O usuário pode ter notificação ou não

Para isso vamos cadastrar mais 2 campos como celular e email

Pra isso adicionamos mais 2 campos na nossa classe abstrata

  String? \_nome;

  String \_sobrenome = "";

  String \_endereco = "";

  String \_celular = "";

  String \_email = "";

  TipoNotificacao \_tipoNotificacao = TipoNotificacao.NENHUM;

Também no construtor

Se caso você quiser entrar com esses dados direto no construtor

Se não você pode entrar depois usando o set...

  Pessoa(

    String nome,

    String sobrenome,

    String endereco, String celular, String email, {

    TipoNotificacao tipoNotificacao = TipoNotificacao.NENHUM,

  }) {

    \_nome = nome;

    \_sobrenome = sobrenome;

    \_endereco = endereco;

    \_tipoNotificacao = tipoNotificacao;

  }

Set e get

  void setCelular(String celular) {

    \_celular = celular;

  }

  void setEmail(String email) {

    \_email = email;

  }

  String getCelular() {

    return \_celular;

  }

  String getEmail() {

    return \_email;

  }

toString

  @override

  String toString() {

    // TODO: implement toString

    return {

      "Nome": getNome(),

      "Sobrenome": getSobrenome(),

      "Endereco": getEndereco(),

      "Celular" : getCelular(),

      "Email" : getEmail(),

      "TipoNotificacao" : getTipoNotificacao()

    }.toString(); // Nao podemos retornar um MAP aqui, para isso convertemos para toString.

  }

}

Mesma coisa vale para as outras classes, so adicione no construtor se você quiser entrar com esses dados direto la.

# Interface

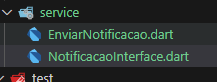
Interface é um contrato que devemos seguir, imagina que temos um contrato

Implemente um método para enviar mensagem automática

Todas as classes que tiverem aquela interface, deve ter aquele contrato.

A forma que vai implementar não importa para interface, mas deve ter aquele método lá.

Aqui estamos criando a classe e interface



No dart a interface fica dessa forma

import 'package:dart\_poo/classes/Pessoa.dart';

abstract class Notificacaointerface {

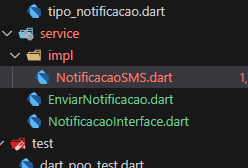
  void EnviarNotificacao(Pessoa pessoa);

}

Ela usa a palavra abstract

Agora criando classes que vai implementar a interface

# Classes



import 'package:dart\_poo/service/NotificacaoInterface.dart';

class Notificacaosms implements Notificacaointerface {

}

Devemos implementar o método da interface.

Criamos 3, mas vamos por apenas essa de exemplo

import 'package:dart\_poo/classes/Pessoa.dart';

import 'package:dart\_poo/service/NotificacaoInterface.dart';

class NotificacaoEmail implements Notificacaointerface{

  @override

  void EnviarNotificacao(Pessoa pessoa) {

    print("Enviando Email para : ${pessoa.getNome()}");

  }

}

Também criamos uma classe genérica de notificação que vai instanciar todas as outras.

import 'package:dart\_poo/Enum/tipo\_notificacao.dart';

import 'package:dart\_poo/classes/Pessoa.dart';

import 'package:dart\_poo/service/NotificacaoInterface.dart';

import 'package:dart\_poo/service/impl/NotificacaoEmail.dart';

import 'package:dart\_poo/service/impl/NotificacaoPush.dart';

import 'package:dart\_poo/service/impl/NotificacaoSMS.dart';

class Enviarnotificacao {

  Notificacaointerface? notificacao;

  // Aqui estamos falando que se o tipo da notificacao for email

  // Vamos instanciar o email

  // Se for celular

  // Vamos instanciar o celular

  void notificar(Pessoa pessoa) {

    switch (pessoa.getTipoNotificacao()) {

      case TipoNotificacao.EMAIL:

        notificacao = NotificacaoEmail();

        break;

      case TipoNotificacao.SMS:

        notificacao = Notificacaosms();

        break;

      case TipoNotificacao.PUSH\_NOTIFICACAO:

        notificacao = Notificacaopush();

        break;

      default:

        break;

    }

    if (notificacao != null) {

      notificacao!.EnviarNotificacao(pessoa);

    }

  }

}

Agora fazendo o teste...

Leia o código para entender mais depois.