

Git Básico

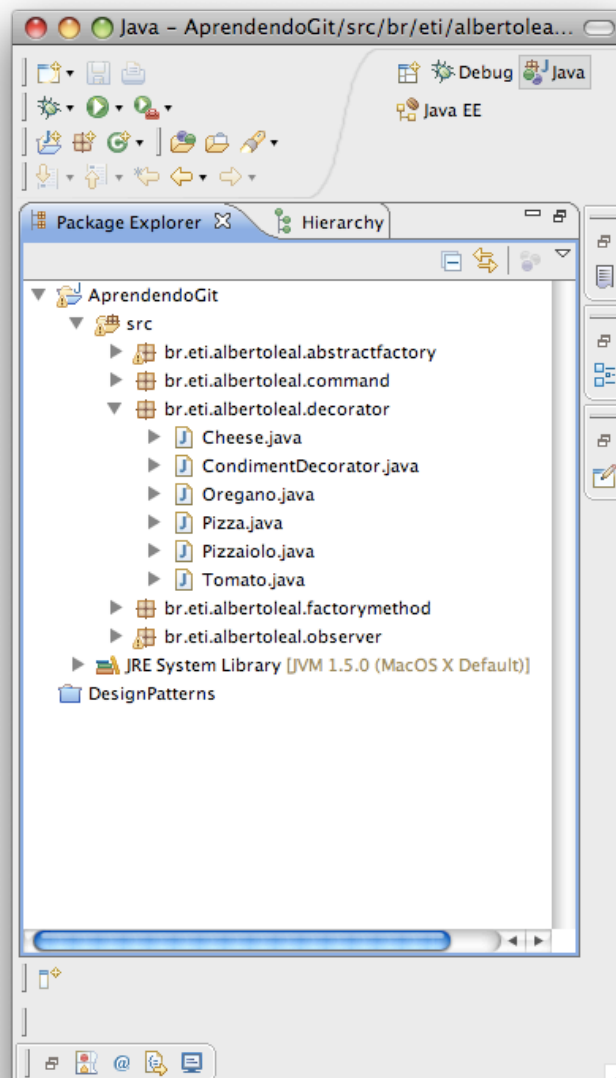
06/2009

O Rebase pode te assustar

Cenário: Você vêm trabalhando há um bom tempo em um branch. Mas, para facilitar o seu trabalho na hora de fazer o merge no repositório central, visando diminuir a quantidade expressiva de conflitos a serem resolvidos de uma única vez ao final do desenvolvimento, você faz diariamente um *rebase* com o repositório principal.

O projeto:

O projeto em que estou trabalhando é um projeto Java. Estou fazendo algumas implementações de padrões de projeto utilizando a linguagem Java. A estrutura do projeto é a seguinte:



O pacote contendo o padrão *decorator* está aberto, pois utilizaremos ele como exemplo.

Executando o comando `git log`, você verá todos os *commits* já realizados:

```
$ git log
commit d2e78a824c98c98a0eec6d083533f36dc35a7bfa
Author: Alberto Leal <albertonb@gmail.com>
Date:   Mon Jun 8 20:48:00 2009 -0300

    Adding Factory Method

commit be3a2a8d22536d427098eaa8bd474f7cb8944bcc
Author: Alberto Leal <albertonb@gmail.com>
Date:   Mon Jun 8 20:47:46 2009 -0300

    Adding Command

commit a0e3a5e1fe3f4c31a3bc3f20283b58ed0d16fc7b
Author: Alberto Leal <albertonb@gmail.com>
Date:   Mon Jun 8 20:47:38 2009 -0300

    Adding Decorator

commit a33c008642c24028477a9c1e7e7d5fca6661c7c9
Author: Alberto Leal <albertonb@gmail.com>
Date:   Mon Jun 8 20:47:26 2009 -0300

    Adding Abstract Factory

commit 52e4505b66ab91f2a8e8511b79ace49aa407e333
Author: Alberto Leal <albertonb@gmail.com>
Date:   Mon Jun 8 20:47:10 2009 -0300

    Adding Observer
```

Repare que o padrão *decorator* foi o terceiro a ser implementado e comitado no git. Como estamos no último *commit*, conseguimos visualizar normalmente todas as alterações/inserções que nele se encontram.

