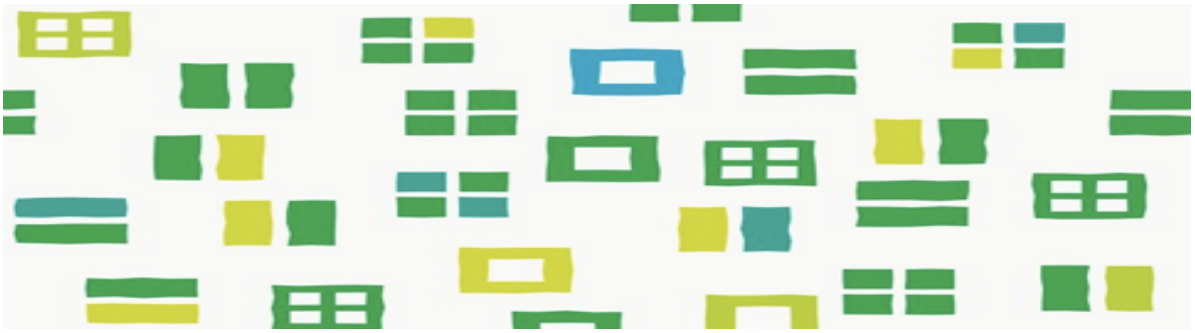
Introdução a Padrões de Projeto

Clovis Fernandes
Divisão de Ciência da Computação
ITA



O objetivo deste curso é fornecer um aprofundamento nas técnicas de programação Java através do uso de padrões de design no contexto do desenvolvimento ágil de software. A ideia é expor os 13 principais padrões de projetos dentre os 23 propostos pela Gangue dos Quatro (Gang of Four – **GoF**), referência aos 4 autores do primeiro e mais abrangente livro da área, bem como 3 outros padrões de projeto muito usados na indústria de software.

	Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John. **Design Patterns**: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994.

Padrão de Projeto é uma solução para um problema comum em um contexto [GoF]. Padrões de Projeto são usados para capturar as características mais evidentes de uma solução a um dado problema que ocorra repetidamente em muitos diferentes contextos.

"Padrões [de Projeto] são maneiras de descrever melhores práticas e bons projetos e de capturar experiência de uma forma que seja possível para outros reusar essa experiência" [http:// www.hillside.net]

Enfatizamos que um padrão não descreve soluções novas, mas soluções padrões, consolidadas, implementadas e testadas com sucesso de forma recorrente em diferentes contextos.

Usualmente um padrão de projeto envolve a interação entre diversos objetos ou classes. Ou seja, um padrão é uma descrição abstrata de um dado problema de projeto e corresponde a um arranjo geral de classes e objetos que o resolve.

Todo desenvolvedor deve se esforçar para escrever código de fácil manutenção, legível, testável e reutilizável. A estruturação de código torna-se mais importante e desafiante à medida que as aplicações se tornam maiores. Nesse sentido, padrões de projeto são cruciais para fazer frente a esse desafio, fornecendo uma estrutura de organização para questões comuns em um contexto particular.

Além disso, os padrões de projeto devem ser aplicados no caso de desenvolvimento ágil principalmente durante a fase de refatoração do TDD, conforme exemplificado nos diversos hands-on apresentados.

Neste curso, os padrões são apresentados de uma forma distinta de outros livros, em especial do GoF. Os capítulos são organizados de acordo com o princípio de design utilizado pelo padrão. Dessa forma, eles servirão para exemplificar tipos de problema que podem ser resolvidos com o dado princípio de design. Essa forma de tratamento educacional é mais detalhada e ampliada no seguinte livro, de autoria de Eduardo Guerra, um dos instrutores deste curso.

	Guerra, Eduardo M. **Design Patterns com Java**: Projeto Orientado a Objetos Guiado por Padrões. Casa do Código, 2012. http://www.casadocodigo.com.br/products/livro-design-patterns

O livro do GoF e a maioria dos livros sobre padrões de projeto separam os padrões em 3 tipos:

- Padrões de criação
- Padrões estruturais
- Padrões comportamentais.

