entidades.agendamento_manutenção

```
from entidades.empresa_manutenção import get_empresas_manutenção
from entidades.hospital import get_hospitais
from entidades.equipamento_hospitalar import Tomográfo,
EquipamentoRessonânciaMagnética


agendamentos_manutenções = []

def get_agendamentos_manutenção():
    return agendamentos_manutenções
def inserir_agendamento_manutenção(agendamento_manutenção):
    if agendamento_manutenção not in agendamentos_manutenções:
        agendamentos_manutenções.append(agendamento_manutenção)
    else:
        print('Agendamento já tem cadastro --- ' + str(agendamento_manutenção))


def criar_agendamento_manutenção(nome_hospital, nome_empresa_manutenção, data):
    hospital = get_hospitais()[nome_hospital]
    if hospital is None:
        print('Hospital ' + nome_hospital + ' não cadastrada')
        return

    empresa_manutenção = get_empresas_manutenção()[nome_empresa_manutenção]
    if empresa_manutenção is None:
        print('Empresa de Manutenção ' + nome_empresa_manutenção + ' não cadastrada')
        return
    agendamento_manutenção = AgendamentoManutenção(hospital,
empresa_manutenção, data)
    inserir_agendamento_manutenção(agendamento_manutenção)


def filtrar_agendamentos_manutenções(data_máxima_agendamento_manutenção,
manutenção_em_dia_equipamento_hospitalar,

energia_máxima_raio_x_tomografo,tipo_imagem_ressonância_magnética,
                                prefixo_telefone_empresa_manutenção, uf_hospital):
    agendamentos_manutenções_selecionados = []
    for agendamento_manutenção in agendamentos_manutenções:
        if data_máxima_agendamento_manutenção is not None and
agendamento_manutenção.data > data_máxima_agendamento_manutenção:
            continue
        excluir_agendamento = False
        for equipamento_hospitalar in
agendamento_manutenção.hospital.equipamentos_hospitalar.values():
            if manutenção_em_dia_equipamento_hospitalar is not None and
equipamento_hospitalar.manutenção_em_dia !=
manutenção_em_dia_equipamento_hospitalar:
                excluir_agendamento = True
                break
            if isinstance(equipamento_hospitalar, Tomográfo):
                if energia_máxima_raio_x_tomografo is not None and
equipamento_hospitalar.energia_raio_x > energia_máxima_raio_x_tomografo:
```

```
                excluir_agendamento = True
                break
            elif isinstance(equipamento_hospitalar, EquipamentoRessonânciaMagnética):
                if tipo_imagem_ressonância_magnética is not None and
equipamento_hospitalar.tipo_imagem != tipo_imagem_ressonância_magnética:
                excluir_agendamento = True
                break
        if excluir_agendamento: continue

        if prefixo_telefone_empresa_manutenção is not None and not
agendamento_manutenção.empresa_manutenção.telefone.startswith(prefixo_telefone_e
mpresa_manutenção):
            continue
        if uf_hospital is not None and agendamento_manutenção.hospital.uf != uf_hospital:
            continue
        agendamentos_manutenções_selecionados.append(agendamento_manutenção)
    return agendamentos_manutenções_selecionados


class AgendamentoManutenção:
    def __init__(self, hospital, empresa_manutenção, data):
        self.hospital = hospital
        self.empresa_manutenção = empresa_manutenção
        self.data = data

    def __str__(self):
        formato = '{} {:<30} {} {:<24} {} {:<8} {}'
        agendamento_manutenção_formatado = formato.format('|', self.hospital.nome,
                                                          '|', self.empresa_manutenção.nome, '|', str(self.data),
'|')
        return agendamento_manutenção_formatado

    def str_atributos_equipamentos_hospitalar(self):
        atributos_equipamentos_hospitalares_str = ''
        tipos_imagem_ressonancia = []
        for índice, equipamento_hospitalar in
enumerate(self.hospital.equipamentos_hospitalar.values()):
