



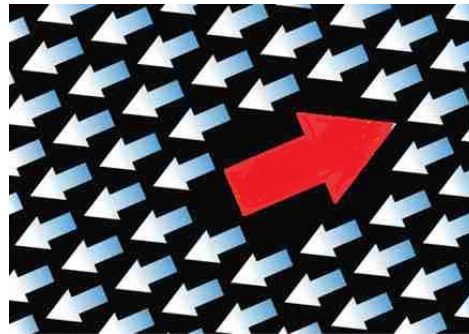
INVERSÃO DE CONTROLE E INJEÇÃO DE DEPENDÊNCIA

Marum Simão Filho

AGENDA

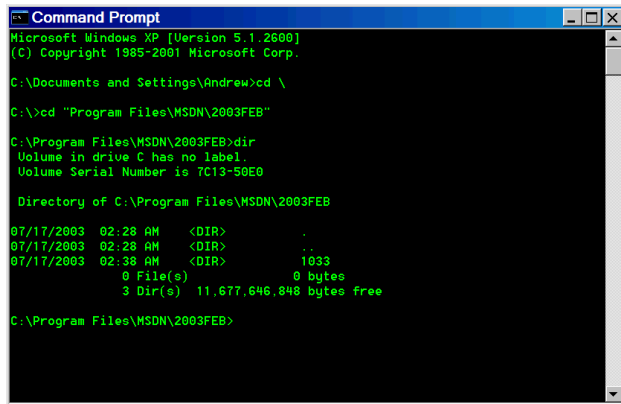
- Inversão de Controle (IoC)
- Injeção de Dependência (DI)

INVERSÃO DE CONTROLE



INVERSÃO DE CONTROLE

- Compare uma aplicação de linha de comando e outra que utiliza janelas em relação ao **fluxo de execução**.



```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Andrew>cd \

C:\>cd "Program Files\MSDN\2003FEB"

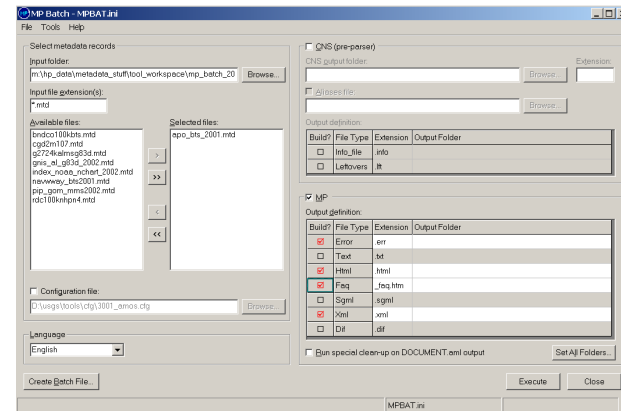
C:\Program Files\MSDN\2003FEB>dir
Volume in drive C has no label.
Volume Serial Number is 7C13-50E0

Directory of C:\Program Files\MSDN\2003FEB

07/17/2003  02:28 AM  <DIR>          .
07/17/2003  02:28 AM  <DIR>          ..
07/17/2003  02:38 AM  <DIR>          1633
               0 File(s)            0 bytes
               3 Dir(s)  11,677,646,848 bytes free

C:\Program Files\MSDN\2003FEB>
```

X



O programador
tem controle
sobre o fluxo

O programador
NÃO tem
controle sobre
o fluxo

INVERSÃO DE CONTROLE

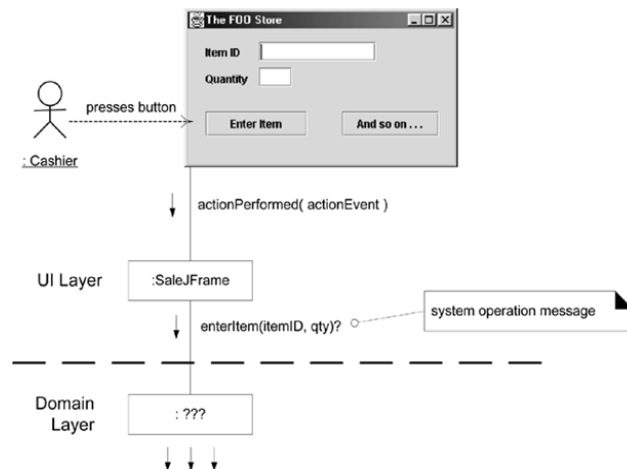
- Um fluxo normal de execução acontece quando um determinado programa cria chamadas para outros programas e assim sucessivamente, ficando a criação dos componentes, o início da execução e o fim da execução sob o controle do programador.
- A inversão de controle ocorre quando, ao invés de se criar explicitamente um código, ou acompanhar todo o ciclo de vida de uma execução, o programador delega alguma dessas funcionalidades para um terceiro.

