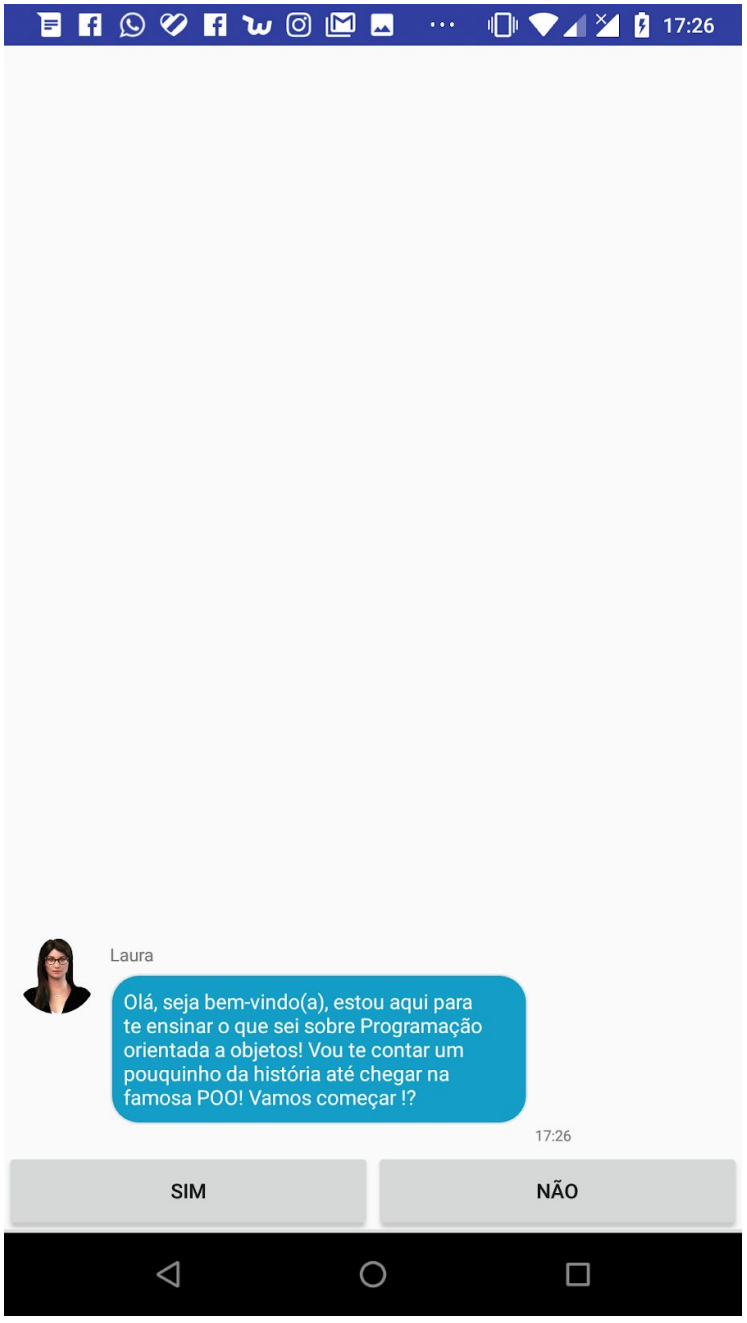
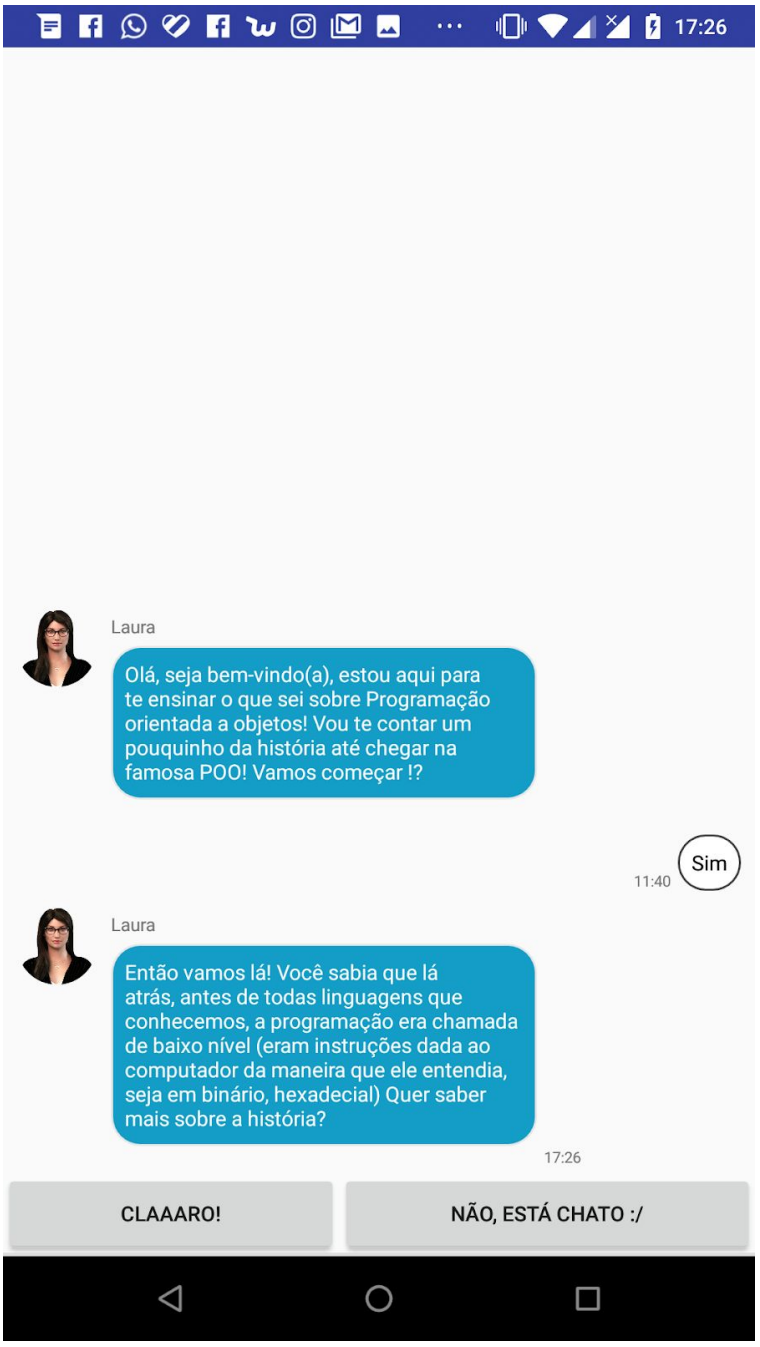
PoOrientação

