Programação Orientada aos Objetos

Projeto Prático

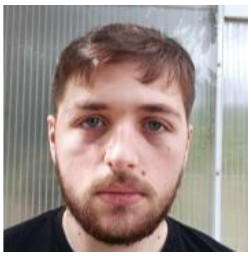
Aplicação Fitness

Universidade do Minho

Licenciatura em Ciências da Computação

Ano letivo 2023/2024

C. R. Faria
a105531



H. S. Marques
a102934

P. D. F. Bastos
a102502

1. Objetivos

Este projeto prático visa o desenvolvimento de uma aplicação Fitness construída na linguagem Java e implementando os princípios de Programação Orientada aos Objetos estudados nas aulas. Trata-se de uma aplicação de prática de exercício físico que permite a criação de perfis de utilizador, atividades físicas e planos de treino, completos com diferentes funcionalidades que serão detalhadamente descritas ao longo deste relatório.

2. Diagrama de Classes

Foi elaborado o seguinte Diagrama de Classes, que representa as classes e o funcionamento do programa:

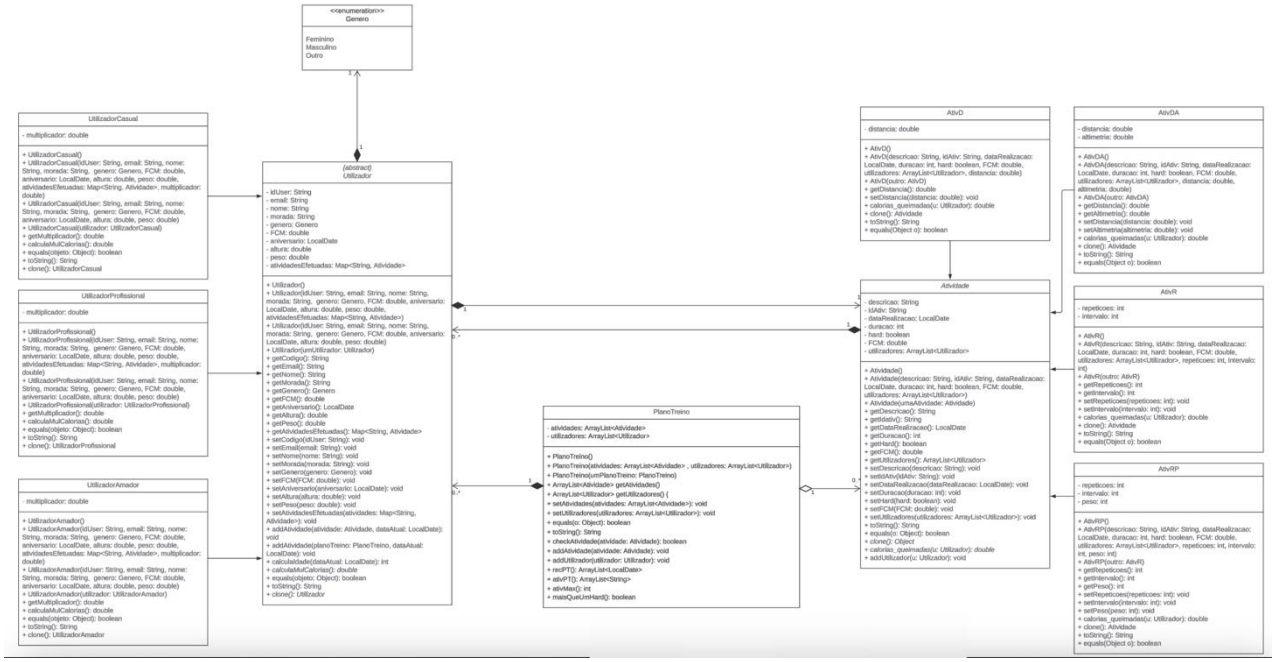


Figura 1 Diagrama de classes (Atividade, Utilizador, e PlanoTreino)