INVERSÃO DE CONTROLE

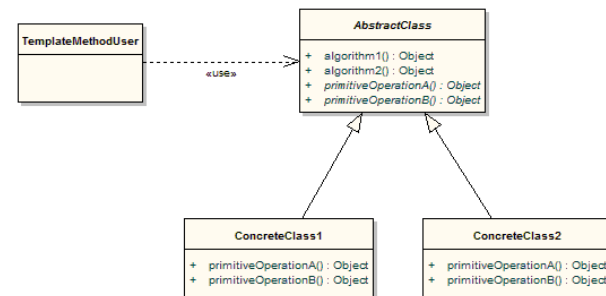
- **Inversão de controle** (*Inversion of Control* ou IoC, em inglês) é o nome dado ao padrão de desenvolvimento de programas de computadores onde a sequência (controle) de chamadas dos métodos é invertida em relação à programação tradicional, ou seja, ela **não** é determinada diretamente pelo programador.
- Este controle é delegado a uma infraestrutura de *software*, muitas vezes chamada de **container**, ou a qualquer outro componente que possa tomar controle sobre a execução, como os **frameworks**, por exemplo.
- Também conhecido como **Princípio de Hollywood**: “não nos chame, nós chamaremos você”. (“*Don't call us, we'll call you*”).

INVERSÃO DE CONTROLE

Exemplos de Inversão de Controle



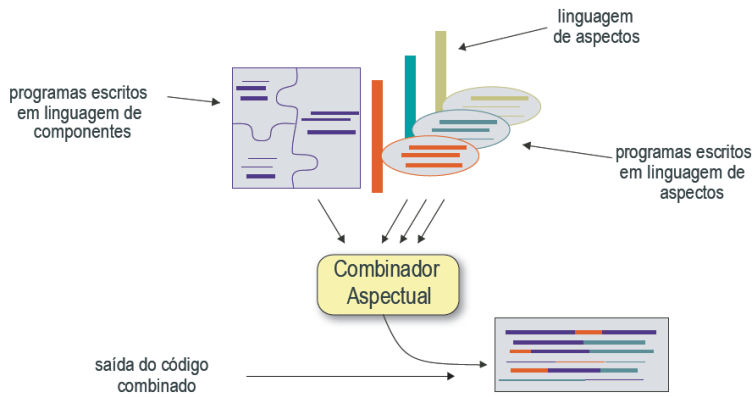
Programação Orientada a Eventos (GUI)



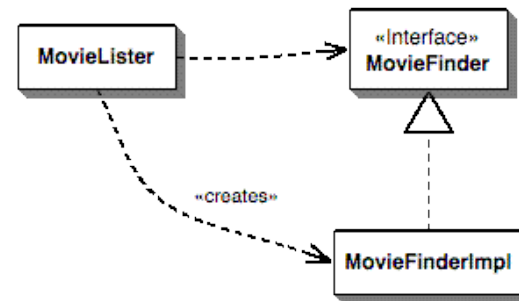
Padrão Template Method

INVERSÃO DE CONTROLE

○ Exemplos de Inversão de Controle



Programação Orientada a Aspectos



Injeção de Dependência (DI)

INJEÇÃO DE DEPENDÊNCIA



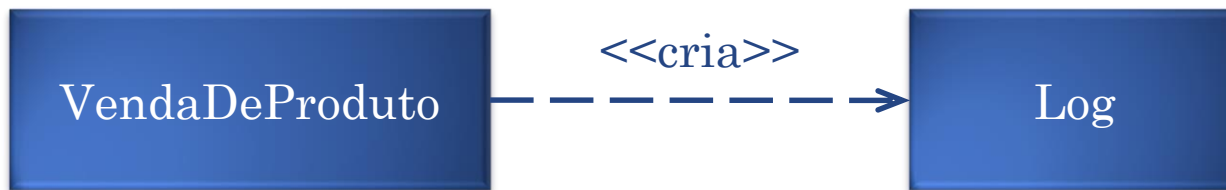
INJEÇÃO DE DEPENDÊNCIA

- **Injeção de Dependência** (*Dependency Injection* ou simplesmente *DI*) é uma técnica utilizada para instanciar classes concretas, sem se manter acoplado a elas.
- Projetos que usam inversão de dependência e inversão de controle são fáceis de se beneficiar de contêineres de injeção de dependência.

PORQUE USAR INJEÇÃO DE DEPENDÊNCIA?

```
public class VendaDeProduto {  
    public void vendeProduto(Produto produto) {  
        //Todo o código para a venda do produto...  
        Log log = new Log("Arquivo.txt");  
        log.grava(produto);  
    }  
}
```

MUDANÇA!!!