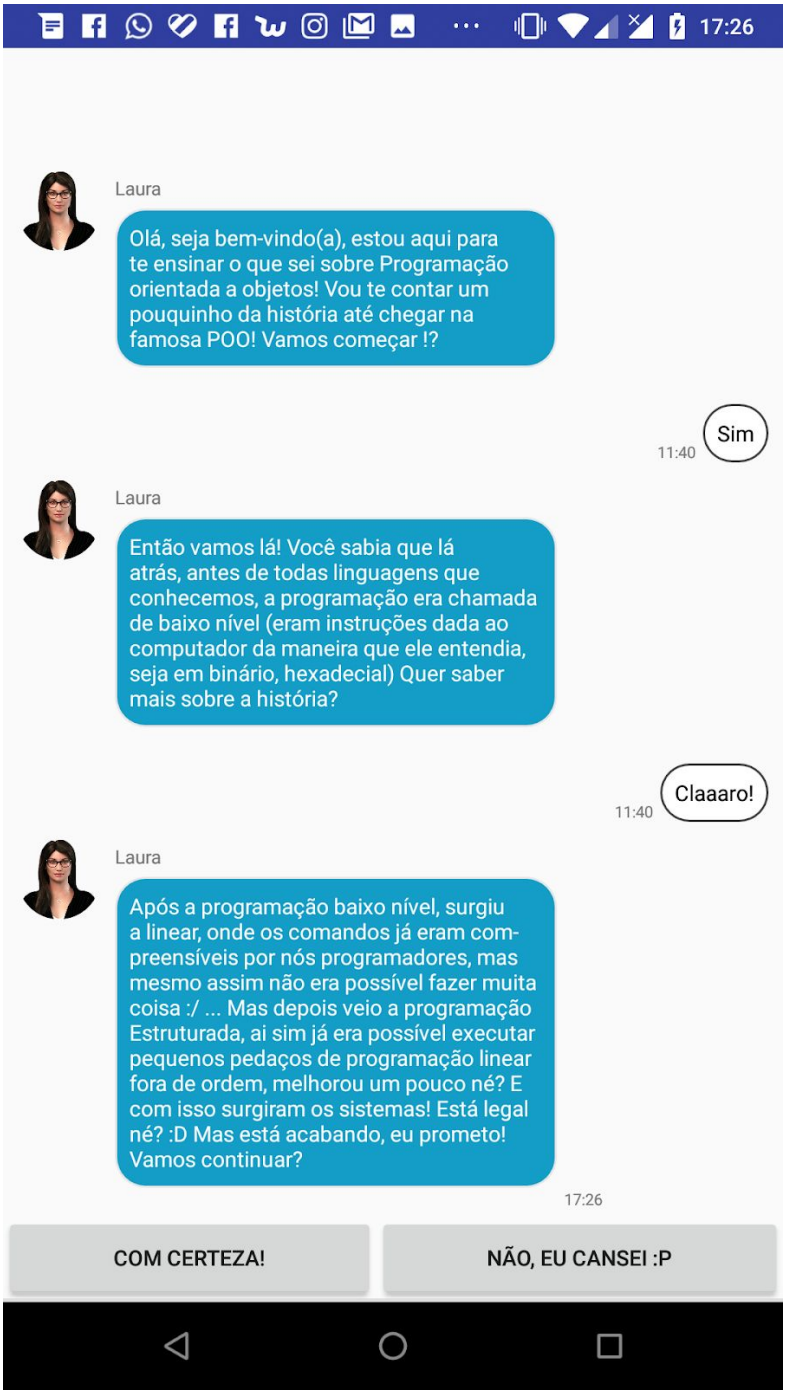
O aplicativo tem como intuito passar um pouco de noção de programação orientada a objetos para quem está iniciando. Atualmente possui 3 funções que são elas:

