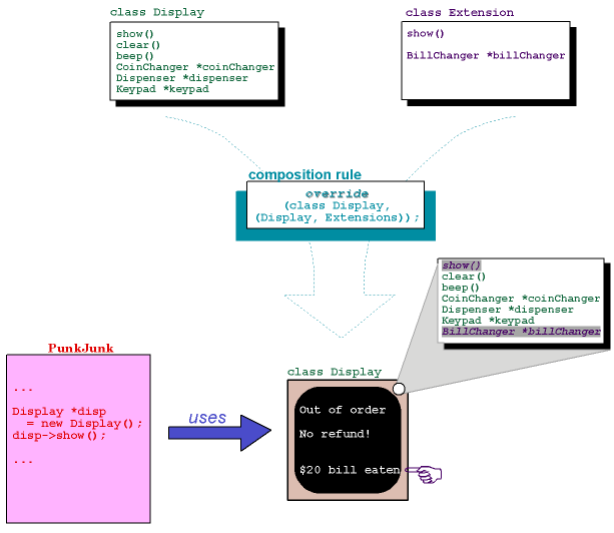
Pattern Decorator



Pattern Decorator



Pattern Decorator



Ferramentas

- Tecnologia independente de linguagem.
- Ferramentas IBM:
  - VAC++ (C++)
  - HyperJ (Java)
  - Protótipo para Smaltalk

Multi-Dimensional Separation of Concerns (MDSC)

- Generalização do SOP
  - Possui regras de integração mais precisas que SOP
- Permite a identificação e encapsulamento de todos os tipos de interesses (concerns)
  - Interesses podem ser identificados a qualquer momento durante o ciclo de vida do sistema
  - Permite a co-existência de múltiplas decomposições, relacionamentos entre módulos, representação de interesses que se sobrepõem
  - Módulos são similares a sujeitos em SOP. Podem representar estruturas completas (hierarquias de classes) ou componentes individuais (classes, aspectos, etc.)

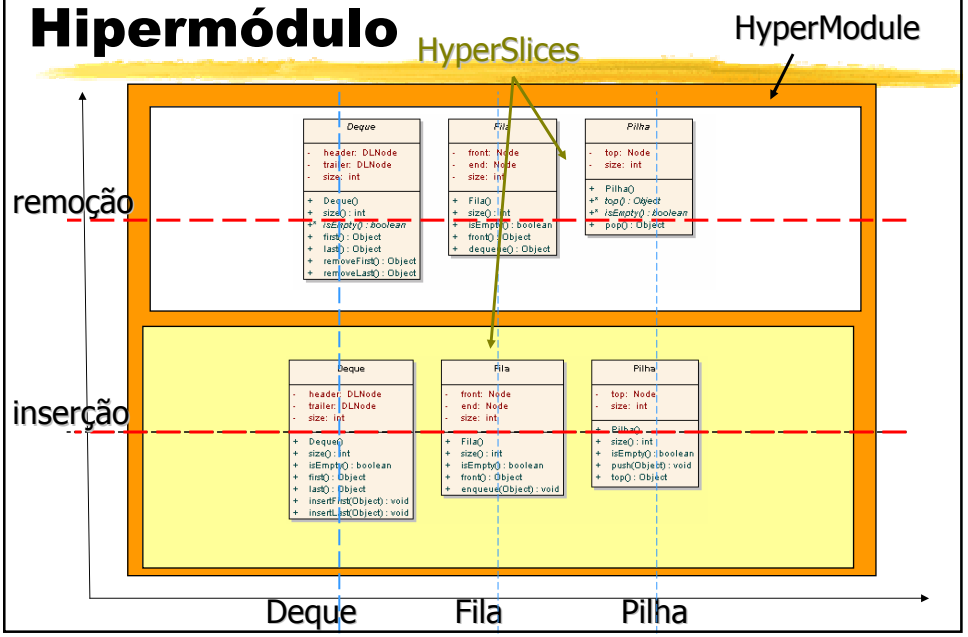
HyperJ

- Ferramenta open-source da IBM
- Ambiente que implementa HyperSpaces em Java
- Com Hyper/J é possível representar diferentes dimensões de interesse para código Java
- Suporta a separação e integração de interesses.
  - Decomposição
  - Composição
- Unidades suportadas são pacotes Java, interfaces, classes, métodos, atributos e construtores
- Gera arquivos .class para todos os hypermodules criados
- Hyper/J não interfere no código existente
- Não é preciso recompilar nada

Usando Hyper/J - passo-a-passo

1) Insira código existente no hiperespaço (hyperspace) identificando os interesses importantes
2) Implemente novos recursos em Java
3) Insira novo código no hiperespaço encapsulado como um novo interesse
4) Crie um hipermódulo e acrescente a ele os interesses desejados
5) Indique os relacionamentos de composição(regras de composição) entre os interesses no hipermódulo

Hipermódulo



Arquivo de Especificação do Hipermódulo

Arquivos que serão compostos

```
-hyperspace
  hyperspace ContainerHyperspace
      composable class container.*;
      composable class container.sort.*;
```

Mapeamento de Interesses

```
-concerns
  package container : Feature.container
  package container.sort : Feature.sort
```

Relacionamentos (Regras) de Composição

```
-hypermodules
   hypermodule ContainerSCE
     hyperslices:
        Feature.container,
        Feature.sort;
     relationships:
        mergeByName;
   end hypermodule;
```

Conclusões

- **MDSC** que ingloba a **SOP** permite o encapsulamento de quaisquer interesses em estruturas que são reutilizadas através de composição.
- Hyper/J possibilita
  - Separação de interesses em hyperslices, que podem ser definidos arbitrariamente
  - Composição de hyperslices em hipermódulos, que podem ser usados para gerar código 100% Java
  - Definição de regras de composição entre unidades
  - Introdução de novos recursos e interesses sem que seja necessário ter acesso ou modificar o código-fonte original
- Hyper/J pode ser usado tanto em projetos novos como em projetos existentes

Material

- http://www.inf.ufsc.br/~sbet/sop