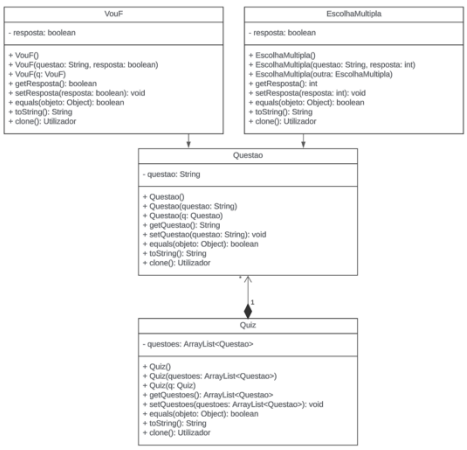


Figura 2 Diagrama de classes (Quiz)

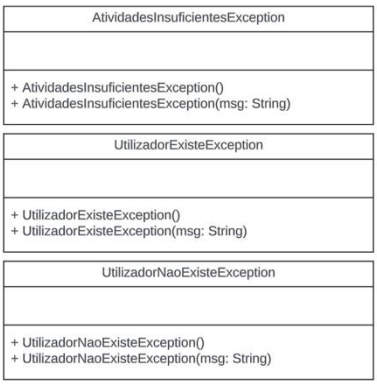


Figura 3 Diagrama de classes (Exceptions)

