CAPÍTULO 1

Tornando-se um desenvolvedor pragmático

"Na maioria dos casos, as pessoas, inclusive os facínoras, são muito mais ingênuas e simples do que costumamos achar. Aliás, nós também."
— Fiodór Dostoiévski, em Irmãos Karamazov

Por que fazer esse curso?

1.1 - O que é realmente importante?

Você já passou pelo FJ-11 e, quem sabe, até pelo FJ-21. Agora chegou a hora de codificar bastante para pegar os truques e hábitos que são os grandes diferenciais do programador Java experiente.

Pragmático é aquele que se preocupa com as questões práticas, menos focado em ideologias e tentando colocar a teoria pra andar.

Esse curso tem como objetivo trazer uma visão mais prática do desenvolvimento Java através de uma experiência rica em código, onde exercitaremos diversas APIS e recursos do Java. Vale salientar que as bibliotecas em si não são os pontos mais importantes do aprendizado neste momento, mas sim as boas práticas, a cultura e um olhar mais amplo sobre o design da sua aplicação.

Os *design patterns*, as boas práticas, a refatoração, a preocupação com o baixo acoplamento, os testes de unidade (também conhecidos como testes unitários) e as técnicas de programação (idiomismos) são passados com afinco.

Para atingir tal objetivo, esse curso baseia-se fortemente em artigos, blogs e, em especial, na literatura que se consagrou como fundamental para os

desenvolvedores Java. Aqui citamos alguns desses livros:

http://blog.caelum.com.br/2006/09/22/livros-escolhendo-a-trindade-do-desenvolvedor-java/

Somamos a esses mais dois livros, que serão citados no decorrer do curso, e influenciaram muito na elaboração do conteúdo que queremos transmitir a vocês. Todos os cinco são:

- **Effective Java, Joshua Bloch** Livro de um dos principais autores das maiores bibliotecas do Java SE (como o java.io e o java.util), arquiteto chefe Java na Google atualmente. Aqui ele mostra como enfrentar os principais problemas e limitações da linguagem. Uma excelente leitura, dividido em mais de 70 tópicos de 2 a 4 páginas cada, em média. Entre os casos interessantes está o uso de *factory methods*, os problemas da herança e do `protected`, uso de coleções, objetos imutáveis e serialização, muitos desses abordados e citados aqui no curso.

- **Design Patterns, Erich Gamma et al** Livro de Erich Gamma, por muito tempo líder do projeto Eclipse na IBM, e mais outros três autores, o que justifica terem o apelido de *Gang of Four* (GoF). Uma excelente leitura, mas cuidado: não saia lendo o catálogo dos patterns decorando-os, mas concentre-se especialmente em ler toda a primeira parte, onde eles revelam um dos princípios fundamentais da programação orientada a objetos:"*Evite herança, prefira composição*" e "*Programe voltado às interfaces e não à implementação*".

- **Refactoring, Martin Fowler** Livro do cientista chefe da ThoughtWorks. Um excelente catálogo de como consertar pequenas falhas do seu código de maneira sensata. Exemplos clássicos são o uso de herança apenas por preguiça, uso do `switch` em vez de polimorfismo, entre dezenas de outros. Durante o curso, faremos diversos refactoring clássicos utilizando do Eclipse, muito mais que o básico *rename*.

- **Pragmatic Programmer, Andrew Hunt** As melhores práticas para ser um bom desenvolvedor: desde o uso de versionamento, ao bom uso do logging, debug, nomenclaturas, como consertar bugs, etc.

- **The mythical man-month, Frederick Brooks** Um livro que fala dos problemas que encontramos no dia a dia do desenvolvimento de software, numa abordagem mais gerencial. Aqui há, inclusive, o clássico artigo "No Silver Bullet", que afirma que nunca haverá uma solução única (uma linguagem, um método de desenvolvimento, um sistema operacional) que se adeque sempre a todos os tipos de problema.

1.2 - A importância dos exercícios

É um tanto desnecessário debater sobre a importância de fazer exercícios, porém neste curso específico eles são vitais: como ele é focado em boas práticas, alguma parte da teoria não está no texto - e é passado no decorrer de exercícios.

Não se assuste, há muito código aqui nesse curso, onde vamos construir uma pequena aplicação que lê um XML com dados da bolsa de valores e plota o gráfico de *candlesticks*, utilizando diversas APIs do Java SE e até mesmo bibliotecas externas.

1.3 - Tirando dúvidas e referências

Para tirar dúvidas dos exercícios, ou de Java em geral, recomendamos o fórum do site do GUJ (http://www.guj.com.br/), onde sua dúvida será respondida prontamente.

Fora isso, sinta-se à vontade para entrar em contato com seu instrutor e tirar todas as dúvidas que tiver durante o curso.

Você pode estar interessado no livro TDD no mundo real, da editora Casa do Código:

http://www.tddnomundoreal.com.br/

1.4 - Para onde ir depois?

Se você se interessou pelos testes, design e automação, recomendamos os cursos online de testes da Caelum:

http://www.caelum.com.br/curso/online/testes-automatizados/

O FJ-21 é indicado para ser feito antes ou depois deste curso, dependendo das suas necessidades e do seu conhecimento. Ele é o curso que apresenta o desenvolvimento Web com Java e seus principais ferramentas e frameworks.

Depois destes cursos, que constituem a Formação Java da Caelum, indicamos dois outros cursos, da Formação Avançada:

http://www.caelum.com.br/formacao-java-avancada/

O FJ-25 aborda Hibernate e JPA 2 e o FJ-26 envolve JSF 2, Facelets e CDI. Ambos vão passar por tecnologias hoje bastante utilizadas no desenvolvimento server side para web, e já na versão do Java EE 6.

PRÓXIMO CAPÍTULO:

O modelo da bolsa de valores, datas e objetos imutáveis

Facebook

Newsletter

Casa do Código

Twitter