Relação de dependência direta entre classes concretas!!!

PORQUE USAR INJEÇÃO DE DEPENDÊNCIA?

- A classe *VendaDeProduto* sabe demais sobre a classe *Log*.
- A classe *VendaDeProduto* sabe **criar** um objeto da classe *Log*.
- Pior, a classe de vendas sabe que a classe *Log* precisa do nome de um arquivo!
- E isso não é justo com a classe *VendaDeProduto*.
- A responsabilidade dela é somente de fechar uma venda.
- Portanto, podemos ver um alto acoplamento de classes neste minúsculo código.

SOLUCIONANDO O PROBLEMA COM DI

- Em vez de deixarmos a responsabilidade da criação da classe *Log* para a classe *VendaDeProduto*, vamos dar a ela esta dependência. Vamos **injetar esta dependência** nela.

```
public class VendaDeProduto {  
    private Log log;  
    public VendaDeProduto(Log logVenda) {  
        this.log = logVenda;  
    }  
    public void vendeProduto(Produto produto) {  
        //Todo o código para a venda do produto...  
        log.grava(produto);  
    }  
}
```

SOLUCIONANDO O PROBLEMA COM DI

- A classe *VendaDeProduto* precisa da classe *Log* para criar um Log mas neste código a classe *VendaDeProduto* recebeu uma instância da classe *Log*!
- Ou seja, agora ela **não se preocupa mais com a criação** da classe *Log*. *VendaDe Produto* simplesmente recebe uma instância criada e a usa.
- Como a classe *Log* é criada agora? Isso **não interessa** à classe *VendaDeProduto*!
- A única coisa que *VendaDeProduto* precisa é de uma instância da classe *Log*, independentemente de como ela foi criada!