- História da programação
- Teoria de POO
- Padrões de Projeto

História da programação: é um chatbot que permite o usuário interagir através de botões, e caso as respostas sejam positivas ele vai obtendo informação sobre a história da programação. Segue os prints do fluxo do chat.




17:26
Laura
Olá, seja bem-vindo(a), estou aqui para te ensinar o que sei sobre Programação orientada a objetos! Vou te contar um pouquinho da história até chegar na famosa POO! Vamos começar !?
Sim
11:40
Laura
Então vamos lá! Você sabia que lá atrás, antes de todas linguagens que conhecemos, a programação era chamada de baixo nível (eram instruções dada ao computador da maneira que ele entendia, seja em binário, hexadecial) Quer saber mais sobre a história?
17:26
CLAAARO!
NÃO, ESTÁ CHATO :/


17:26
Laura
Olá, seja bem-vindo(a), estou aqui para te ensinar o que sei sobre Programação orientada a objetos! Vou te contar um pouquinho da história até chegar na famosa POO! Vamos começar !?
Sim
11:40
Laura
Então vamos lá! Você sabia que lá atrás, antes de todas linguagens que conhecemos, a programação era chamada de baixo nível (eram instruções dada ao computador da maneira que ele entendia, seja em binário, hexadecial) Quer saber mais sobre a história?
Claaaro!
11:40
Laura
Após a programação baixo nível, surgiu a linear, onde os comandos já eram compreensíveis por nós programadores, mas mesmo assim não era possível fazer muita coisa :/ ... Mas depois veio a programação Estruturada, ai sim já era possível executar pequenos pedaços de programação linear fora de ordem, melhorou um pouco né? E com isso surgiram os sistemas! Está legal né? :D Mas está acabando, eu prometo! Vamos continuar?
17:26
COM CERTEZA!
NÃO, EU CANSEI :P

Laura

Então vamos lá! Você sabia que lá atrás, antes de todas linguagens que conhecemos, a programação era chamada de baixo nível (eram instruções dada ao computador da maneira que ele entendia, seja em binário, hexadecial) Quer saber mais sobre a história?

11:40 Claaaro!

Laura

Após a programação baixo nível, surgiu a linear, onde os comandos já eram compreensíveis por nós programadores, mas mesmo assim não era possível fazer muita coisa :/ ... Mas depois veio a programação Estruturada, ai sim já era possível executar pequenos pedaços de programação linear fora de ordem, melhorou um pouco né? E com isso surgiram os sistemas! Está legal né? :D Mas está acabando, eu prometo! Vamos continuar?