            if not equipamento_hospitalar.manutenção_em_dia:
                atributos_equipamentos_hospitalares_str = 'Manutenção Necessária - '
            else:
                atributos_equipamentos_hospitalares_str = 'Não Necessaria - '

            if isinstance(equipamento_hospitalar, Tomográfo):
                atributos_equipamentos_hospitalares_str +=
f'{equipamento_hospitalar.energia_raio_x} mSv'
            elif isinstance(equipamento_hospitalar, EquipamentoRessonânciaMagnética):
                tipos_imagem_ressonancia.append(equipamento_hospitalar.tipo_imagem)

        if tipos_imagem_ressonancia:
            atributos_equipamentos_hospitalares_str += ' -- '.join(tipos_imagem_ressonancia)

        return atributos_equipamentos_hospitalares_str

    def str_filtro(self):
        formato = '{:<8} {} {:<35} {} {:<2} {} {:<12} {}'
        filtro_formatado = formato.format(str(self.data), '|',
self.str_atributos_equipamentos_hospitalar(), '|',
```

```
                                    self.hospital.uf, '|', self.empresa_manutenção.telefone, '|')
        return self.__str__() + filtro_formatado

def set_agendamentos_manutenção(agendamentos_manutenção1):
    global agendamentos_manutenção
    agendamentos_manutenção = agendamentos_manutenção1
```

entidades.empresa_manutenção

```
empresas_manutenção = {}

def get_empresas_manutenção(): return empresas_manutenção

def inserir_empresa_manutenção(empresa_manutenção):
    nome_empresa_manutenção = empresa_manutenção.nome
    if nome_empresa_manutenção not in empresas_manutenção.keys():
        empresas_manutenção[nome_empresa_manutenção] = empresa_manutenção
        return True
    else:
        print('Empresa de Manutenção ' + nome_empresa_manutenção + ' já tem cadastro')
        return False

def selecionar_empresas_manutenção(prefixo_telefone=None, prefixo_nome=None,
uf=None):
    filtros = '\nFiltros -- '
    if prefixo_telefone is not None:
        filtros += ' - Prefixo do Telefone: ' + str(prefixo_telefone)
    if prefixo_nome is not None:
        filtros += ' - Prefixo Nome: ' + str(prefixo_nome)
    if uf is not None:
        filtros += ' - Uf: ' + str(uf)

    empresas_manutenção_selecionadas = []
    for empresa_manutenção in empresas_manutenção:
        if prefixo_telefone is not None and not
empresa_manutenção.telefone.startswith(prefixo_telefone):
            continue
        if prefixo_nome is not None and not
empresa_manutenção.nome.startswith(prefixo_nome):
            continue
        if uf is not None and empresa_manutenção.uf != uf:
            continue
        empresas_manutenção_selecionadas.append(empresa_manutenção)
    return filtros, empresas_manutenção_selecionadas

class EmpresaManutenção:
    def __init__(self, cnpj, nome, telefone, email, fora_estado):
        self.cnpj = cnpj
        self.nome = nome
        self.telefone = telefone
        self.email = email
        self.fora_estado = fora_estado
        self.id = cnpj

    def __str__(self):
```

```
        if self.fora_estado:
            fora_estado_str = 'Fora do Estado'
        else:
            fora_estado_str = ''
        formato = '{} {:<18} {} {:<24} {} {:<9} {} {:<35} {} {:<14} {}'
        empresa_manutenção_formatado = formato.format('|', self.cnpj, '|', self.nome, '|',
self.telefone, '|', self.email, '|', fora_estado_str,'|')
        return empresa_manutenção_formatado

def set_empresas_manutenção(empresas_manutenção1):
    global empresas_manutenção
    empresas_manutenção = empresas_manutenção1
```

entidades.equipamento_hospitalar

```
equipamentos_hospitalar = []

def get_equipamentos_hospitalar():
    return equipamentos_hospitalar

def inserir_equipamento_hospitalar(equipamento_hospitalar):
    id_equipamento_hospitalar = equipamento_hospitalar.n_série
    if id_equipamento_hospitalar not in equipamentos_hospitalar.keys():
