DCC / ICEx / UFMG

Técnicas de Implementação para Linhas de Produtos

Eduardo Figueiredo

http://www.dcc.ufmg.br/~figueiredo

Técnicas de Implementação

- Compilação condicional
  - Antenna
- Programação orientada a aspectos
  - AspectJ
- Programação orientada a características
  - AHEAD

Compilação Condicional

Compilação Condicional

- Consiste na anotação de trechos de código associados a uma determinada característica
  - Estas anotações são interpretadas por um pré-processador que decide sobre a inclusão do código no produto
- Exemplos comuns
  - *#ifdef*, *#else*, *#endif*

Exemplo de Uso (MobileMedia)

```
public class PhotoListScreen extends List {

  public static final Command viewCommand;
  public static final Command addCommand;
  public static final Command deleteCommand;
  public static final Command editLabelCommand;
  // #ifdef includeSorting
  public static final Command sortCommand;
  // #endif
  // #ifdef includeFavourites
  public static final Command favoriteCommand;
  public static final Command viewFavoritesCommand;
  // #endif
...
}
```

O código que implementa as características opcionais delimitado por compilação condicional.

Exemplo de Uso (MobileMedia)

```
public class PhotoListScreen extends List {

  public static final Command viewCommand;
  public static final Command addCommand;
  public static final Command deleteCommand;
  public static final Command editLabelCommand;
```

S

```
  // #ifdef includeSorting
  public static final Command sortCommand;
  // #endif
```

F

```
  // #ifdef includeFavourites
  public static final Command favoriteCommand;
  public static final Command viewFavoritesCommand;
  // #endif
...
}
```

Exemplo de Uso (MobileMedia)

```
public class PhotoListScreen extends List {
  ...
  public void initMenu() {
    this.addCommand(viewCommand);
    this.addCommand(addCommand);
    this.addCommand(deleteCommand);
    this.addCommand(editLabelCommand);
    // #ifdef includeSorting
    this.addCommand(sortCommand);
    // #endif
    // #ifdef includeFavourites
    this.addCommand(favoriteCommand);
    this.addCommand(viewFavoritesCommand);
    // #endif
  }
}
```

O código que implementa as características pode estar dentro de métodos.

Exemplo de Uso (MobileMedia)

```
public class PhotoListScreen extends List {
  ...
  public void initMenu() {
    this.addCommand(viewCommand);
    this.addCommand(addCommand);
    this.addCommand(deleteCommand);
    this.addCommand(editLabelCommand);
    // #ifdef includeSorting
    this.addCommand(sortCommand);
    // #endif
    // #ifdef includeFavourites
    this.addCommand(favoriteCommand);
    this.addCommand(viewFavoritesCommand);
    // #endif
  }
}
```

S

F

Configuração

- Considerando apenas
  - Uma característica mandatória (Central)
  - Duas características opcionais (Ordenação e Favoritos)

Instâncias	Central	Ordenação	Favoritos
Produto 1	Sim	Sim	Sim
Produto 2	Sim	Sim	Não
Produto 3	Sim	Não	Sim
Produto 4	Sim	Não	Não

Como é feita a configuração

- Várias formas...
- Exemplo: é dito ao pré-processador quais características (símbolos) devem ser incluídos no produto final

S F preprocessor.symbols = central, includeSorting, includeFavourites

S preprocessor.symbols = central, includeSorting

F preprocessor.symbols = central, includeFavourites

preprocessor.symbols = central

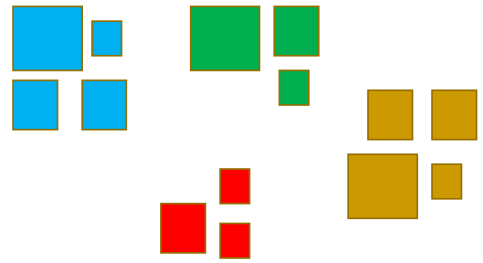
Programação Orientada a Aspectos (POA) para LPS

AspectJ

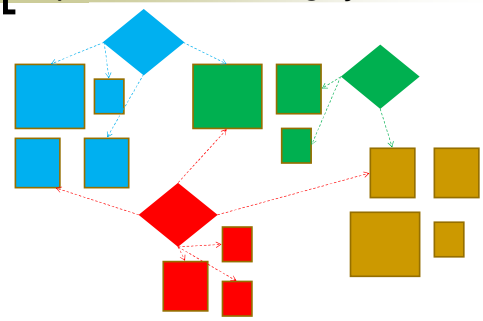
POA para LPS

- Aspectos podem ser usados para
  - Separar características transversais
  - Compor características do produto
- Cada característica variável pode ser implementada por um conjunto de classes e aspectos
  - A extração de características para aspectos exige identificação do código que implementa esta característica

Características Modularizadas

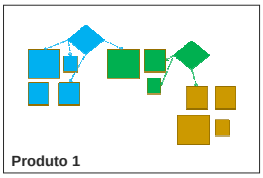


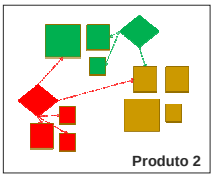
Aspectos fazem a ligação



Configuração

- Feito pela inclusão ou exclusão das classes e aspectos que implementam uma característica





Exemplo de Uso (MobileMedia)

```
public class PhotoListScreen extends List {
  public static final Command viewCommand;
  public static final Command addCommand;
  public static final Command deleteCommand;
  public static final Command editLabelCommand;
  ...
}
```

```
public aspect SortingAspect {
  public static final Command sortCommand;
  pointcut initMenu(PhotoListScreen screen):
    execution(public void PhotoListScreen.initMenu()) && this(screen);
  after(PhotoListScreen screen) : initMenu(screen) {
    screen.addCommand(sortCommand);
  }
  ...
}
```

Programação Orientada a Características (FOP)

AHEAD

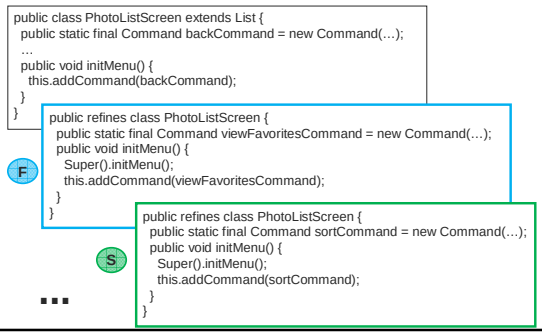
Definições

- Programação Orientada a Características (FOP) é um paradigma de programação para desenvolvimento de linha de produtos de software
- Uma característica é um incremento funcional no desenvolvimento do software

Refinamento Sucessivo

- O código deve ser sucessivamente refinado para posterior composição
  - Enfatiza a simplicidade e facilidade de compreensão
- Cada característica é chamada de refinamento

Exemplo de Uso (MobileMedia)



Bibliografia da Aula

- Artigos
  - E. Figueiredo, *et al*. **Evolving Software Product Lines with Aspects: An Empirical Study on Design Stability**. International Conference on Software Engineering (ICSE), 2008. (CC e POA)
  - G. Ferreira, F. Gaia, E. Figueiredo e M. Maia. **On the Use of Feature-Oriented Programming for Evolving Software Product Lines - A Comparative Study**. Simpósio Brasileiro de Linguagens de Programação (SBLP), 2011. (POA e FOP)