Universidade do Minho
Escola de Ciências

Programação Orientada aos Objetos

Universidade do Minho

2023/2024

Grupo 12

A97554	a95917	a100113
Marco Silva	Eduardo Soares	David Gonçalves

INTRODUÇÃO

AppTreinos é uma “aplicação” que tem como objetivo principal fazer a gestão de atividades e planos de treino, de praticantes de atividades físicas.

Os utilizadores registam-se, criando assim o seu próprio perfil. Perfil esse que contém alguns dados do usuário que podem ser alterados a qualquer momento. O usuário também tem a possibilidade de eliminar o seu perfil.

A *AppTreinos* dá a possibilidade ao usuário de selecionar as atividades que quer realizar. Não só a atividade, mas também a dificuldade e o tempo que o usuário quer dedicar para essa atividade. O programa também regista o consumo de calorias por cada atividade, ou seja, cada tipo de atividade tem uma fórmula diferente para o cálculo do consumo de calorias.

Cada usuário possui o seu próprio histórico de atividades. Junto de cada atividade é exposta a data de realização da mesma.

Neste projeto foi usado o sistema MVC (Model-View-Controller) na linguagem Java. Como editor de código, usámos tanto o IntelliJ como o Visual Studio Code.

ARQUITETURA DAS CLASSES

CLASSES MODEL

- **AppTreinos**

```
public class  AppTreinos {  11 usages
    private LocalDateTime date = LocalDateTime.now();  2 usages
    private LocalDateTime lastDateChange = LocalDateTime.now();  3 usages

    private Utilizador user;  3 usages
    private Atividade atividade;  3 usages
    private HashMap<String, Utilizador> users; // email → User  6 usages
    private HashMap<String, Atividade> atividades;  4 usages
```

Esta classe é responsável pelo monitoramento da "aplicação", ou seja, é responsável por adicionar e remover users (e-mail) ou atividades, e para verificar se a atividade escrita pelo usuário corresponde com as atividades disponíveis. Para além disso, tinha com objetivo registar alterações do tempo, o que acabou por não ser utilizado devido a não conseguir realizar saltos no tempo.

- **Atividade**

```
public class Atividade {  26 usages  4 inheritors
    private String codigo;  6 usages
    private String nome;  7 usages
    private int dificuldade; //(1-easy, 2-medium, 3-hard)  7 usages
    private int tempo;  7 usages
    private int consumoDeCalorias; // escolher como fazer por tipo de dificuldade  7 usages
```

Esta classe inicializa variáveis usadas para os tipos de atividades:

private String codigo: Código único gerado para cada atividade
private String nome: Nome da atividade
private int dificuldade: Tipo de dificuldade (easy,medium,hard)
private int tempo: Tempo que demora a atividade
private int consumoDeCalorias: Consumo de calorias de cada atividade

- **Distancia**

```
public class Distancia extends Atividade{  18 usages
    private String codigo;  no usages
    private String nome;  no usages
    private int dificuldade;  no usages
    private int distancia;  6 usages
    private int tempo;  no usages
    private int consumoDeCalorias;  no usages
```

Esta classe corresponde a um dos 4 tipos de atividade. Aqui é inicializada a variável *distancia*, que vai ser utilizada para o cálculo de calorias. Logo, a única diferença entre as classes Atividade e Distancia é a variável *distancia*.

- **Distancia_Altimetria**

```
public class Distancia_Altimetria extends Atividade{  18 usages
    private String codigo;  1 usage
    private String nome;  no usages
    private int dificuldade;  no usages
    private int distancia;  6 usages
    private int altitude;  6 usages
    private int tempo;  no usages
    private int consumoDeCalorias;  no usages
```

Esta classe corresponde a um dos 4 tipos de atividade. Para além da adição da variável *distancia*, a variável *altitude* também vai ser utilizada para o cálculo de calorias.

- **Historico**

```
public class Historico {  21 usages
    Atividade atividade;  6 usages
    LocalDateTime data;  6 usages
```

Esta classe é responsável por guardar as atividades realizadas com o seu respetivo tempo.

- **Objetivo**

```
public class Objetivo {  10 usages
    String atividade;  6 usages
    Integer caloriasParaGastar;  6 usages
```

Esta classe guarda os objetivos do usuário, dependendo de cada atividade e das calorias que o usuário quer gastar.

- **Series_Peso**

```
public class Series_Pesos extends Atividade{  20 usages
    private String codigo;  1 usage
    private String nome;  1 usage
    private int dificuldade;  1 usage
    private int tempo;  1 usage
    private int repeticoes;  7 usages
    private int peso;  2 usages
    private int consumoDeCalorias;  no usages
```

Esta classe corresponde a um dos 4 tipos de atividade. As variáveis *repeticoes* e *peso são* usadas para o cálculo de calorias.

- **Series_Repeticoes**

```
public class Series_Repeticoes extends Atividade{  20 usages
    private String codigo;  no usages
    private String nome;  no usages
    private int dificuldade;  no usages
    private int tempo;  no usages
    private int repeticoes;  6 usages
    private int consumoDeCalorias;  no usages
```

Esta classe corresponde a um dos 4 tipos de atividade. Tal como a classe Series_Peso, esta classe usa a variável *repeticoes* .

