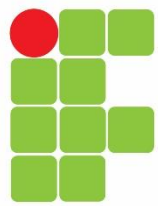


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
TRIÂNGULO MINEIRO
Campus Uberlândia Centro

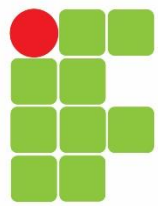
Pós-Graduação Lato Sensu Análise e Desenvolvimento de Sistemas Aplicados a Gestão Empresarial

Coordenação Geral: Prof. Ricardo Boaventura

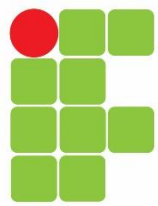
Assessoria Pedagógica: Prof. Clarimundo Machado



- OCP
- LSP
- ISP
- Exercícios de fixação

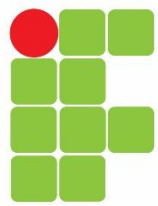


- Pergunta->Resposta->Validação
- Dupla
- Leitura Capítulo 4 – 20 minutos
- Uma dupla pergunta e escolhe quem vai responder
- A dupla responde
- Quem escolheu valida a resposta com base no texto



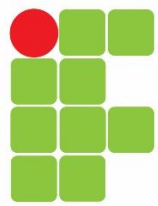
Evolução do código

- Open Close Principle
- Ver versão inicial da `ocp.primeiro`
`CalculadoraDePrecos` (código no Eclipse)
- Agora vamos evoluir o software acrescentando
 - ✓ Outra tabela de preços
 - ✓ Outra forma de entrega



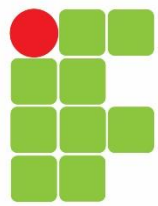
Primeira alternativa

```
public class CalculadoraDePrecos {  
  
    public double calcula(Compra produto) {  
        Frete correios = new Frete();  
        double desconto;  
        if (REGRA 1){  
            TabelaDePrecoPadrao tabela = new TabelaDePrecoPadrao();  
            desconto = tabela.descontoPara(produto.getValor());  
        }  
        if (REGRA 2){  
            TabelaDePrecoDiferenciada tabela = new  
                TabelaDePrecoDiferenciada();  
            desconto = tabela.descontoPara(produto.getValor());  
        }  
        double frete = correios.para(produto.getCidade());  
        return produto.getValor() * (1 - desconto) + frete;  
    }  
}
```

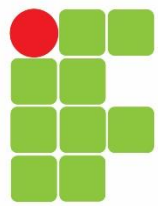


Segunda alternativa

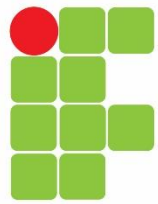
```
public class Frete {  
    public double para(String cidade) {  
        if(REGRA 1) {  
            if("SAO  
                PAULO".equals(cidade.toUpperCase())) {  
                return 15;}  
            return 30;  
        }  
  
        if(REGRA 2) { ... }  
        if(REGRA 3) { ... }  
        if(REGRA 4) { ... }  
    }  
}
```



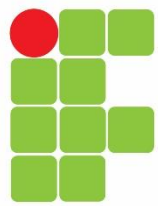
- Criar interfaces para diminuir o acoplamento e dependência
- A classe está aberta para extensão e fechada para alteração – Princípio OCP
- Ver segunda versão da ocp.segundo.
CalculadoraDePrecos (código no Eclipse)



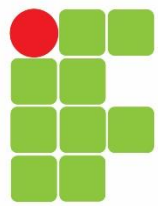
- Ao pensar em classes abertas, o programador precisa pensar em abstrações. Afinal, é por meio delas que ele vai conseguir estender o comportamento.
- Ao pensar em abstrações, idealmente o programador também pensa na estabilidade de cada uma dessas abstrações. Afinal, ele precisa gerenciar o problema do acoplamento.



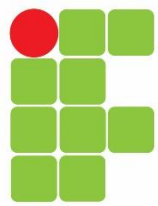
O encapsulamento e a propagação de mudanças



- Pergunta->Resposta->Validação
- Dupla
- Leitura Capítulo 5 – 20 minutos
- Uma dupla pergunta e escolhe quem vai responder
- A dupla responde
- Quem escolheu valida a resposta com base no texto




- Vamos analisar alguns códigos
 - ✓ ProcessadorDeBoletos (Eclipse)
 - ✓ Intimidade inapropriada

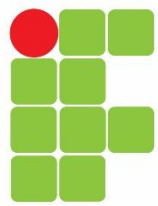


- Qual o problema com esse código?

```
public void algumMetodo() {  
    Fatura fatura = pegaFaturaDeAlgumLugar();  
    fatura.getCliente().marcaComoInadimplente();  
}
```



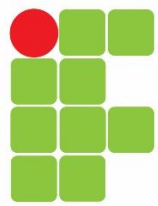
- Este método depende indiretamente de Cliente. E se a classe Cliente excluir o método marcaComoInadimplente()?
- Esse é o problema de invocações em cadeia. Se temos a.getB().getC().getD(), se B mudar, ou se C mudar, ou se D mudar, esse código quebrará.