        equipamentos_hospitalar[id_equipamento_hospitalar] = equipamentos_hospitalar
        return True
    else:
        print('Equipamento' + id_equipamento_hospitalar + 'já tem cadastro')
        return False

def selecionar_equipamentos_hospitalar(fabricante=None, data_aquisição_maxima=None,
manutenção_em_dia=None):
    filtros = '\nFiltros -- '
    if fabricante:
        filtros += ' - Fabricante: ' + str(fabricante)
    if data_aquisição_maxima is not None:
        filtros += ' - Data da Aquisição: ' + str(data_aquisição_maxima)
    if manutenção_em_dia is not None:
        filtros += ' - Manutenção em Dia: ' + str(manutenção_em_dia)

    equipamentos_hospitalar_selecionados = []
    for equipamento_hospitalar in equipamentos_hospitalar:
        if fabricante is not None and equipamento_hospitalar.fabricante != fabricante:
            continue
        if data_aquisição_maxima is not None and
equipamento_hospitalar.data_aquisição.__lt__(data_aquisição_maxima):
            continue
        if manutenção_em_dia is not None and equipamento_hospitalar.manutenção_em_dia
!= manutenção_em_dia:
            continue
        equipamentos_hospitalar_selecionados.append(equipamento_hospitalar)
    return filtros, equipamentos_hospitalar_selecionados

class EquipamentoHospitalar:
    def __init__(self, n_série,  marca_modelo, fabricante, data_aquisição,
manutenção_em_dia):
        self.n_série = n_série
```

```
        self.marca_modelo = marca_modelo
        self.fabricante = fabricante
        self.data_aquisição = data_aquisição
        self.manutenção_em_dia = manutenção_em_dia
        self.id = n_série

    def __str__(self):
        if self.manutenção_em_dia:
            manutenção_em_dia_str = ''
        else:
            manutenção_em_dia_str = 'Manutenção Necessária'
        formato = '{} {:<6} {} {:<28} {} {:<14} {} {:<8} {} {:<21} {}'
        equipamento_hospitalar_formatado = formato.format('|', self.n_série,
'|',self.marca_modelo, '|', self.fabricante, '|',
                                                str(self.data_aquisição), '|', manutenção_em_dia_str,'|')
        return equipamento_hospitalar_formatado

class Tomográfo(EquipamentoHospitalar):
    def __init__(self, n_série,  marca_modelo, fabricante, data_aquisição,
manutenção_em_dia, energia_raio_x):
        super().__init__(n_série,  marca_modelo, fabricante, data_aquisição,
manutenção_em_dia)
        self.energia_raio_x = energia_raio_x

    def __str__(self):
        if self.manutenção_em_dia:
            manutenção_em_dia_str = ''
        else:
            manutenção_em_dia_str = 'Manutenção Necessária'
        formato = '{} {:<6} {} {:<28} {} {:<14} {} {:<8} {} {:<21} {} {:<5} {}'
        tomográfo_formatado = formato.format('|', self.n_série, '|', self.marca_modelo, '|',
                                                self.fabricante, '|',
                                                str(self.data_aquisição), '|', manutenção_em_dia_str, '|',
self.energia_raio_x, '|')
        return tomográfo_formatado

class EquipamentoRessonânciaMagnética(EquipamentoHospitalar):
    def __init__(self, n_série,  marca_modelo, fabricante, data_aquisição,
manutenção_em_dia,tipo_imagem):
        super().__init__(n_série,  marca_modelo, fabricante, data_aquisição,
manutenção_em_dia)
        self.tipo_imagem = tipo_imagem if tipo_imagem in ('T1', 'T2', 'Flair') else ''

    def __str__(self):
        if self.manutenção_em_dia:
            manutenção_em_dia_str = ''
        else:
            manutenção_em_dia_str = 'Manutenção Necessária'
        formato = '{} {:<6} {} {:<28} {} {:<14} {} {:<8} {} {:<21} {} {:<5} {}'
        EquipamentoRessonânciaMagnética_formatado = formato.format('|', self.n_série, '|',
self.marca_modelo, '|',
                                    self.fabricante, '|',
                                    str(self.data_aquisição), '|', manutenção_em_dia_str, '|',
