

Frameworks

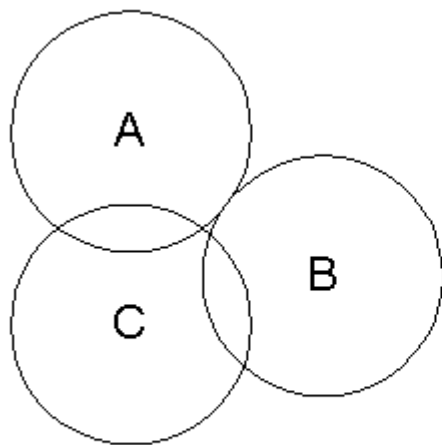
O que é um framework?

Qual é o problema?

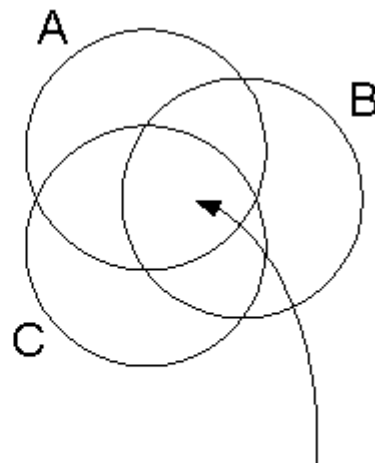
- ✍ Programar é difícil
- ✍ Onde está a maior dificuldade?
 - ✍ "Interface design and functional factoring constitute the key intellectual content of software and is far more difficult to create or re-create than code" (Peter Deutsch)
- ✍ Mas nossos programadores são mortais
 - ✍ "It shouldn't take a good programmer to build a good program"
- ✍ Solução: Temos que fornecer formas de re-uso que vão além de código: re-uso de análise, design, código.
 - ✍ Framework orientado a objeto

O que é um Framework?

- ✍ Um framework captura a funcionalidade comum a várias aplicações
- ✍ As aplicações devem ter algo razoavelmente grande em comum: pertencem a um mesmo domínio de problema



Impossível criar
Framework



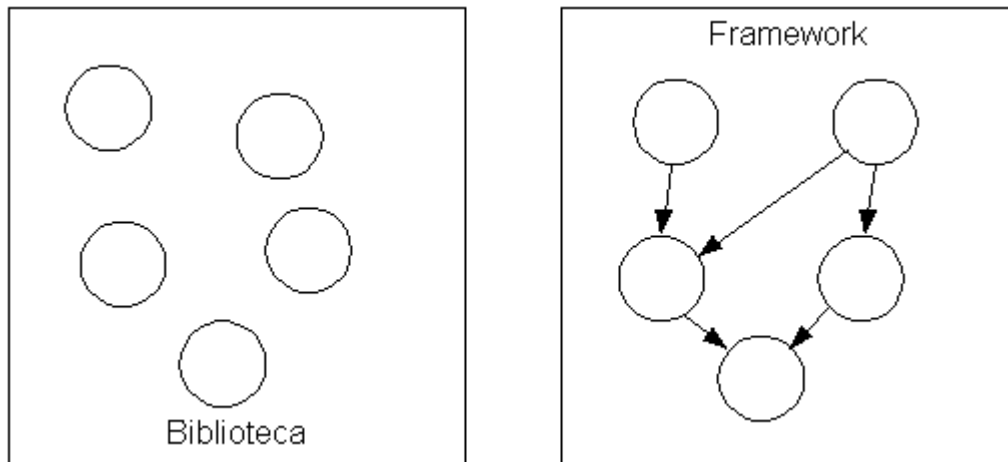
Interseção grande
Possível criar Framework

- ⌘ Há várias definições de frameworks (ver [aqui](#))
- ⌘ A definição que usamos foca quatro características principais de um framework (Orientado a Objeto):
 - ⌘ "Um framework provê uma **solução para uma família de problemas** semelhantes, ...
Usando um **conjunto de classes e interfaces que mostra como decompor** a família de problemas, ...
E como objetos dessas classes **colaboram** para cumprir suas responsabilidades, ...
O conjunto de classes deve ser **flexível e extensível** para permitir a construção de várias aplicações com pouco esforço, especificando apenas as particularidades de cada aplicação"
- ⌘ Observe que um framework é uma aplicação *quase* completa, mas com pedaços faltando
 - ⌘ Ao receber um framework, seu trabalho consiste em prover os pedaços que são específicos para sua aplicação
 - ⌘ As técnicas básicas são **Template Method** e **Composição**

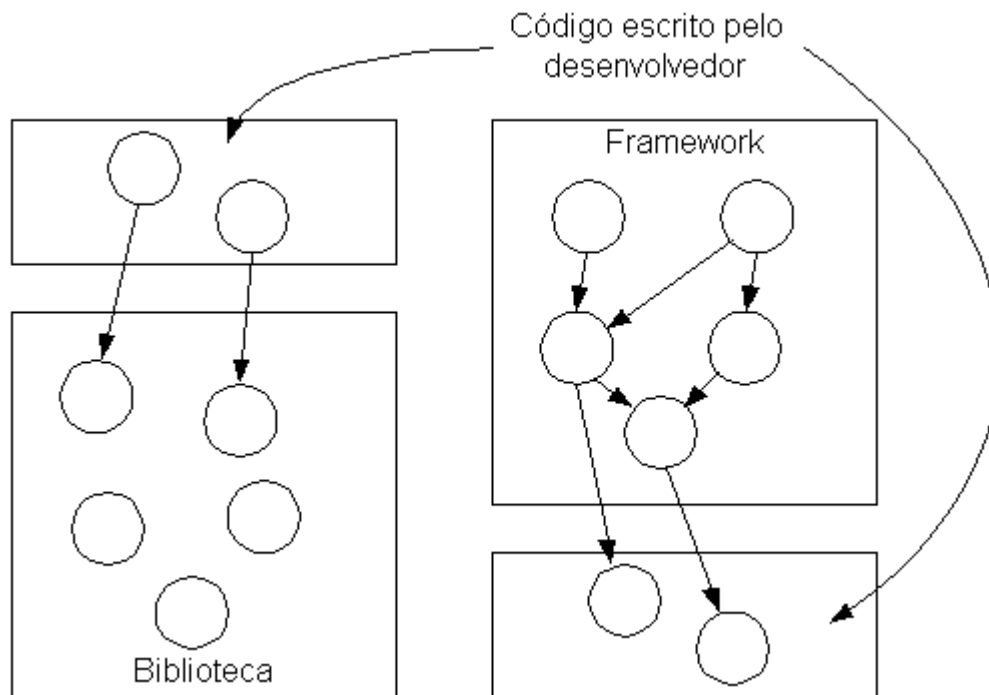
Diferenças entre um Framework e uma Biblioteca de Classes OO