Lei de Demeter

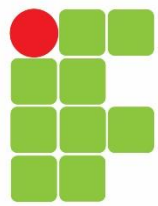
```
public void algumMetodo() {  
    Fatura fatura = pegaFaturaDeAlgumLugar();  
    fatura.getCliente().marcaComoInadimplente();  
}
```

```
public void algumMetodo() {  
    Fatura fatura = pegaFaturaDeAlgumLugar();  
    fatura.marcaClienteComoInadimplente();  
}
```



Getters / Setters

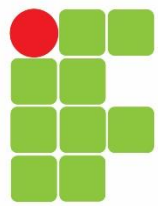
- Também é fonte de problemas de encapsulamento
- Pense bem antes de criar os métodos Getters / Setters



- O problema do encapsulamento está aqui

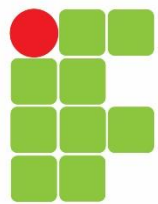
```
if(total >= fatura.getValor()) {  
    fatura.setPago(true);  
}
```

- Essa regra de negócios precisar estar encapsulada na classe Fatura
- O método setPago também precisa ser excluído de da classe Fatura



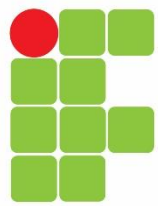
- Vamos adicionar um método à classe Fatura para realizar a regra de negócio que estava na classe ProcessadorDeBoletos

```
public void adicionaPagamento(Pagamento pagamento) {  
    this.pagamentos.add(pagamento);  
    if (valorTotalDosPagamentos() > this.valor) {  
        this.pago = true;  
    }  
}
```

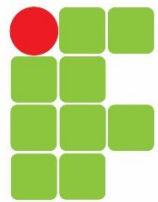
- Agora a classe `ProcessadorDeBoletos` fica assim:

```
5 public class ProcessadorDeBoletos {  
6     public void processa(List<Boleto> boletos, Fatura fatura) {  
7         for(Boleto boleto : boletos) {  
8             Pagamento pagamento = new Pagamento(boleto.getValor(),  
9                 MeioDePagamento.BOLETO);  
10            fatura.adicionaPagamento(pagamento);  
11        }  
12    }  
13 }
```



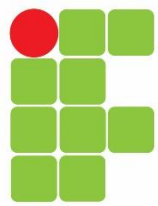
Encapsulamento

- Mostrar a interface e esconder a implementação, ou seja, saber o que faz sem saber como faz
- Se o código não está bem encapsulado, isso implica em termos a regra de negocio espalhada por lugares diferentes.
- Para encontrar problemas de encapsulamento, faça as Perguntas
 - ✓ **O que** o método faz?
 - ✓ **Como** o método faz?
- Lembre-se que precisamos sempre diminuir a quantidade de pontos de mudança

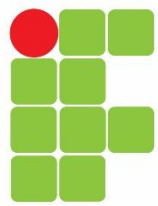


LSP

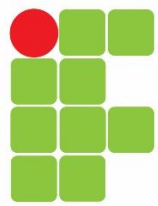
Liskov Substitutive Principle



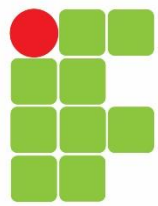
- Pergunta->Resposta->Validação
- Dupla
- Leitura Capítulo 6 – 20 minutos
- Uma dupla pergunta e escolhe quem vai responder
- A dupla responde
- Quem escolheu valida a resposta com base no texto



- O problema da herança
- Código exemplo
- Veja na classe ProcessadorDeInvestimentos o laço com contas. Depois que foi feita a herança de ContaComum este código pode quebrar
- As classes filhas precisam respeitar os contratos definidos pela classe pai