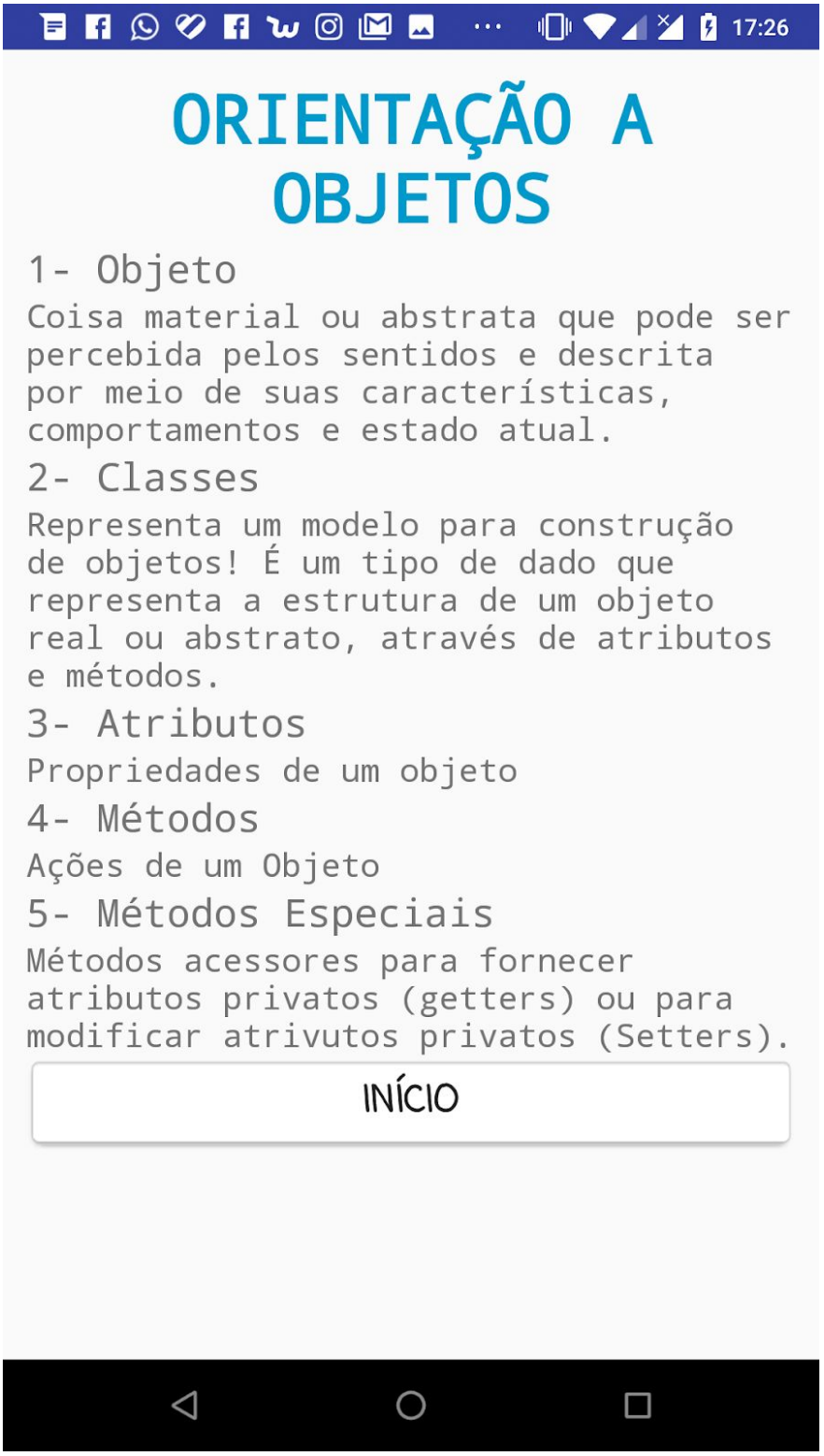
11:40 Com certeza!

