Data: 12/10/2017

**01 - Well-Designed Apps Rock**

Como escrever realmente um software ótimo?

Nunca é fácil saber por aonde começar:

* A aplicação faz o que deveria fazer?
* Duplicação de códigos: não pode ser bom, pode?
* Aonde trabalhar primeiro?
* Ter certeza de não bagunçar tudo durante o processo.

No final deste capítulo, você irá saber como escrever um excelente software e saberá como aprimorar isso para sempre.

Primeiro software:

Loja do Rick de Instrumentos Musicais

Hoje ela já utiliza um software desenvolvido por outra empresa.

Segue o diagrama de classes (UML) que eles entregaram ao Rick:

|  |
| --- |
| **Guitar** |
| serialNumber: String  price: double  model: String  type: String  backWood: String  topWood: String |
| getSerialNumber(): String  getPrice(): double  setPrice(float)  getBuilder(): String  getModel(): String  getType(): String  getBackWood(): String  getTopWood(): String |

|  |
| --- |
| **Inventory** |
| Guitars: List |
| addGuitar(String, double, String, String, String,  String,String)  getGuitar(String): Guitar  search(Guitar): Guitar |

Observações da Classe Guitar.java:

* Possui 6 variáveis: ( 5 String e uma double)
* Possui 8 métodos (incluindo o método construtor).
* A referência this é utilizada para atribuir os valores passados nos argumentos do método construtor Guitar.
* A Classe possui métodos getters and setters para que sejam consultados e editados as varávies do objeto guitar. (existe apenas um método setter para a variável preço) Isso é devido ao valor financeiro da guitarra poder ser alterado após o cadastro da mesma.

Observações da Classe Inventory.java:

* Possui apenar uma variável (do tipo List).
* Possui 3 métodos (incluindo o método construtor);
* O método getGuitar percorre a lista de guitarras comparando o número de série informado até encontrar a guitarra com o mesmo número de série.
* O método search é utilizado da seguinte forma: o cliente insere pelo menos uma dessas informações: builder, model, type, backWood ou topWood. Serial e preço não são informados, uma vez que eles são únicos.

Ao realizar um teste com o FindGuitarTester.java não é possível encontrar a guitarra desejada.

Como você refaria o programa do Rick?

* Pesquisas sem case sensitive: Não será levado em consideração letras maiúsculas e minúsculas durante a pesquisa, desde que as informações sejam comparadas.
* Alteração de cada variável para constantes Enums.

O que realmente significa escrever um ótimo programa a partir de certos pontos de vista:

**Do usuário:** O programa sempre faz o que o usuário quer que faça. Mesmo que os clientes pensem em novas formas de usar o programa, ele não dá pau ou dá resultados inesperados.

**Dos programadores:** É um código que possui orientação a objetos. Então não há códigos duplicados, e cada objeto realmente controla seu próprio comportamento. Também é fácil aprimorar pois seu design é sólido e flexível.

**Gurus:** É quando você usa princípios e padrões de design. Você mantém seus objetos fracamente acoplados, e o mantém aberto para expandir mas fechado para modificação. Isso também ajuda o código ser mais reutilizado, para que você não tenha que retrabalhar tudo para usar partes da sua aplicação de novo e de novo.

**Do Jonatas:** Um programa excelente é aquele que faz o que o cliente pediu para fazer. Mas a nível de código, ele tem coesão e coerência para que futuros programadores possam aprimorá-lo sem que ele perca a base da funcionalidade. Fazendo que ele seja eterno e não-dependente de hardware/software.

De fato, o que realmente um programa excelente faz é:

*Primeiro: Um programa excelente precisa* ***satisfazer o cliente****. O software precisa fazer o que o cliente quer que ele faça.*

*Segundo: Um programa excelente é bem* ***desenhado/projetado****, bem escrito, e de fácil manutenção, reutilizável e expansível.*