DCC / ICEx / UFMG

Reutilização em Programação Orientada a Objetos

Eduardo Figueiredo

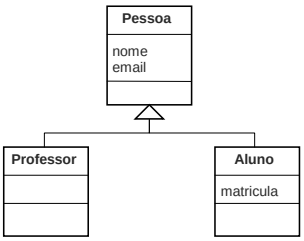
http://www.dcc.ufmg.br/~figueiredo

Técnicas de Reutilização

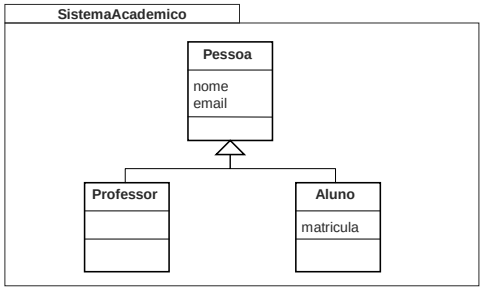
- Reuso de classes
- Bibliotecas
- Frameworks

Reuso de Classes

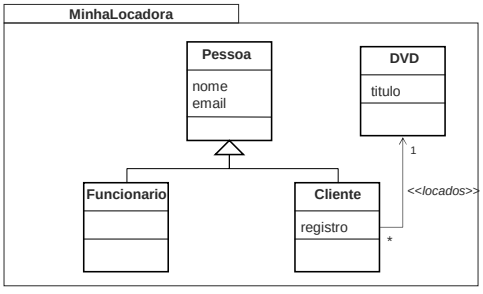
Considere três classes...



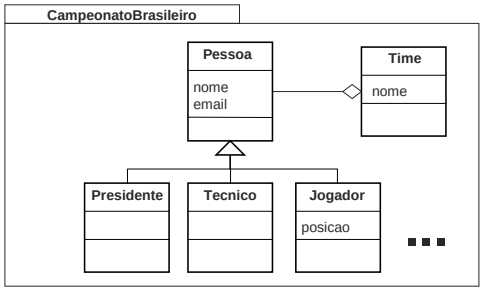
Sistema Acadêmico



Locadora de DVD



Campeonato de Futebol



Bibliotecas

Bibliotecas de Software

- Bibliotecas implementam serviços que podem ser usados por programas
  - É uma forma bem comum de reuso
- Disponibiliza funcionalidades comuns a diferentes tipos de sistemas
  - Converter informação entre formatos conhecidos (e.g., string para inteiro)
  - Acesso a recursos, arquivos, BD, etc.
  - Tipos abstratos de dados: fila, pilha, lista...

Use de Biblioteca em Java

```
import java.util.Vector;

public class Customer {

  String name;
  Vector phoneNumbers = new Vector();

  void removePhoneNumber(String c){
    phoneNumbers.removeElement(c);
  }

  void addPhoneNumber(String c){
    phoneNumbers.addElement(c);
  }

  ...
}
```

Importar Classe da Biblioteca

```
import java.util.Vector;

public class Customer {

  String name;
  Vector phoneNumbers = new Vector();

  void removePhoneNumber(String c){
    phoneNumbers.removeElement(c);
  }

  void addPhoneNumber(String c){
    phoneNumbers.addElement(c);
  }

  ...
}
```

A classe Vector é incorporada ao sistema.

Instanciar um Objeto da Biblioteca

```
import java.util.Vector;

public class Customer {

  String name;
  Vector phoneNumbers = new Vector();

  void removePhoneNumber(String c){
    phoneNumbers.removeElement(c);
  }

  void addPhoneNumber(String c){
    phoneNumbers.addElement(c);
  }

  ...
}
```

Um objeto da classe Vector é criado da mesma forma que qualquer outro objeto do sistema.

Acessar Funções da Biblioteca

```
import java.util.Vector;

public class Customer {

  String name;
  Vector phoneNumbers = new Vector();

  void removePhoneNumber(String c){
    phoneNumbers.removeElement(c);
  }

  void addPhoneNumber(String c){
    phoneNumbers.addElement(c);
  }

  ...
}
```

As funcionalidades *removeElement* e *addElement* estão implementadas na biblioteca.

Produtividade e Confiabilidade

- Desenvolvedores não tem que "re-inventar a roda"
  - Alguns funcionalidade estão disponíveis
- Bibliotecas são bem testadas por muitos usuários
  - Mesmos situações pouco usuais e valores extremos já foram explorados

Principais Desvantagens

- Difícil fazer adaptações para atender detalhes específicos do sistema
  - Melhor performance
  - Mais robusto
- Tempo e custo para aprender ou encontrar a funcionalidade desejada

Frameworks

Motivação

- Objetos e funções (em bibliotecas) são geralmente pequenos e especializados demais
  - Torna-se necessário o reuso de abstrações maiores, como *frameworks*
- *Framework* é um conjunto de classes e interfaces que formam uma estrutura genérica

Framework

- *Frameworks* são aplicações incompletas
  - São formados por interfaces, classes abstratas e classes concretas
  - Permitem criar vários aplicações diferentes
- Detalhes específicos do sistema devem ser implementados
  - Pela adição de novas classes
  - Pela implementação das classes abstratas
  - Por arquivos de configuração, etc.

Características Principais

- Estensibilidade
  - Framewoks são extensíveis
- Inversão de controle
  - Diferente de bibliotecas, o fluxo de controle da aplicação é geralmente coordenada pelo framework
- Código não pode ser modificado
  - Diferente das aplicações, o código do *framework* não deve ser modificado

Extensão de *Frameworks*

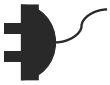
- *Frameworks* são entidades grandes que devem ser estendidas
- Exemplos de extensão
  - Adição de classes concretas que herdam operações de classes abstratas
  - Adição (sobrescrita) de métodos que respondem os eventos conhecidos do framework

Frozen Spots e *Hot Spots*

- Frameworks são constuídos de *frozen spots* e *hot spots*
- *Frozen spots* definem a arquitetura principal do *framework*
  - Eles não são alterados em nenhuma extensão do *framework* (congelados)
- *Hot spots* definem os locais que programadores usam para extensão

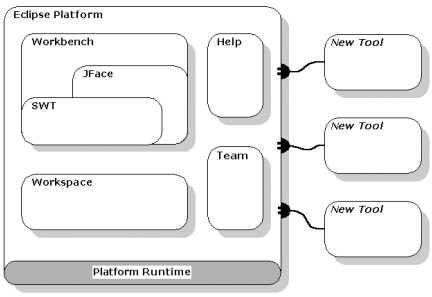
Representação (*Hot Spot*)

Ponto de extensão do framework



Extensão da aplicação

Exemplo de Extensão



Principal Problema

- *Framework* são geralmente entidades grandes e complexas
  - Leva um longo tempo para entendê-los e usá-los efetivamente
- Em grandes organizações, é comum ter profissionais especialistas em *frameworks* específicos

Bibliografia da Aula

- Ian Sommerville. **Engenharia de Software**, 9ª Edição. Pearson Education, 2011.
  - Cap. 16 Reuso de Software