- **Utilizador**

```
public class Utilizador {
    private AppTreinos app;  no usages
    private String codigo;  5 usages
    private String nome;  6 usages
    private String morada;  6 usages
    private String email;  6 usages
    private int peso;  7 usages
    private int idade;  7 usages
    private int tipo; // profissional(3), amador(2), ocasional(1), null(0);  6 usages
    private ArrayList<Historico> historico; // atividade e data  5 usages
    private ArrayList<Objetivo> objetivos; //tipo de exercicio & calorias a gastar  5 usages
    private int frequenciaCardiaca;  6 usages
```

Esta classe é responsável por inicializar as variáveis correspondentes aos dados do perfil de cada usuário, dados esses que vão poder ser visíveis nas classes view.

Para além disso, são inicializados dois array lists *(classe Historico e classe Objetivo*), que são, respetivamente, o registo de atividades do utilizador, e os exercícios que o utilizador pretende fazer e as calorias que pretende gastar.

CLASSES VIEW

- **MainView**

```
public class MainView {  5 usages
    private PerfilView perfilview;  2 usages
    private PlanodeTreinoView planodetreinoview;  no usages
    private AppController appcontroller;  15 usages

    private boolean sair;  3 usages
```

Esta classe é responsável pela visualização do menu inicial, menu logged, menu das estatísticas e responsável por iniciar o menu do perfil. A classe também é responsável pela interação entre o usuário e programa, ou seja, regista e disponibiliza dados.

- **PerfilView**

```
public class PerfilView {  2 usages
    private AppController appController;  3 usages
```

Esta classe é apenas usada para a interface do menu de perfil. Tal como a classe MainView, disponibiliza os dados do perfil de cada usuário.

CLASSES CONTROLLER

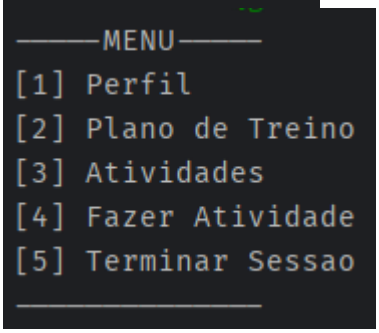
- **AppController**

```
public class AppController {  12 usages
    private AppTreinos app;  23 usages
```

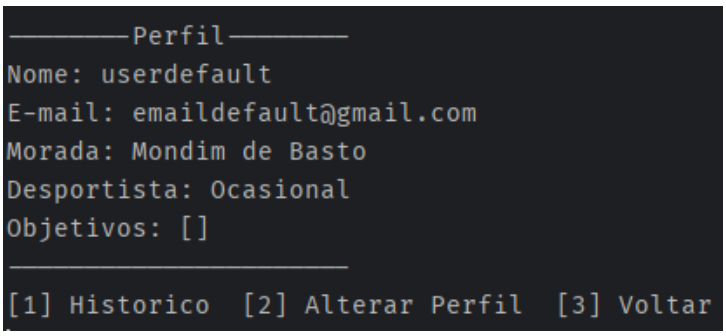
Esta classe é responsável por todas as comunicações entre as views e os models, assegurando o sistema MVC.

FUNCIONAMENTO

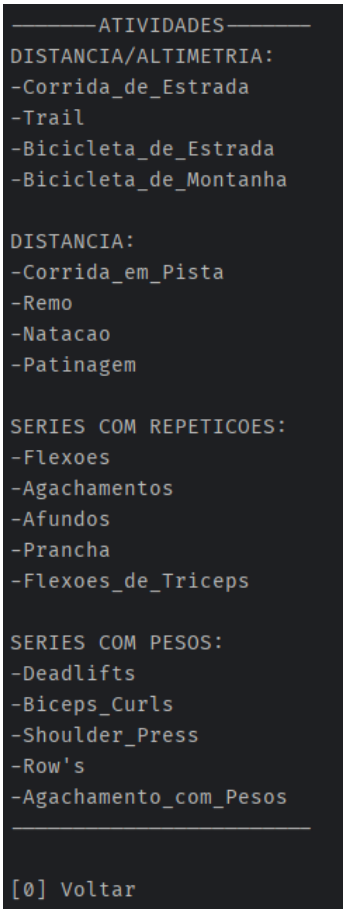
Login – Ao inserir o e-mail, entra no menu perfil do usuário.
Registar – Cria um perfil com dados solicitados pelo programa.
Atividades – Mostra as atividades do programa que podem ser realizadas.
Estatística - Mostra os rankings do programa.
Sair – Termina o programa



Perfil – Mostra os dados do utilizador.
Plano de Treino - Inicializa um novo plano de treino.
Atividades - Mostra as atividades do programa que podem ser realizadas pelo usuário.
Fazer Atividade – Executa uma atividade (e esta fica registada).
Terminar sessão - Volta para o menu principal



Disposição do Perfil do usuário userdefault.
Histórico - Vai direto para o histórico do usuário.
Alterar Perfil – O programa altera cada dado do usuário (à exceção do e-mail) de acordo com o que ele escreve.
Voltar- Volta para o menu perfil.



Disposição dos 4 tipos de atividades e das suas respetivas atividades.

CONCLUSÃO

Criar um programa de treino orientado aos objetos foi desafiante. Conseguimos criar um menu e um perfil dinâmico, onde é possível realizar o objetivo principal no projeto. Porém não atingimos todos os objetivos solicitados no enunciado do trabalho prático.

Uma das maiores dificuldades do grupo, foi respeitar as regras da programação orientada a objetos. O que implicou refazer muitas classes e mudar a estrutura do nosso trabalho.

Em suma, achamos o trabalho uma ótima introdução, tanto à programação orientada aos objetos, como à própria linguagem Java. Embora não tenhamos ficado satisfeitos com o projeto final, conseguimos obter experiência e conhecimento.