Até o momento, estamos no branch principal, no master. Mas, foi solicitado a adição de um novo atributo na classe *Pizza*, chamado “size” que dirá se pizza é “pequena,

média ou grande”. Para realizar tal tarefa, cria-se um novo branch chamado “Ticket123”:

```
$ git checkout -b Ticket123
```

Agora que já estamos no *branch* relacionado ao ticket, fazemos a adição do novo atributo na classe Pizza. Eis o código da classe Pizza após a alteração:

```
package br.eti.albertoleal.decorator;

public abstract class Pizza {

    private String description = "Unknown Pizza";
    private String size;

    public String getSize() {
        return size;
    }

    public void setSize(String size) {
        this.size = size;
    }

    public void setDescription(String desc) {
        description = desc;
    }

    public String getDescription(){
        return description;
    }

    public abstract double cost();

}
```

Imagine que durante 1(uma) semana você ficou trabalhando em uma série de alterações no código acima, ou até mesmo em outros. E como citado no cenário, ao final do dia você sempre faz um *rebase* com o projeto original para poder pegar as

alterações de outros desenvolvedores e manter o seu *branch* atualizado, consequentemente com uma quantidade menor de conflitos a serem resolvidos quando você for fazer a entrega do seu *branch*.

Para ficar mais evidente o que eu quero mostrar, crie um novo arquivo ou simplesmente faça alguma outra alteração em um código já existente, e depois faça o *commit*. Aqui, eu vou criar um arquivo chamado “TestAprendendoGit.java”

Assim como você, existem outros desenvolvedores criando *branch* a partir do master e fazendo *merge* no mesmo.

Após um longo dia de trabalho, chegou a hora de fazer o *rebase*:

```
$ git rebase master
```

```
First, rewinding head to replay your work on top of it...
```

```
Applying: Adding a new field in the Pizza
```

```
error: patch failed: src/br/eti/albertoleal/decorator/Pizza.java:3
```

```
error: src/br/eti/albertoleal/decorator/Pizza.java: patch does not apply
```

```
Using index info to reconstruct a base tree...
```

```
Falling back to patching base and 3-way merge...
```

```
Auto-merging src/br/eti/albertoleal/decorator/Pizza.java
```

```
CONFLICT (content): Merge conflict in src/br/eti/albertoleal/decorator/Pizza.java
```

```
Failed to merge in the changes.
```

```
Patch failed at 0001 Adding a new field in the Pizza
```

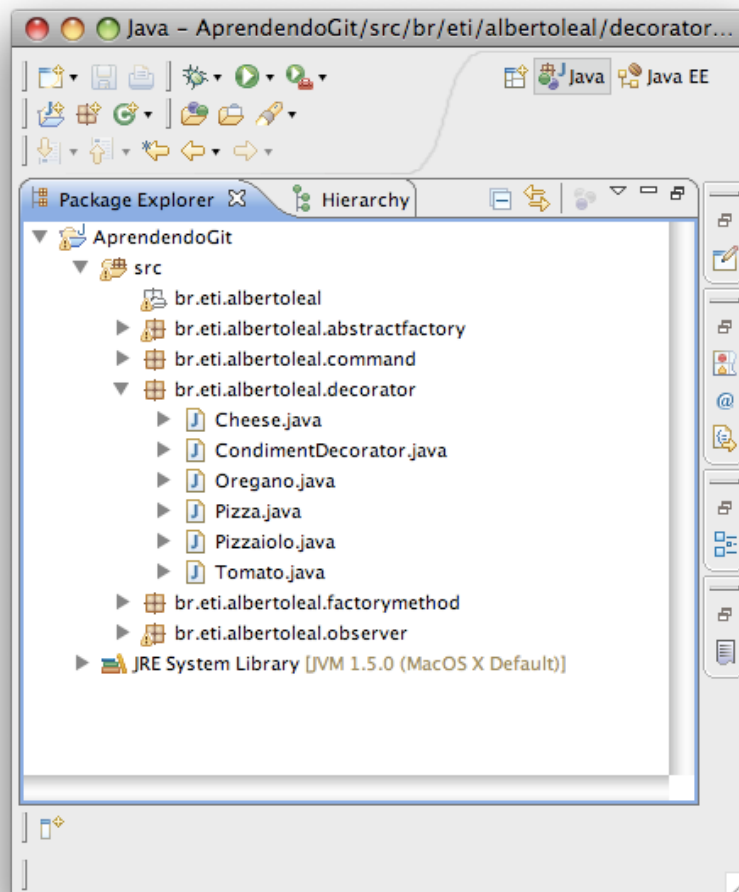
```
When you have resolved this problem run "git rebase --continue".
```

```
If you would prefer to skip this patch, instead run "git rebase --skip".
```

```
To restore the original branch and stop rebasing run "git rebase --abort"
```

Oooops, conflito! Algum desenvolvedor fez alguma alteração na classe *Pizza* também.