Laura

Após a programação estruturada, surgiu a Programação Modular, onde já era possível criar pequenos módulos estruturados (Funções), valorizando cada vez mais dados e funcionalidades! Com intuito de colocar os módulos em sistemas cada vez maiores. E agora sim chegamos na Programação Orientada a Objetos :D Vamos descobrir um pouquinho mais sobre ela???

17:26

ATÉ QUE ENFIM! DEMOROU O/



Após essa básica noção de java é possível estudar sobre os padrões de projetos: Strategy, Singleton e Decorator. Para cada padrão tem um exemplo que foi desenvolvido no código, mas também tem uma tela explicando os mesmos.
segue em anexo o fluxo:

strategy

Strategy

O padrão Strategy, por meio de polimorfismo, permite adicionar/alterar implementações de um método comum, deixando-as encapsuladas, sem que impacte as chamadas realizadas pelo cliente. Ele permite que os algoritmosvariem independentemente entre os clientes que os utilizam

EXEMPLO STRATEGY

MENU PRINCIPAL

Padrão Strategy

1- Exemplo

Vamos fazer o cálculo de impostos, Temos 4 classes (CalculadorDeimposto, ICMS, ISS e Orcamento) e 1 interface (Imposto). Você já deve estar imaginando que os impostos ICMS e ISS implementam a interface Imposto, e é isso mesmo!

2- CalculadorDeImposto

Possui apenas um método que é o realizaCalculo(Orcamento orcamento, Imposto impostoQualquer), esse método calcula o imposto para o orçamento utilizando o método da interface Imposto calculaImposto()

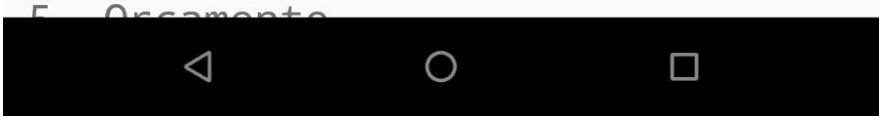
3- ICMS

A classe ICMS implementa a interface Imposto e com isso sobreescreve seu método calculaImposto(Orcamento orcamento) retornando o valor do orçamento + o imposto aplicado a 10% do valor do orçamento

4- ISS

A classe ICMS implementa a interface Imposto e com isso sobreescreve seu método calculaImposto(Orcamento orcamento) retornando o valor do orçamento + o imposto aplicado a 6% do valor do orçamento

5- Orcamento



valor do orçamento

4- ISS

A classe ICMS implementa a interface Imposto e com isso sobreescreve seu método calculaImposto(Orcamento orcamento) retornando o valor do orçamento + o imposto aplicado a 6% do valor do orçamento

5- Orcamento

A classe orçamento posui o atributo double valor, e os métodos get e set do atributo

6- Imposto

A interface imposto apenas possui o método que calculaImposto(Orcamento orcamento), que as classes ICMS e ISS tem em comum.

7- Strategy

E onde usamos o Strategy? Exatamente na hora de criar os impostos! Na hora de criar um ICMS por exemplo, devemos criar assim: Imposto icms = new ICMS() conseguiram ver ?? Estamos passando a estratégia que a interface deve usar, com isso ainda temos a classe que vai calcular o imposto para nós! Se você reparar ela é genérica, pois independente do imposto que mandamos, se nós instanciamos passando a estratégia, a classe genérica vai calcular e devolver o resultado!

VOLTAR PADRÕES DE PROJETOS

Padrão Decorator

1- Exemplo

Vamos fazer o cálculo de impostos, Temos 4 classes (CalculadorDeimposto, ICMS, ISS e Orcamento) e 1 interface (Imposto). Você já deve estar imaginando que os impostos ICMS e ISS implementam a interface Imposto, e é isso mesmo!

2- CalculadorDeImposto

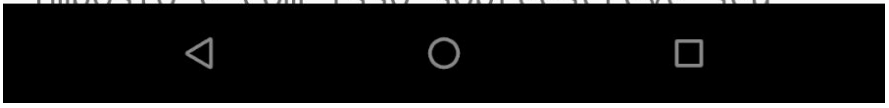
Possui apenas um método que é o realizaCalculo(Orcamento orcamento, Imposto impostoQualquer), esse método calcula o imposto para o orçamento utilizando o método da interface Imposto calculaImposto()

3- ICMS

A classe ICMS implementa a interface Imposto e com isso sobreescreve seu método calculaImposto(Orcamento orcamento) retornando o valor do orçamento + o imposto aplicado a 10% do valor do orçamento. E agora também possui dois construtores, um sem receber nada (public ICMS()) e o outro recebendo o outroImposto (public ICMS(Imposto outroImposto)) para os casos de impostos complexos.