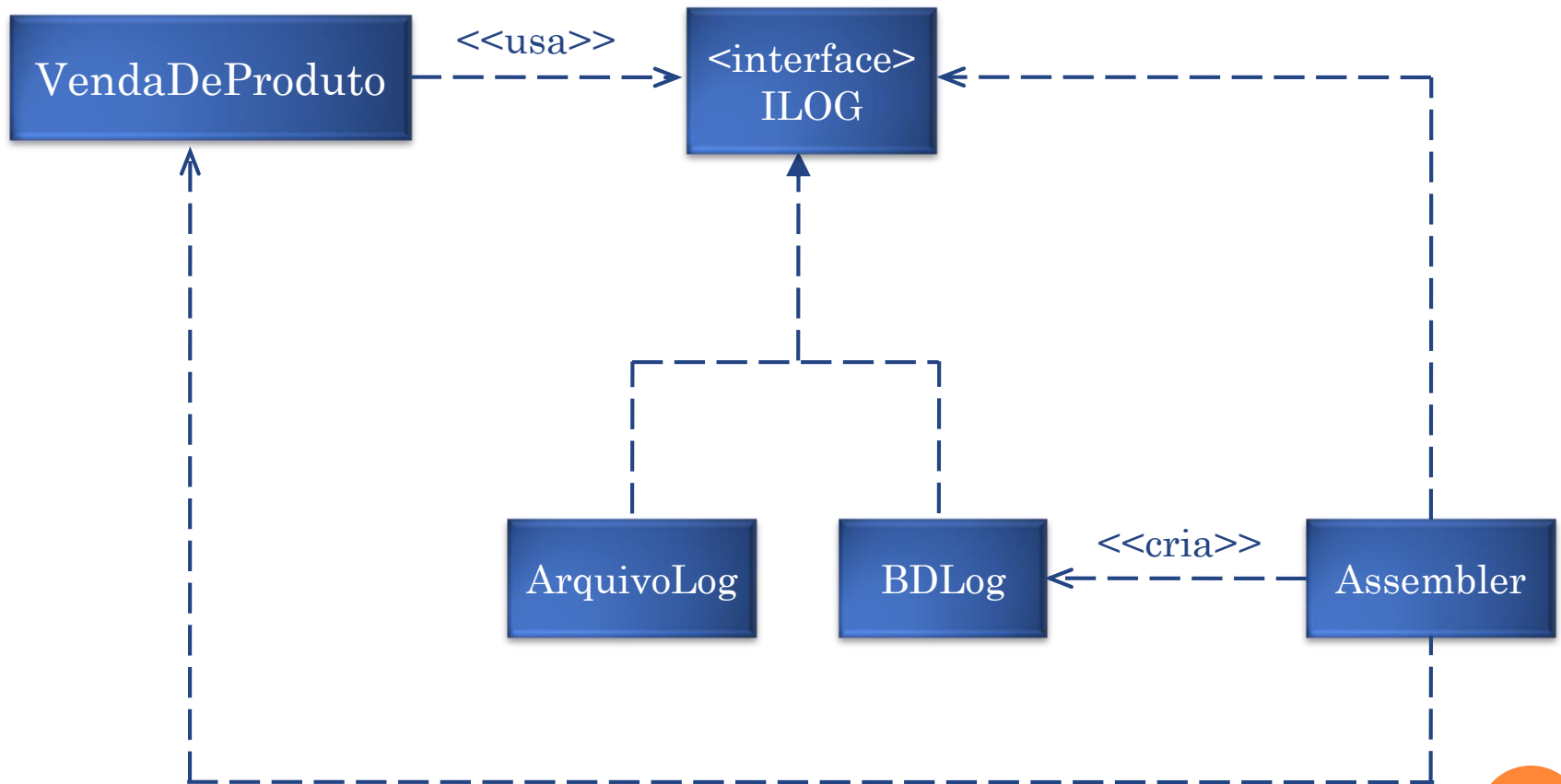
MELHORANDO NOSSA SOLUÇÃO DE DI

- Vamos criar uma interface *ILOG* e usá-la no lugar da classe concreta *Log*.
- Dessa forma, estaremos atendendo ao princípio “Programa para uma interface, não para uma implementação”.
- Isso nos dá mais flexibilidade pois qualquer tipo de ILOG pode ser associado à VendaDeProduto.

```
public interface ILOG { ... }
```

```
public class VendaDeProduto {  
    private ILog log;  
    public VendaDeProduto(ILog logVenda) {  
        this.log = logVenda;  
    }  
}
```

VISUALIZANDO NO DIAGRAMA



FORMAS DE INJEÇÃO DE DEPENDÊNCIA

- Algumas formas de Injeção de Dependência:
 - Injeção por Construtor (*Constructor Injection*)
 - Injeção por Propriedade ou Método Set (*Setter Injection*)
 - Injeção por Interface (*Interface Injection*)

FORMAS DE INJEÇÃO DE DEPENDÊNCIA

- **Construtor (*Constructor Injection*)**: as dependências do objeto são injetadas diretamente em seu construtor.

```
public interface ILOG { ... }

public class VendaDeProduto {
    private ILog log;
    public VendaDeProduto(ILog logVenda) {
        this.log = logVenda;
    }
    public void vendeProduto(Produto produto) {
        //Todo o código para a venda do produto...
        log.grava(produto);
    }
}
```

FORMAS DE INJEÇÃO DE DEPENDÊNCIA

- **Propriedade (*Setter Injection*):** dependências do objeto são injetadas via *setter* em alguma(s) propriedade(s).

```
public interface ILOG { ... }

public class VendaDeProduto {
    private ILog log;
    public void setLog(ILog logVenda) {
        this.log = logVenda;
    }
    public void vendeProduto(Produto produto) {
        //Todo o código para a venda do produto...
        log.grava(produto);
    }
}
```

FORMAS DE INJEÇÃO DE DEPENDÊNCIA

- **Interface (*Interface Injection*):** o objeto a ser injetado é uma abstração da classe concreta (na forma interface ou mesmo classe abstrata).

```
public interface InjectLog { void injectLog ( ILOG ); }
```

```
public class VendaDeProduto implements InjectLog {  
    private ILog log;  
    public void injectLog(ILog logVenda) {  
        this.log = logVenda;  
    }  
    public void vendeProduto(Produto produto) {  
        //Todo o código para a venda do produto...  
        log.grava(produto);  
    }  
}
```

CONSIDERAÇÕES FINAIS

- Injeção de Dependência é um tipo ou uma técnica de Inversão de Controle.
- Injeção de Dependência torna o código mais flexível, menos acoplado e mais fácil de manter.
- Existem formas diferentes de fazer a injeção de dependência.
- Diversos contêineres e *frameworks* fazem uso de este recurso.

REFERÊNCIAS

- http://pt.wikipedia.org/wiki/Inversão_de_controle
- <http://martinfowler.com/articles/injection.html>
- <http://martinfowler.com/bliki/InversionOfControl.html>
- <http://javafree.uol.com.br/artigo/871453/Inversion-Of-Control-Containers-de-Inversao-de-Controle-e-o-padrao-Dependency-Injection.html>
- <http://www.devmedia.com.br/inversao-de-controle-x-injecao-de-dependencia/18763>
- <http://www.k19.com.br/artigos/inversao-de-controle-x-injecao-de-dependencia/>

Programação Orientada a Objetos

Obrigado!!!

Marum Simão Filho
marumsimao@gmail.com