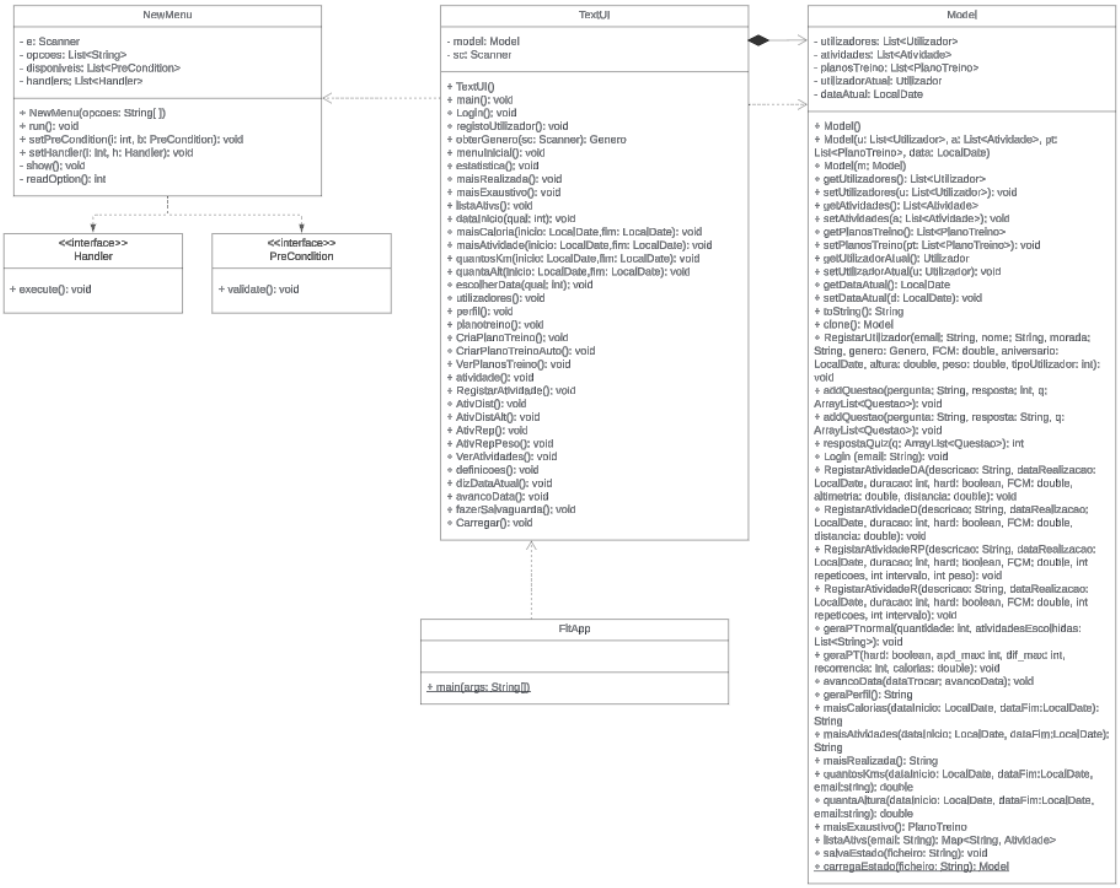


Figura 4 Diagrama de classes (FitApp, NewMenu, TextUI, Model)

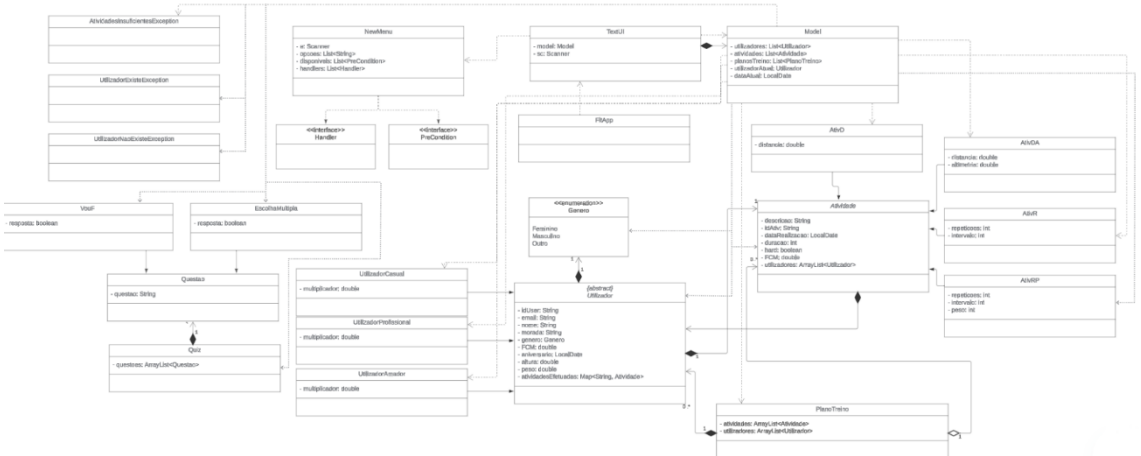


Figura 5 Diagrama de classes completo

3. Arquitetura de Classes

3.1 Classe Utilizador e subclasses

Implementou-se uma classe Utilizador abstrata, uma vez que esta classe não pode ser instanciada por si só mas serve como modelo para outras classes que a estendem. Nos seus atributos estão as variáveis presentes em todos os utilizadores (ID de utilizador, email, nome, morada, género, frequência cardíaca média, data de nascimento, altura, peso, e um Mapa de String e Atividade que contém as atividades efetuadas pelo utilizador).

A classe Utilizador estende a três subclasses públicas: **UtilizadorAmador, UtilizadorCasual e UtilizadorProfissional.** Cada um tem como variável um diferente multiplicador.

Esta atividade contém os métodos habituais: **gets, sets, equals, toString, clone**. Além destes, contém ainda os seguintes métodos:

- **calculaIdade**: recebe a data de nascimento do utilizador e devolve a idade atual;
- **calculaMulCalorias**: calcula o multiplicador do utilizador (será usado mais à frente neste relatório no cálculo de calorias).

Estas classes permitem criar diferentes tipos de utilizadores.

3.2 Classe Atividade e subclasses

Implementou-se uma classe Atividade abstrata, uma vez que esta classe não pode ser instanciada por si só mas serve como modelo para outras classes que a estendem. Nos seus atributos estão as variáveis presentes em todas as atividades (descrição, ID, data de realização, duração, classificação Hard, frequência cardíaca média durante a atividade, e a lista de utilizadores que realizam essa atividade).