                                    self.tipo_imagem, '|')
        return EquipamentoRessonânciaMagnética_formatado
```

entidades.hospital

```
hospitais = {}

def get_hospitais():
    return hospitais

def inserir_hospital(hospital):
    nome_hospital = hospital.nome
    if nome_hospital not in hospitais.keys():
        hospitais[nome_hospital] = hospital
        return True
    else:
        print('Hospital ' + nome_hospital + ' já tem cadastro')
        return False

class Hospital:
    def __init__(self, nome, entidade_mantenedora, cidade, uf):
        self.nome = nome
        self.entidade_mantenedora = entidade_mantenedora
        self.cidade = cidade
        self.uf = uf
        self.equipamentos_hospitalar = {}

    def __str__(self):
        formato = '{} {:<30} {} {:<40} {} {:<12} {} {:<2} {}'
        hospital_formatada = formato.format('|', self.nome, '|', self.entidade_mantenedora, '|',
self.cidade, '|', self.uf, '|')
        return hospital_formatada

    def inserir_equipamento_hospitalar(self, equipamento_hospitalar):
        id_equipamento_hospitalar = equipamento_hospitalar.n_série
        if id_equipamento_hospitalar not in self.equipamentos_hospitalar.keys():
            self.equipamentos_hospitalar[id_equipamento_hospitalar] =
equipamento_hospitalar
        else:
            print('Equipamento ' + id_equipamento_hospitalar + ' já tem cadastro no Hospital')

def set_hospitais(hospitais1):
    global hospitais
    hospitais = hospitais1
```

interfaces.interface_textual

```
from util.gerais import imprimir_objetos, imprimir_objetos_internos,
imprimir_objetos_associação_filtros, imprimir_objeto
from util.data import Data,  converte_str_para_data

from entidades.empresa_manutenção import inserir_empresa_manutenção,
EmpresaManutenção, get_empresas_manutenção
from entidades.equipamento_hospitalar import Tomográfo,
EquipamentoRessonânciaMagnética
from entidades.hospital import inserir_hospital, Hospital, get_hospitais
from entidades.agendamento_manutenção import criar_agendamento_manutenção,
```

```
get_agendamentos_manutenção, filtrar_agendamentos_manutenções,
inserir_agendamento_manutenção


def loop_opções_execução():
    sair_loop = False
    cabeçalho_empresa_manutenção = '\nEmpresas de Manutenção : cnjp - nome -
telefone - email - estado'
    cabeçalho_hospital_equipamentos_hospitalares = ('\nHospitais : nome - entidade
mantenedora - cidade - uf'
                                + '\n - Equipamentos : numero de série - marca modelo - fabricante
- data aquisição - manutenção em dia - Tomografo:[energia do raio x] | Equipamento de
Ressonância Magnética:[tipos de imagem]')
    cabeçalho_agendamento_manutenção = ('\nAgendamento de Manutenção : nome do
hospital - nome da empresa de manutenção'
                        + ' - data do agendamento')
    while not sair_loop:
        print()
        operação = ler_str('Opções [C: Cadastrar / I: Imprimir / S: Selecionar / T: imprimir
Todos / <ENTER>: Parar]', retornar=True)
        if operação == None:
            break
        elif operação in ('C', 'I'):
            opção_conteúdo = ler_str('E: Empresas de Manutenção / H: Hospitais / A:
Agendamentos de Manutenção / <ENTER>: retornar]', retornar=True)
            if opção_conteúdo == None:
                pass
            elif opção_conteúdo == 'E':
                if operação == 'C':
                    loop_leitura_empresas_manutenção()
                imprimir_objetos(cabeçalho_empresa_manutenção,
get_empresas_manutenção().values())
            elif opção_conteúdo in 'H':
                if operação == 'C':
                    loop_leitura_hospitais()

imprimir_hospitais_equipamentos_hospitalares(cabeçalho_hospital_equipamentos_hospita
lares)
            elif opção_conteúdo == 'A':
                if operação == 'C':
                    loop_leitura_agendamentos_manutenção()