4- ISS

A classe ICMS implementa a interface Imposto e com isso sobreescreve seu



casos de impostos complexos.

4- ISS

A classe ICMS implementa a interface Imposto e com isso sobreescreve seu método calculaImposto(Orcamento orcamento) retornando o valor do orçamento + o imposto aplicado a 6% do valor do orçamento. E agora também possui dois construtores, um sem receber nada (public ICMS()) e o outro recebendo o outroImposto (public ICMS(Imposto outroImposto)) para os casos de impostos complexos.

5- Orcamento

A classe orçamento posui o atributo double valor, e os métodos get e set do atributo

6- Imposto

A interface imposto também sofre algumas mudanças comparado aos outros padrões de projeto. Agora temos um atributo Imposto outroImposto, que vai indicar se é um imposto composto, além disso, temos 2 construtores, o normal public Imposto(), e o construtor para quando houver mais de um imposto instanciado public Imposto (Imposto outroImposto), que recebe por parâmetro o outro imposto. Ainda temos mais algumas mudanças, que é o método calculaOutroImposto(Orcamento orcamento), precisamos apenas chamar o método calculaImposto() do outro imposto passando o orcamento (outroI

método calculaImposto(Orcamento orcamento) retornando o valor do orçamento + o imposto aplicado a 6% do valor do orçamento. E agora também possui dois construtores, um sem receber nada (public ICMS()) e o outro recebendo o outroImposto (public ICMS(Imposto outroImposto)) para os casos de impostos complexos.

5- Orcamento

A classe orçamento posui o atributo double valor, e os métodos get e set do atributo

6- Imposto

A interface imposto também sofre algumas mudanças comparado aos outros padrões de projeto. Agora temos um atributo Imposto outroImposto, que vai indicar se é um imposto composto, além disso, temos 2 construtores, o normal public Imposto(), e o construtor para quando houver mais de um imposto instanciado public Imposto (Imposto outroImposto), que recebe por parâmetro o outro imposto. Ainda temos mais algumas mudanças, que é o método calculaOutroImposto(Orcamento orcamento), precisamos apenas chamar o método calculaImposto() do outro imposto passando o orcamento (outroI mposto.calculaImposto(orcamento)) E pronto! Está implementado o decorator.

VOLTAR PADRÕES DE PROJETOS

orator

singleton

Singleton

O padrão Singleton garante que uma classe tenha apenas uma instância de si mesma, fornecendo umponto global de acesso a ela. Ou seja, a classe gerencia sua própria instância, evitando que qualqueroutra classe crie um instância dela. Normalmente o ponto global de acesso a instância da classe chama-se: getInstance e é através dele que as demais classes receberão a instância da classe com o padrão Singleton, caso não exista a instância a classe cria, se existir a classe retorna a instância existente.

EXEMPLO SINGLETON

Padrão Singleton

1- Exemplo

Vamos fazer o cálculo de impostos, Temos 4 classes (CalculadorDeimposto, ICMS, ISS e Orcamento) e 1 interface (Imposto). Você já deve estar imaginando que os impostos ICMS e ISS implementam a interface Imposto, e é isso mesmo!

2- CalculadorDeImposto

Possui apenas um método que é o realizaCalculo(Orcamento orcamento, Imposto impostoQualquer), esse método calcula o imposto para o orçamento utilizando o método da interface Imposto calculaImposto()