Para manter uma compatibilidade da nossa apresentação com essa forma usual de apresentação, enquadramos, sempre que possível, os padrões mostrados em cada semana do curso no formato do GoF, caracterizando cada padrão num desses três tipos.

Os seguintes conceitos são apresentados neste curso, onde os nomes de padrões de projeto em negrito e sublinhados são os apresentados pelo GoF:

- Revisão de conceitos de orientação a objetos: abstração de dados, encapsulamento, herança e polimorfismo
- Conceitos avançados de orientação a objetos: Hook Methods, Hook Classes, Composição Recursiva, Composição Recursiva com Polimorfismo
- Reúso por meio de herança: padrões Null Object, Template Method e Factory Method
- Delegando comportamento com composição: padrões Strategy, Bridge, State e Observer
- Composição recursiva: padrões Composite e Chain of Responsibility
- Envolvendo objetos: padrões Proxy, Decorator e Adapter
- Estratégias para criar objetos: padrões Static Factory Methods, Factory Method, Singleton e Builder
- Modularidade: padrão Dependency Injection

Padrões de Criação

O objetivo com os padrões de criação é abstrair o processo de criação ou instanciação de objetos em situações em que a forma básica de criação de objetos, por meio do new, pode resultar em problemas de projeto ou adicionar complexidade indesejada ao código. Os padrões de criação resolvem esses problemas controlando a criação de objetos de diferentes maneiras.

De acordo com o GoF, existem cinco padrões de projeto de criação; os três que estão em negrito e sublinhados são os que apresentaremos neste curso:

1. Factory Pattern
2. Singleton Pattern
3. Builder Pattern
4. Abstract Factory
5. Prototype Pattern

Iremos apresentar também o padrão Static Factory Method, que não foi relatado pelo GoF.

Padrões Estruturais

Padrões estruturais fornecem maneiras diferentes de criar uma estrutura de classe, aliviando o desenvolvedor dessa preocupação. Por exemplo, mecanismos de herança, composição e composição recursiva ajudam a criar um objeto grande a partir de pequenos objetos.

De acordo com o GoF, existem sete padrões de projeto estruturais; os cinco que estão em negrito e sublinhados são os que apresentaremos neste curso:

1. **Bridge Pattern**
2. **Composite Pattern**
3. **Proxy Pattern**
4. **Decorator Pattern**
5. **Adapter Pattern**
6. Flyweight Pattern
7. Facade Pattern

Padrões Comportamentais

Padrões comportamentais fornecem solução para a melhor interação entre objetos, com baixo acoplamento e flexibilidade para estender o código sempre que necessário.

De acordo com o GoF, existem onze padrões de projeto comportamentais; os cinco que estão em negrito e sublinhados são os que apresentaremos neste curso:

1. **Strategy Pattern**
2. **Template Method Pattern**
3. **State Pattern**
4. **Observer Pattern**
5. **Chain of Responsibility Pattern**
6. Command Pattern
7. Iterator Pattern
8. Mediator Pattern
9. Visitor Pattern
10. Interpreter Pattern
11. Memento Pattern

Iremos apresentar também o padrão Null Object, que não foi relatado pelo GoF.

Padrão de Modularização

A modularidade é cada vez mais um requisito não funcional importante nas aplicações. Porém, modularidade não é apenas dividir o software em módulos que formam um único bloco, mas permitir que novos módulos possam ser criados e incorporados no software sem a necessidade de sua modificação.

Uma forma de evitar que a própria classe consumidora seja responsável por criar suas dependências ou serviços, é criá-las de forma externa e inseri-las no objeto no momento ou logo depois de sua criação.

O padrão Dependency Injection é um padrão que desacopla a classe consumidora de suas dependências ou serviços, tornando-a uma classe reutilizável em diferentes contextos. A classe ou módulo consumidor é reutilizável com o Dependency Injection porque ele não precisará ser mudado se, num dado contexto, uma dependência (ou serviço) precisa ser trocada por outra diferente.