                imprimir_objetos(cabeçalho_agendamento_manutenção,
get_agendamentos_manutenção())
            elif operação == 'S':
                loop_seleção_agendamentos_manutenção()
            elif operação == 'T':
                imprimir_objetos(cabeçalho_empresa_manutenção,
get_empresas_manutenção().values())

imprimir_hospitais_equipamentos_hospitalares(cabeçalho_hospital_equipamentos_hospita
lares)
                imprimir_objetos(cabeçalho_agendamento_manutenção,
get_agendamentos_manutenção())

def
imprimir_hospitais_equipamentos_hospitalares(cabeçalho_hospital_equipamentos_hospita
lares):
```

```
    print(cabeçalho_hospital_equipamentos_hospitalares)

    for índice, hospital in enumerate(get_hospitais().values()):
        imprimir_objeto(índice=índice, objeto_str=str(hospital))
        imprimir_objetos_internos(hospital.equipamentos_hospitalar.values())

def loop_leitura_empresas_manutenção():
    sair_loop = False
    print('--- Leitura de Dados das Empresas de Manutenção ---')
    while not sair_loop:
        empresa_manutenção = ler_empresa_manutenção()
        if empresa_manutenção is not None:
            inserir_empresa_manutenção(empresa_manutenção)
        else:
            print(' - ERRO : na leitura da empresa de manutenção')
        sair_loop = ler_sair_loop('cadastro de empresa_manutenção')

def loop_leitura_hospitais():
    sair_loop = False
    print('--- Leitura de Dados dos Hospitais ---')
    while not sair_loop:
        hospital = ler_hospital()
        if hospital is not None:
            inserir_hospital(hospital)
            loop_leitura_equipamentos_hospitalares_hospital(hospital)
        else:
            print(' - ERRO : na leitura do hospital')
        sair_loop = ler_sair_loop('cadastro de hospitais')

def loop_leitura_equipamentos_hospitalares_hospital(hospital):
    sair_loop = False
    print('--- Leitura de Dados dos Equipamentos do Hospital: ' + hospital.nome + ' ---')
    while not sair_loop:
        equipamento_hospitalar = ler_equipamento_hospitalar()
        if equipamento_hospitalar is not None:
            hospital.inserir_equipamento_hospitalar(equipamento_hospitalar)
        else:
            print(' - ERRO : na leitura de equipamento hospitalar')
        sair_loop = ler_sair_loop('cadastro de equipamentos do hospital')

def loop_leitura_agendamentos_manutenção():
    sair_loop = False
    print('--- Leitura de Dados de Agendamentos de manutenção ---')
    while not sair_loop:
        agendamento_manutenção = ler_agendamento_manutenção()
        if agendamento_manutenção is not None:
            inserir_agendamento_manutenção(agendamento_manutenção)
        else:
            print(' - ERRO : na leitura de agendamento de manutenção')
        sair_loop = ler_sair_loop('cadastro de agendamento de manutenção')

def ler_sair_loop(loop):
    try:
        sair = input('-- sair do loop de ' + loop + ' [S]: ')
        if sair == 'S':
            return True
    except IOError:
```

```
            pass
    return False

def loop_seleção_agendamentos_manutenção():
    sair_loop = False
    print('--- Seleção de Agendamentos de Manutenção ---')
    while not sair_loop:
        filtros, agendamentos_manutenção_selecionados =
selecionar_agendamentos_manutenções()
        if filtros is not None:
            cabeçalho = ('Agendamento Manutenção : manutenção em dia do equipamento
hospitalar - uf do hospital - prefixo do telefone da empresa de manutenção - data do
agendamento'
                         + '\n - Tomografo:[energia do raio x] | Equipamento de Ressonância
Magnética:[tipos de imagem]')
            imprimir_objetos_associação_filtros(cabeçalho,
agendamentos_manutenção_selecionados, filtros)
            sair_loop = ler_sair_loop('seleção de agendamento de manutenção')

def ler_empresa_manutenção():
    cnpj = ler_str('cnpj da empresa manutenção')
    if cnpj == None:
        return None
    nome = ler_str('nome da empresa manutenção')
    if nome == None:
        return None
    telefone = ler_str('telefone da empresa manutenção')
    if telefone == None:
        return None
    email = ler_str('email da empresa manutenção')
    if email == None:
        return None
    fora_estado = ler_bool('fora do uf da empresa manutenção')
    if fora_estado == None:
        return None
    return EmpresaManutenção(cnpj, nome, telefone, email, fora_estado)

def ler_hospital():
    nome = ler_str('nome do hospital')
    if nome == None:
        return None
    entidade_mantenedora = ler_str('entidade mantenedora do hospital')
    if entidade_mantenedora == None:
        return None
    cidade = ler_str('cidade do hospital')
    if cidade == None:
        return None
    uf = ler_str('UF do hospital')
    if uf == None:
        return None
    return Hospital(nome, entidade_mantenedora, cidade, uf)


def ler_equipamento_hospitalar():
    n_série = ler_str('número de série do equipamento')
    if n_série == None:
        return None
```

```
    marca_modelo = ler_str('marca modelo do equipamento')
    if marca_modelo == None:
        return None
    fabricante = ler_str('fabricante do equipamento')
    if fabricante == None:
        return None
    data_aquisição = ler_data('data de aquisição do equipamento')
    if data_aquisição is None:
        return None
    manutenção_em_dia = ler_bool('manutenção em dia do equipamento')
    if manutenção_em_dia == None:
        return None
    espécie_equipamento = ler_str('espécie do equipamento [Tg=Tomográfo /
Rm=EquipamentoRessonânciaMagnética]')
    if espécie_equipamento == 'Tg':
        energia_raio_x = ler_int_positivo('energia raio-x do Tomográfo (mSv)')
        if energia_raio_x == None:
            return None
        return Tomográfo(n_série,  marca_modelo, fabricante, data_aquisição,
manutenção_em_dia, energia_raio_x)
    if espécie_equipamento == 'Rm':
        tipo_imagem = ler_str('Tipo da Imagem do equipamento de ressonância magnética')
        if tipo_imagem == None:
            return None
        return EquipamentoRessonânciaMagnética(n_série,  marca_modelo, fabricante,
data_aquisição, manutenção_em_dia,tipo_imagem)
    else:
        return None

def ler_agendamento_manutenção():
    nome_hospital = ler_str('nome do hospital')
    if nome_hospital == None:
        return None
    nome_empresa_manutenção = ler_str('nome da empresa de manutenção')
    if nome_empresa_manutenção == None:
        return None
    data = ler_data('data do agendamento')
    if data is None:
        return None
    return criar_agendamento_manutenção(nome_hospital, nome_empresa_manutenção,
data)

def selecionar_agendamentos_manutenções():
    filtros = '\nFiltros -- '
    data_máxima_agendamento_manutenção = ler_data('Data Máxima do Agendamento de
Manutenção', filtro=True)
    if data_máxima_agendamento_manutenção is not None:
        filtros += 'Data Máxima do Agendamento de Manutenção: ' +
str(data_máxima_agendamento_manutenção)

    manutenção_em_dia_equipamento_hospitalar = ler_bool('Manutenção em dia do
Equipamento Hospitalar', filtro=True)
    if manutenção_em_dia_equipamento_hospitalar is not None:
        filtros += ' - Manutenção em dia do Equipamento Hospitalar: ' +
str(manutenção_em_dia_equipamento_hospitalar)

    energia_máxima_raio_x_tomografo = ler_int_positivo('Energia Raio X do Tomográfo',
```

```
filtro=True)
    if energia_máxima_raio_x_tomografo is not None:
        filtros += (' - Energia Raio X do Tomográfo: ' +
str(energia_máxima_raio_x_tomografo))

    tipo_imagem_ressonância_magnética = ler_str('Tipo da Imagem Do Equipamento de
Ressonancia Magnetica', filtro=True)
    if tipo_imagem_ressonância_magnética is not None:
        filtros += (' - Tipo da Imagem Do Equipamento de Ressonancia Magnetica: ' +
tipo_imagem_ressonância_magnética)