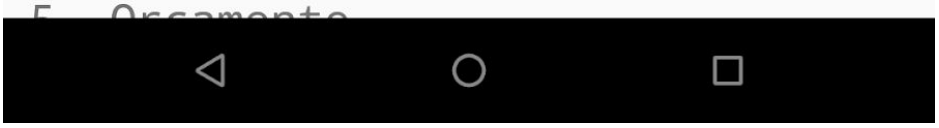
3- ICMS

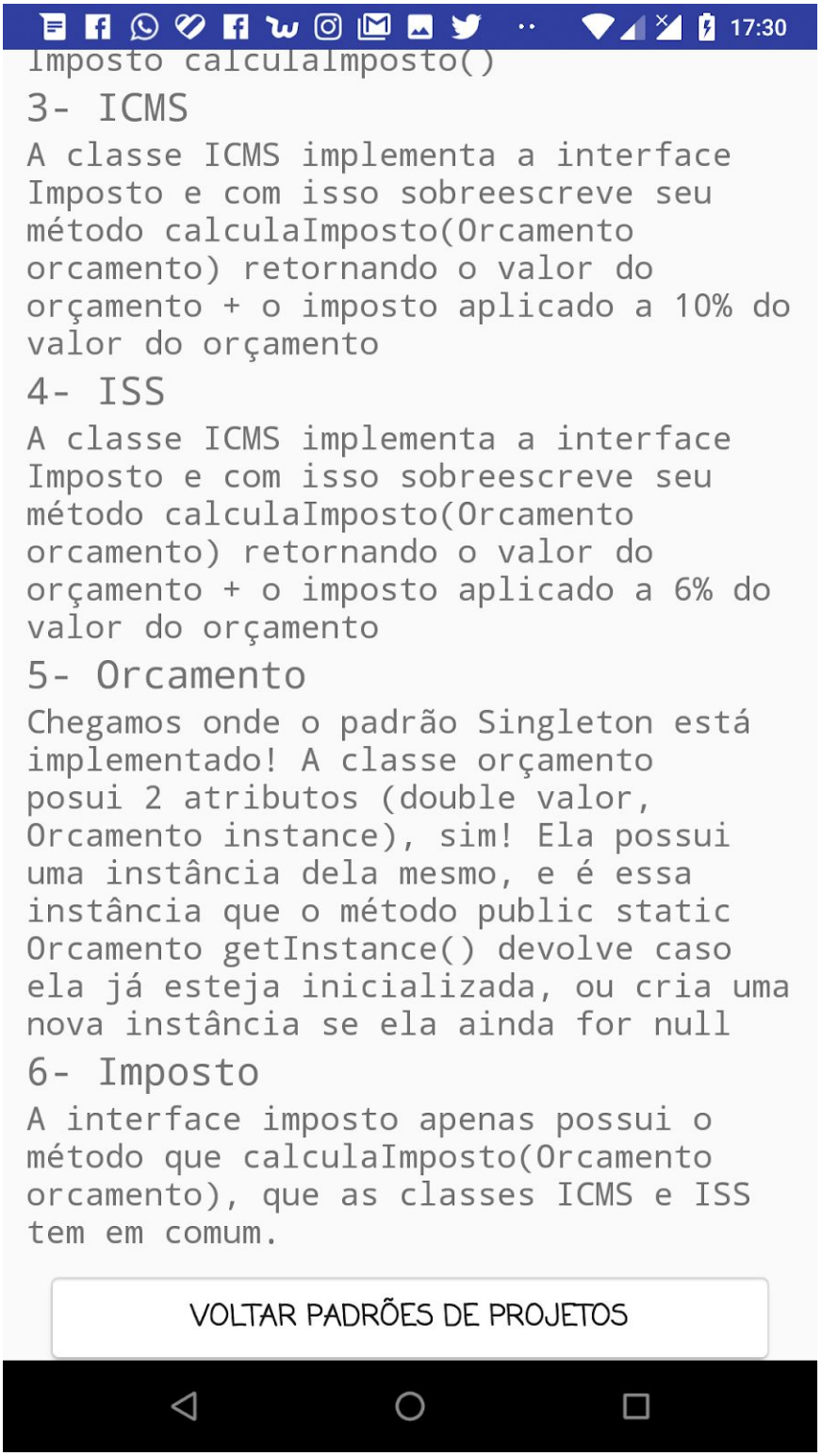
A classe ICMS implementa a interface Imposto e com isso sobreescreve seu método calculaImposto(Orcamento orcamento) retornando o valor do orçamento + o imposto aplicado a 10% do valor do orçamento

4- ISS

A classe ICMS implementa a interface Imposto e com isso sobreescreve seu método calculaImposto(Orcamento orcamento) retornando o valor do orçamento + o imposto aplicado a 6% do valor do orçamento

5- Orcamento





Existe 3 pacotes com os nomes dos padrões de projetos implementados onde estão suas implementações. A mainActivity está com MVVM, mas as demais não deu tempo de passar.