Agora que vem o susto! Cadê o tal arquivo “TestAprendendoGit.java” que foi criado?

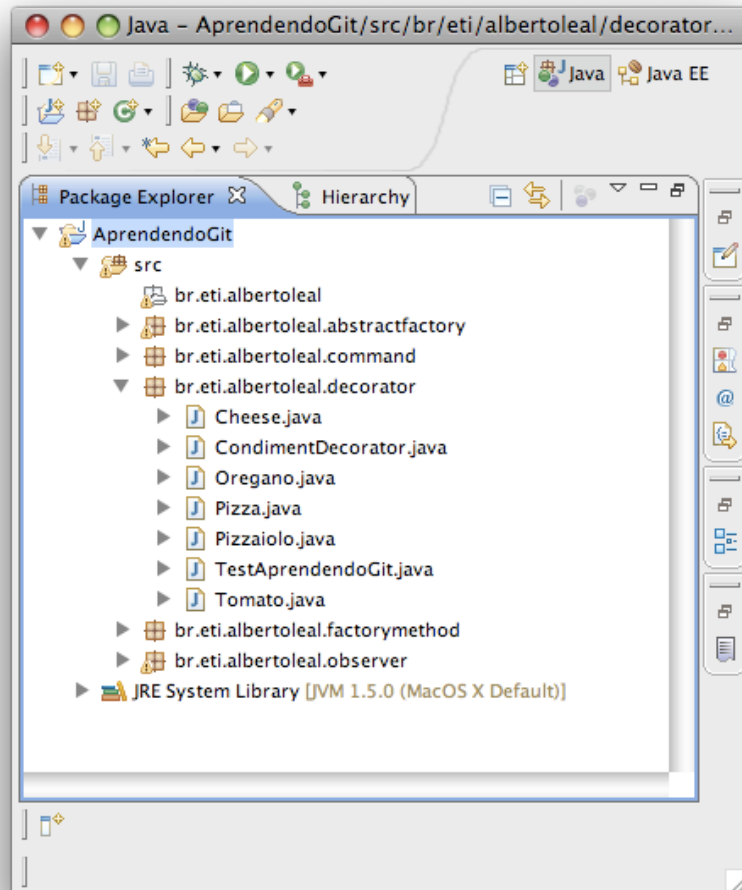


Quando você começa a fazer um *rebase* você não está em nenhum *branch*, você se encontra em uma área “neutra”, vamos por assim dizer. Não acredita em mim?! Agora que você já iniciou o seu *rebase* e obteve os conflitos, execute o comando:

```
git branch
* (no branch)
  Ticket123
  master
```

Viu só? =)

Quando eu falei que o git pode te assustar, o que eu queria dizer é o seguinte: Para aqueles que não estão acostumados com a forma que o Git trabalha, levará um baita susto ao ver que os arquivos e/ou alterações que ele criou, depois do *commit* que deu o conflito, “sumiram”. Não sumiu nada! Eles estão lá apenas aguardando você resolver todos os conflitos e executar todos os comandos “*git rebase --continue*” até chegar no último *commit*, o qual possui o estado final do seu código. Agora que terminou de resolver todos os conflitos e o *rebase*, olha o arquivo aí de novo (TestAprendendoGit.java):



Tentado exemplificar um pouco mais...

Como o *rebase* funciona?

O *rebase* analisa *commit* a *commit*. Ou seja, ele pega todo o seu histórico de *commits* e analisa um a um, do primeiro até o último!

Como um *rebase* pode te assustar?

Vamos supor que você tenha 5 *commits*. Você fez uma mudança específica no último *commit*, ok? Enquanto o *rebase* estava sendo executado, foi constatado um conflito no 2º *commit*. Então, para continuar com o *rebase* você deve fixar o(s) conflito(s).

Com isso, como o *rebase* se encontra no 2º *commit*, você não é capaz de ver as mudanças realizadas nos *commits* posteriores. Se você tentar encontrar aquelas alterações feitas nos *commits* acima, você não conseguirá visualizá-las.