    prefixo_telefone_empresa_manutenção = ler_str('Prefixo Telefone Empresa de
Manutenção', filtro=True)
    if prefixo_telefone_empresa_manutenção is not None:
        filtros += '\n - Prefixo Telefone Empresa de Manutenção: ' +
str(prefixo_telefone_empresa_manutenção)

    uf_hospital = ler_str('UF do Hospital', filtro=True)
    if uf_hospital is not None:
        filtros += '\n - UF do Hospital: ' + uf_hospital
    agendamentos_manutenções_selecionados =
filtrar_agendamentos_manutenções(data_máxima_agendamento_manutenção,
manutenção_em_dia_equipamento_hospitalar,

energia_máxima_raio_x_tomografo,tipo_imagem_ressonância_magnética,
                                        prefixo_telefone_empresa_manutenção, uf_hospital)

    return filtros, agendamentos_manutenções_selecionados

def ler_str(dado, filtro=False, retornar=False):
    try:
        string = input('- ' + dado + ' : ')
        if len(string) == 0 and (filtro or retornar):
            return None
        if len(string) > 0:
            return string
    except IOError:
        pass
    print('Erro na leitura do dado: ' + dado)
    return None

def ler_int_positivo(dado, filtro=False):
    try:
        string = input('- ' + dado + ' : ')
        if len(string) == 0 and filtro:
            return None
        int_positivo = int(string)
        if int_positivo > 0:
            return int_positivo
    except ValueError:
        pass
    print('Erro na leitura/conversão do inteiro positivo: ' + dado)
    return None

def ler_float_positivo(dado, filtro=False):
    try:
        string = input('- ' + dado + ' : ')
```

```
        if len(string) == 0 and filtro:
            return None
        float_positivo = float(input('- ' + dado + ' : '))
        if float_positivo > 0.0:
            return float_positivo
    except ValueError:
        pass
    print('Erro na leitura/conversão do flutuante positivo: ' + dado)
    return None

def ler_bool(dado, filtro=False):
    try:
        string = input('- ' + dado + ' [S/N]: ')
        if len(string) == 0 and filtro:
            return None
        if string == 'S':
            return True
        elif string == 'N':
            return False
    except ValueError:
        pass
    print('Erro na leitura do booleano: ' + dado)
    return None

def ler_data(dado, filtro=False):
    try:
        string = input('- ' + dado + ' [dd/mm/aaaa]: ')
        if len(string) == 0 and filtro:
            return None
        data = converte_str_para_data(string)
        if data is not None:
            return data
    except IOError:
        pass
    print('Erro na leitura da data: ' + dado)
    return None
```

util.data

```
from datetime import date
from re import match
class Data:
    def __init__(self, dia, mês, ano):
        self.dia = dia
        self.mês = mês
        self.ano = ano

    def __str__(self):
        if self.dia < 10:
            data_str = '0' + str(self.dia)
        else:
            data_str = str(self.dia)
        if self.mês < 10:
            data_str += "/0" + str(self.mês) + "/"
        else:
            data_str += "/" + str(self.mês) + "/";
```

```
        data_str += str(self.ano)
        return data_str

    def __eq__(self, data):
        if self.dia == data.dia and self.mês == data.mês and self.ano == data.ano:
            return True
        return False

    def __ne__(self, data):
        return not self == data

    def __gt__(self, data):
        if self.ano > data.ano:
            return True
        elif self.ano < data.ano:
            return False
        if self.mês > data.mês:
            return True
        elif self.mês < data.mês:
            return False
        if self.dia > data.dia:
            return True
        elif self.dia < data.dia:
            return False
        return False

    def __lt__(self, data):
        if self.ano < data.ano:
            return True