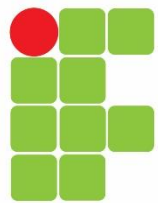


- Observar pré-condições e pós-condições da classe pai
- Pré-condições
 - ✓ dados de entrada – parâmetros do método
- Pós-condições
 - ✓ o que devolve
- “Regra” de Liskov
 - ✓ a classe filha só pode “afrouxar” a pré-condição
 - ✓ pós-condição só pode ser “apertada”



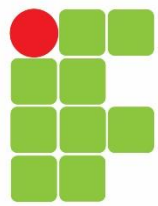
Favoreça a composição

- Analisar código `liskov.fatorado`

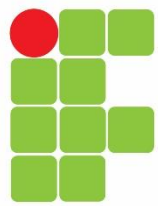


ISP

Interface Segregation Principle

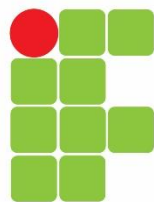


- Pergunta->Resposta->Validação
- Dupla
- Leitura Capítulo 7 – 20 minutos
- Uma dupla pergunta e escolhe quem vai responder
- A dupla responde
- Quem escolheu valida a resposta com base no texto



- Suponha a seguinte interface, responsável por calcular um imposto e gerar uma Nota Fiscal:

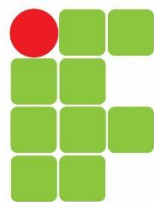
```
interface Imposto {  
    NotaFiscal geraNota();  
    double imposto(double valorCheio);  
}
```



Interfaces coesas

```
class ISS implements Imposto {  
    public double imposto(double valorCheio) {  
        return 0.1 * valorCheio;  
    }  
  
    public NotaFiscal geraNota() {  
        return new NotaFiscal(  
            "Alguma informacao aqui",  
            "Alguma outra informacao aqui"  
        );  
    }  
}
```

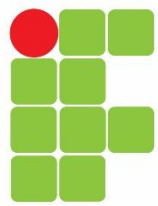
Implementa os dois
métodos da interface



Interfaces coesas

```
class IXMX implements Imposto {  
    public double imposto(double valorCheio) {  
        return 0.2 * valorCheio;  
    }  
  
    public NotaFiscal geraNota() {  
        // lança uma exceção  
        throw new NaoGeraNotaException();  
        // ou retornar nulo  
        return null;  
    }  
}
```

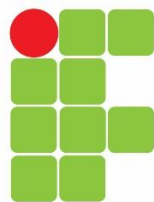
Este imposto
não gera Nota
Fiscal, o que
fazer?



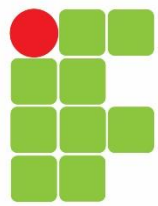
- Dividir a interface em duas mais coesas

```
interface CalculadorDeImposto {  
    double imposto(double valorCheio);  
}
```

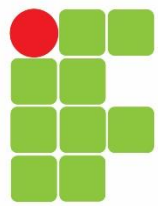
```
interface GeradorDeNota {  
    NotaFiscal geraNota();  
}
```



```
class ISS implements CalculadorDeImposto, GeradorDeNota {  
    // os dois métodos aqui  
}  
  
class IXMX implements CalculadorDeImposto {  
    // só implementa uma interface, pois  
    // esse aqui não gera nota fiscal  
}
```

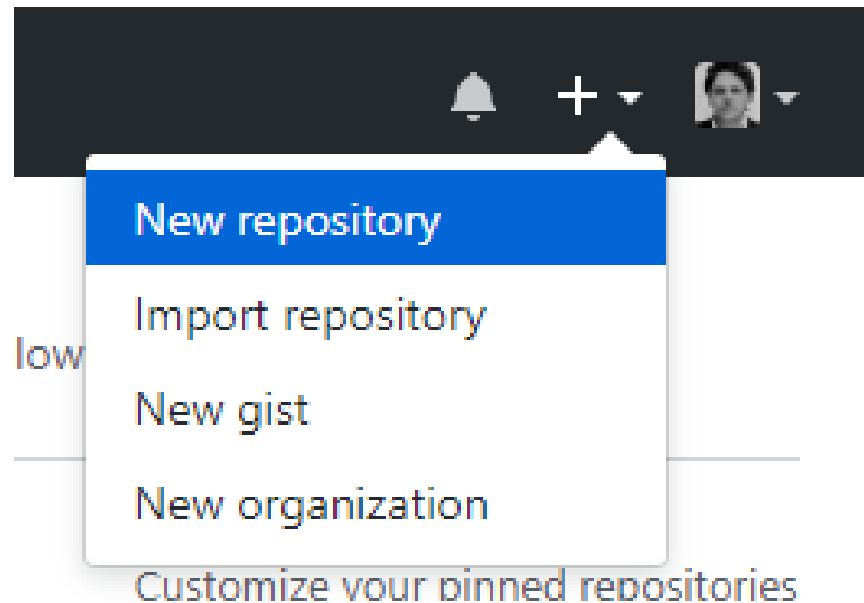


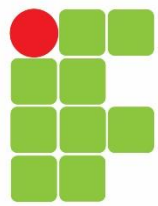
- Baixar o código de <https://github.com/angoti/ListaExerciciosAula5-POO>
- Postar a solução no GitHub e postar o endereço no classroom