A classe Atividade estende a quatro subclasses públicas: **AtivD** (atividade com distância), **AtivDA** (atividade com distância e altimetria), **AtivR** (atividade com repetições), **AtivRP** (atividade com repetições e pesos).

Além dos métodos habituais, esta classe contém ainda:

- **calorias_queimadas:** recebe um utilizador e calcula as calorias queimadas ao realizar esta atividade;
- **addUtilizador:** recebe um utilizador e adiciona-o à lista de utilizadores que realizam a atividade;

Estas classes permitem criar diferentes tipos de atividades.

3.3 Classe PlanoTreino

A classe PlanoTreino tem como atributos uma lista de atividades que compõe o plano e uma lista de utilizadores que realizam este plano de treino.

Além dos métodos habituais, esta classe contém ainda os seguintes métodos:

- **addAtividade:** recebe uma atividade e adiciona-a ao plano de treino;
- **addUtilizador:** recebe um utilizador e adiciona-o ao plano de treino;
- **recPT:** retorna uma lista que contém as datas em que as atividades do plano de treino são realizadas;
- **ativPT:** retorna uma lista que contém os diferentes tipos de atividades presentes no plano de treino;
- **ativMax**: retorna o número máximo de atividades realizadas num único dia no plano de treino;
- **maisQueUmHard**: verifica se existe mais do que uma atividade classificada como Hard num único dia no plano de treino;

Esta classe permite criar planos de treino com diferentes atividades e diferentes utilizadores associados.

3.4 Classes Quiz, Questão e Subclasses

A classe Questão tem como atributo uma String questao e contém os métodos habituais. A subclasse EscolhaMultipla tem como atributo uma resposta (número inteiro) e os métodos habituais. A sua subclasse VouF tem como atributo uma resposta (boolean), os métodos habituais e um método **valorResposta** que retorna o valor da resposta a esta pergunta.

A classe Quiz tem como atributos uma lista de questões e contém os métodos habituais. Além dos métodos habituais, contém os seguintes métodos:

- **calculaResultadoQuiz**: percorre as questões do quiz e retorna a soma dos valores das respostas.
- **takeQuiz**: este método chama o método calculaResultadoQuiz e consoante o resultado da soma anterior, devolve um número inteiro que identifica o tipo de utilizador.

Estas classes permitem a realização de um Quiz que determina o tipo de utilizador ao criar uma conta.

3.5 Menu, Interface Interativa e as suas Classes

A aplicação é ainda constituída pelas classes FitApp, Model, NewMenu e TextUI.

- **FitApp** inicializa aplicação.
- **NewMenu** gera um menu através de handlers.
- **Model** permite a chamada dos métodos das classes definidas e define ainda o utilizador atual (que fez login), a data atual, e os utilizadores, atividades e planos de treino existentes na aplicação.

- **TextUI** é o Controller/View, onde são chamados os métodos definidos em Model e onde se definem as opções dos menus e o seu funcionamento.

3.6 Exceptions

Quando algum erro ocorre, para evitar a quebra do programa, surge uma Exception que informa o utilizador do erro e permite o programa continuar.

Definiu-se Exceptions para os seguintes acontecimentos:

- Quando não existem atividades suficientes para gerar um plano de treino automático;
- Quando se tenta criar um utilizador que já existe;
- Quando se tenta aceder a um utilizador que não existe.

Além destas, são ainda incorporadas as seguintes Exceptions pré-definidas:

- FileNotFoundException;
- IOException (usadas no carregamento e na salvaguarda);
- ClassNotFoundException (usada no carregamento da salvaguarda);
- DateTimeException (usada quando é pedido ao utilizador que insira uma data e o utilizador responde com uma data inválida, exemplo: mês 15, dia 90)

4. Funcionalidades

4.1 Criação de Objetos

A aplicação permite a criação de diferentes utilizadores e, após o registo e login do utilizador, permite ainda uma fase de criação de atividades e planos de treino. O utilizador pode criar atividades e planos de treino personalizados manualmente ou pode ainda pedir à aplicação a criação automática de um plano de treino com base em requisitos que o utilizador forneça (esta funcionalidade é descrita em maior detalhe ao longo do relatório).