- ⌘ *Numa biblioteca de classes, cada classe é única e independente das outras*

- Num framework, as dependências/colaborações estão embutidas (*wired-in interconnections*)
- Com biblioteca, as aplicações criam as colaborações



- Vê-se portanto que um framework impõe um **modelo de colaboração** (o resultado da análise e design) ao qual você deve se adaptar
 - Já que a comunicação entre objetos já está definida, o projetista de aplicações não precisa saber quando chamar cada método: é o **framework que faz isso**
- Não se pode embutir **conhecimento do domínio** (análise + design) numa biblioteca de classes
- O framework é usado de acordo com o **Hollywood Principle** ("*Don't call us, we'll call you*")
 - É o framework que chama o código da aplicação (que trata das **particularidades** dessa aplicação)
 - Framework = Upside-down library

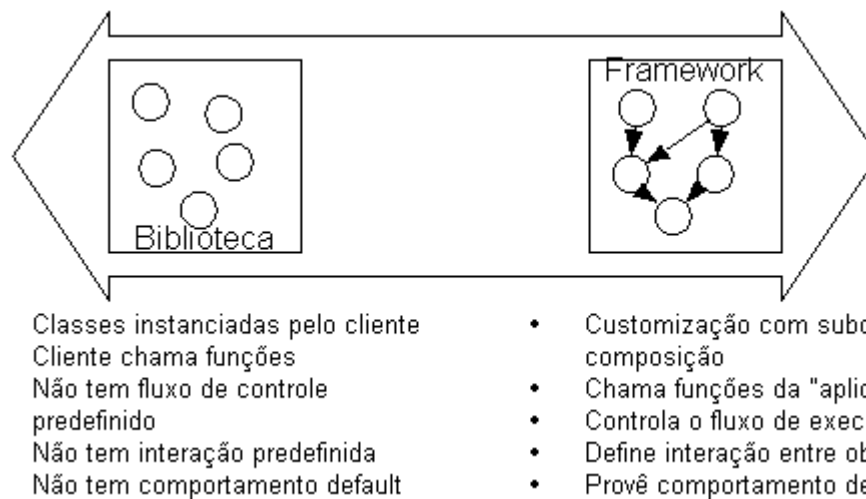


Exemplo do Hollywood Principle

- Modelo de eventos em Java/AWT
- AWT é um framework
- No código abaixo, `mouseClicked()` e `mousePressed()` são chamados pelo framework (AWT)

```
public class MeuMouseListener implements MouseListener {
    public void mouseClicked(MouseEvent event) {
        ...
    }
    public void mousePressed(MouseEvent event) {
        ...
    }
    ...
}
...
MeuMouseListener mouseListener = new MeuMouseListener();
JButton meuBotão = new JButton("clique aqui");
// O seguinte método estabelece a interação entre o obje
// meuBotão e o objeto mouseListener
meuBotão.addMouseListener(mouseListener);
```

- A diferença entre um framework e uma biblioteca de classes não é binária



Diferenças entre Frameworks e Design Patterns

- ⌘ Aparentemente, os dois consistem de classes, interfaces e colaborações prontas
- ⌘ As diferenças são:
 - ⌘ **Design patterns são mais abstratos do que frameworks**
 - ⌘ Um framework inclui código, um design pattern não (só um exemplo do uso de um pattern)
 - ⌘ Devido à presença de código, um framework pode ser estudado a nível de código, executado, e reusado diretamente
 - ⌘ **Design patterns são elementos arquiteturais menores do que frameworks**
 - ⌘ Um framework típico contém vários design patterns mas o contrário nunca ocorre
 - ⌘ Exemplo: Design patterns são frequentemente usados para documentar frameworks
 - ⌘ **Design patterns são menos especializados do que frameworks**
 - ⌘ Frameworks sempre têm um domínio de aplicação particular enquanto design patterns não ditam uma arquitetura de aplicação particular

Características Básicas de Frameworks

- ✍ Um framework deve ser reusável
 - ✍ É o propósito final!
 - ✍ Para ser reusável, deve primeiro ser *usável*
 - ✍ Bem documentado
 - ✍ Fácil de usar
- ✍ Deve ser extensível
 - ✍ O framework contém funcionalidade abstrata (sem implementação) que deve ser completada
- ✍ Deve ser de uso seguro
 - ✍ O desenvolvedor de aplicações não pode destruir o framework
- ✍ Deve ser eficiente
 - ✍ Devido a seu uso em muitas situações, algumas das quais poderão necessitar de eficiência
- ✍ Deve ser completo
 - ✍ Para endereçar o domínio do problema pretendido

frame-1 [programa próxima](#)