Como postar no GitHub

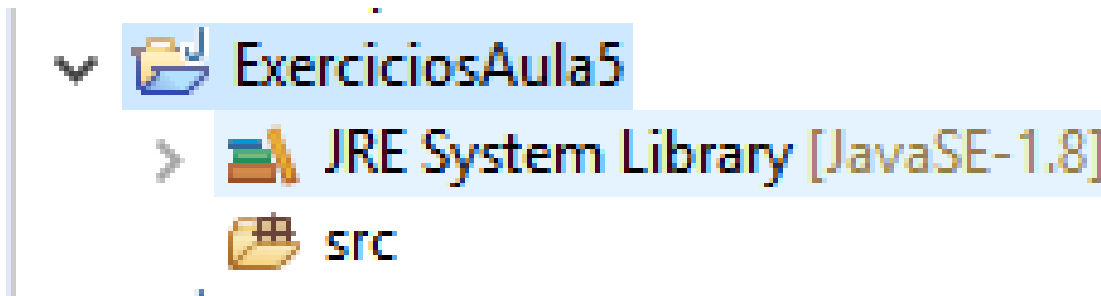
- Abrir uma conta no GitHub
- Criar um repositório, por exemplo, ExerciciosAula5



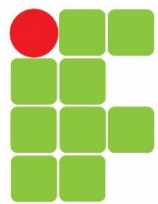


Como postar no GitHub

- Criar um projeto no Eclipse



- Navegue até a pasta onde está seu projeto Eclipse e clique com o botão direito em cima da pasta – veja próxima página



Como postar no GitHub

workspace-oxygen

.metadata

.recommenders

Aula2

Aula3

Aula4

Aula5

Aula2

Aula3

Aula4

Aula5

Exceções

ExemploPolimor

ExerciciosAula5

...

Abrir no Visual Studi



Git GUI Here



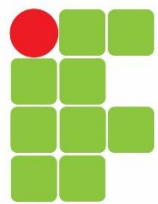
Git Bash Here

Reproduzir com o re

Compartilhar com

Restaurar versões an

Incluir na biblioteca



Como postar no GitHub

- Digite os seguintes comandos

1. git init

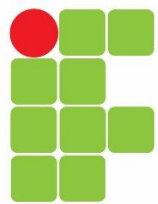
```
$ git init
Initialized empty Git repository in C:/workspace-oxygen/ExerciciosAula5/.git/

angot@SamsungAngoti MINGW64 /c/workspace-oxygen/ExerciciosAula5 (master)
$ |
```

2. git add .

```
angot@SamsungAngoti MINGW64 /c/workspace-oxygen/ExerciciosAula5
$ git add .

angot@SamsungAngoti MINGW64 /c/workspace-oxygen/ExerciciosAula5
$ |
```



Como postar no GitHub

3. git commit -m "Primeiro commit"

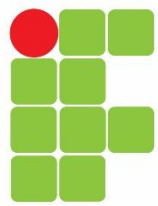
```
$ git commit -m "Primeiro commit"
[master (root-commit) 18c2fd7] Primeiro commit
3 files changed, 34 insertions(+)
create mode 100644 .classpath
create mode 100644 .project
create mode 100644 .settings/org.eclipse.jdt.core.prefs

angot@SamsungAngoti MINGW64 /c/workspace-oxygen/ExerciciosAula5
$ |
```

4. git remote add origin https://github.com/angoti/ExerciciosAula5.git

```
$ git remote add origin https://github.com/angoti/ExerciciosAula5.git

angot@SamsungAngoti MINGW64 /c/workspace-oxygen/ExerciciosAula5 (master)
$ |
```



Como postar no GitHub

5. git push -u origin master

```
$ git push -u origin master
Counting objects: 6, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 938 bytes | 0 bytes/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To https://github.com/angoti/ExerciciosAula5.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.

angot@SamsungAngoti MINGW64 /c/workspace-oxygen/ExerciciosAula5 (master)
$ |
```