        elif self.ano > data.ano:
            return False
        if self.mês < data.mês:
            return True
        elif self.mês > data.mês:
            return False
        if self.dia < data.dia:
            return True
        elif self.ano > data.ano:
            return False
        return False

    def __ge__(self, data):
        if self < data:
            return False
        else:
            return True

    def __le__(self, data):
        if self > data:
            return False
        else:
            return True

def calcular_idade(self, data_referência=None):
    if data_referência is None:
        dia_atual_str, mês_atual_str, ano_atual_str =
date.today().strftime("%d/%m/%Y").split('/')
```

```
        dia_referência, mês_referência, ano_referência = int(dia_atual_str), int(mês_atual_str),
int(ano_atual_str)
    else:
        dia_referência, mês_referência, ano_referência = data_referência.dia,
data_referência.mês, data_referência.ano
    idade = ano_referência - self.ano
    if mês_referência < self.mês or (mês_referência == self.mês and dia_referência <
self.dia):
        idade -= 1
    return idade

def converte_str_para_data(data_str):
    if not match('(0?[1-9]|[12][0-9]|3[01])/(0?[1-9]|1[012])/\d{4}', data_str):
        return None
    dia, mês, ano = data_str.split('/')
    return Data(int(dia), int(mês), int(ano))
```

util.gerais

```
def imprimir_objetos(cabeçalho, objetos, filtros=None):
    if filtros is not None:
        print(filtros)
    print(cabeçalho)
    for índice, objeto in enumerate(objetos):
        imprimir_objeto(índice, str(objeto))

def imprimir_objeto(índice, objeto_str):
    formato = '{} {} {}'
    ordem = índice + 1
    separador = '-'
    string_formatado = formato.format(f'{ordem:2d}', separador, objeto_str)
    print(string_formatado)

def imprimir_objetos_associação_filtros(cabeçalho, objetos, filtros=None):
    if filtros is not None:
        print(filtros)
    print(cabeçalho)
    for índice, objeto in enumerate(objetos):
        imprimir_objeto(índice, objeto.str_filtro())

def imprimir_objetos_internos(objetos):
    for objeto in objetos:
        print(' - ' + str(objeto))
```

util.persistência_arquivo

```
import pickle

def salvar_arquivo(nome_arquivo, objetos):
    arquivo = open('../../dados/' + nome_arquivo + '.bin', 'wb')
    pickle.dump(objetos, arquivo)
    arquivo.close()

def carregar_arquivo(nome_arquivo):
    try:
```

```
        arquivo = open('../../dados/' + nome_arquivo + '.bin', 'rb')
        objetos = pickle.load(arquivo)
    except IOError:
        objetos = None
    return objetos
```

controle.gerenciamento_hospital

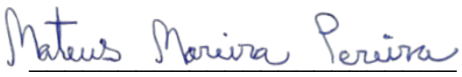
```
from util.persistência_arquivo import carregar_arquivo, salvar_arquivo
from entidades.empresa_manutenção import get_empresas_manutenção,
set_empresas_manutenção
from entidades.hospital import get_hospitais, set_hospitais
from entidades.agendamento_manutenção import get_agendamentos_manutenção,
set_agendamentos_manutenção
from interfaces.interface_textual import loop_opções_execução

nome_arquivo = 'agendamentos_manutenção'

def salvar_aplicação():
    agendamentos_manutenção = []
    agendamentos_manutenção.append(get_empresas_manutenção())
    agendamentos_manutenção.append(get_hospitais())
    agendamentos_manutenção.append(get_agendamentos_manutenção())
    salvar_arquivo(nome_arquivo, objetos=agendamentos_manutenção)

def recuperar_aplicação():
    agendamentos_manutenção = carregar_arquivo(nome_arquivo)
    if agendamentos_manutenção is not None:
        set_empresas_manutenção(agendamentos_manutenção[0])
        set_hospitais(agendamentos_manutenção[1])
        set_agendamentos_manutenção(agendamentos_manutenção[2])

if __name__ == '__main__':
    recuperar_aplicação()
    loop_opções_execução()
    salvar_aplicação()
```



Dourados, 01 de julho de 2024