4.2 Cálculo de Calorias

O método calorias_queimadas, referido anteriormente, é implementado na subclasse da atividade em questão e recebe um utilizador como parâmetro. Esta função aplica a seguinte fórmula concebida pelos autores deste relatório:

- Para atividades com distância:

$$Calorias = MultiplicadorProprio * Distancia * 0.2 + Idade * 0.2 - 55/4$$

- Para atividades com distância e altimetria:

$$Calorias = MultiplicadorProprio + Distancia * 0.2 + Altimetria * 0.2 + Idade * 0.2 - 55/4$$

- Para atividades com repetições:

$$Calorias = MultiplicadorProprio + Repeticoes * 0.2 + Idade * 0.2 \ - 55/4$$

- Para atividades com repetições e pesos:

$$Calorias = MultiplicadorProprio + Repeticoes * Peso * 0.2 + Idade * 0.2 \ - 55/4$$

O valor do multiplicador próprio depende do tipo de utilizador (amador, casual ou profissional) e é calculado a partir da seguinte fórmula:

$$MultiplicadorProprio = 0.2 * Peso * Multiplicador$$

Sendo os valores de multiplicador:

- Para amador: 1.1
- Para casual: 1.3
- Para profissional: 1.5

4.3 Geração Automática de Plano de Treino

Para a aplicação gerar automaticamente um plano de treino, o utilizador deve antes responder a parâmetros para melhor ajustar o plano de treino às suas preferências.

A aplicação pergunta então ao utilizador:

- Se pretende incluir atividades com classificação Hard;
- Qual o número máximo de atividades que quer realizar por dia;
- Qual o número máximo de atividades diferentes quer;
- Qual a recorrência semanal das atividades (n vezes por semana);
- O consumo calórico mínimo que pretende ter.

Com base nas respostas, a aplicação gera planos de treino até obter um que satisfaça todos os requisitos. Note-se que esta funcionalidade só é possível após a criação de suficientes atividades. Para evitar que a função corra infinitamente em caso de disponibilidade insuficiente ou outro problema, foi estabelecido um limite máximo de 10 tentativas que aciona uma Exception e o método desiste.

4.4 Estatística

A aplicação permite calcular e apresentar estatísticas sobre o estado do programa. O utilizador pode consultar estatísticas sobre:

- O utilizador que mais calorias despendeu num período;
- O utilizador que mais atividades realizou num período;
- O tipo de atividade mais realizada;
- Quantos kms é que um utilizador realizou num período ou desde sempre;
- Quantos metros de altimetria é que um utilizar totalizou num período ou desde sempre;
- Qual o plano de treino mais exigente em função do dispêndio de calorias proposto;
- Listar as atividades de um utilizador.

O utilizador encontra esta opção no menu, podendo posteriormente escolher o período a ser contado para as estatísticas ou consultar as estatísticas desde sempre.

4.5 Avanço na Data

A aplicação permite, para maior conveniência, realizar avanços na data para simular a passagem de tempo. Esta opção encontra-se disponível a qualquer momento através do menu nas definições. O utilizador deve escolher uma nova data, e a aplicação passará a contar esta data como data atual, considerando todos os eventos (atividades e planos de treino) agendados até essa data como concluídos.

4.6 Salvaguarda do Estado

A salvaguarda do estado atual da aplicação encontra-se disponível a qualquer a partir do menu nas definições. Depois, para carregar o estado guardado, encontra-se uma opção de carregar no menu inicial.

5. Conclusão

Com a realização deste projeto prático podemos aplicar todos os conhecimentos obtidos nas aulas de Programação Orientada aos Objetos, nomeadamente classes, listas, Maps, e os princípios do paradigma de POO: Abstração, Encapsulamento, herança, Polimorfismo e Abstração de dados.