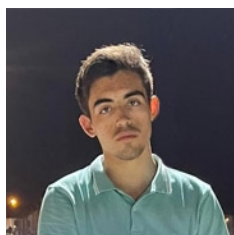


Universidade do Minho
Licenciatura em Engenharia Informática

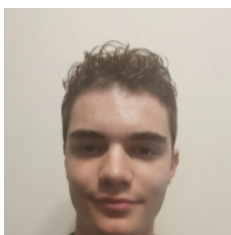
TRABALHO PRÁTICO

Programação Orientada aos Objetos

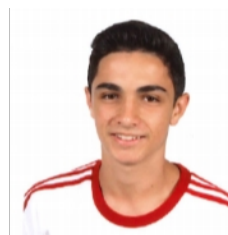
FITNESS HUB
GRUPO 31



a104350
Gonçalo
Freitas



A104358
Alexandre
Monsanto



A104539
João
Ferreira

12 de maio de 2024

Conteúdo

1	Introdução	2
2	Arquitetura	2
3	Diagrama De Classes	3
4	Classes	4
4.1	Model	4
4.1.1	Utilizador	4
4.1.2	Atividade	4
4.1.3	Distancia	5
4.1.4	DistanciaAltimetria	5
4.1.5	SeriesRepeticoes	5
4.1.6	SeriesRepeticoesPeso	5
4.1.7	PlanodeTreino	6
4.1.8	Aplicacion	6
4.1.9	DefaultCreator	7
4.2	View	7
4.3	Controller	7
5	View	7
6	Controller	8
7	Execução	8
7.1	Carregamento de Dados	8
7.2	LogIn/ SignIn	9
7.3	Menu Principal	10
7.4	Criar Atividade	10
7.4.1	Distancia	11
7.4.2	Distancia e Altimetria	12
7.4.3	Series de Repetições	13
7.4.4	Series de Repetições com Peso	14
7.5	Criar Utilizador	14
7.6	Mudança de Data	15
7.7	Guardar num ficheiro de Objetos	15
8	Conclusão	15

1 Introdução

Este trabalho tem como principal objetivo a criação de uma aplicação de fitness para gerir utilizadores, atividades e planos de treino.

A aplicação permite a utilização de vários utilizadores em que cada utilizador possui algumas capacidades na gestão das atividades e dos planos de treino.

A aplicação que foi dado o nome de "Aplication" mantém o controlo das informações relativas aos utilizadores, atividades e planos de treino. Algumas das capacidades faladas anteriormente serão a criação de novas atividades e utilizadores, a simulação da passagem do tempo conseguindo passar para uma data específica a aplicação e a possibilidade de guardar o estado da aplicação para mais tarde mesmo depois de terminar o programa a utilizar.

Este relatório abordará detalhadamente os requisitos funcionais do sistema a sua arquitetura e as tecnologias e metodologias usadas.

2 Arquitetura

A estrutura deste programa segue o padrão MVC(Model - View - Controller). Isso significa que ele divide-se em três partes principais: o Model, que cuida das regras e cálculos; a View, que lida com a interação com o usuário; e o Controller, que conecta a View ao Model. Com isso, o código fica mais organizado e é mais fácil encontrar funções quando necessário.

4 Classes

4.1 Model

4.1.1 Utilizador

```
private static int counter;  
private String codigoUtilizador;  
private String nome;  
private String morada;  
private String email;  
private double frequenciaCardiacaMedia;  
private Tipo tipo;  
private Set<Atividade> atividadesCompletadas;  
public enum Tipo {AMADOR, PRATICANTEOCASIONAL, PROFISSIONAL, NULL}
```

Classe que representa um utilizador

As variáveis de instância apresentadas tem o seguinte significado:

- counter - conta a quantidade de utilizadores é que existe na aplicação
- codigoUtilizador - possui o código do utilizador(unico para cada utilizador)
- nome - nome do utilizador
- morada - morada do utilizador
- email - email do utilizador
- frequenciaCardiacaMedia - a frequencia cardiaca média(calculada em função das atividades)
- tipo - tipo do utilizador(Amador, praticante ocasional ou profissional)
- atividadesCompletadas - conjunto de atividades que o utilizador já realizou

4.1.2 Atividade

```
private String codigo;  
private int duracao;  
private int caloriasGastas;  
private Dificuldade dificuldade;  
public enum Dificuldade {HARD,NOTCLASSIFIED }
```

Esta é a super classe Atividade das classes Distancia, DistanciaAltimetria, SeriesRepeticoes e SeriesRepeticoesPeso faz sentido utilizar este conceito de herança porque todas estas atividades possuem características que as diferem mas que vão utilizar características da super class.

As variáveis de instância apresentadas tem o seguinte significado:

- codigo - é o nome da atividade/código
- duracao - é o tempo que se demora a realizar essa atividade
- caloriasGastas - numero de calorias gastas ao realizar essa atividade
- Dificuldade - dificuldade da atividade(HARD, NOTCLASSIFIED)

4.1.3 Distancia

```
private int distanciaPercorrida;
```

Esta é a class representa Atividades do tipo Distancia
As variáveis de instância apresentadas tem o seguinte significado:

- distanciaPercorrida - é a distância percorrida nessa atividade

4.1.4 DistanciaAltimetria

```
private int altura;
```

Esta é a class representa Atividades do tipo DistanciaAltimetria
As variáveis de instância apresentadas tem o seguinte significado:

- altura - é a altura(altitude) percorrida nessa atividade

4.1.5 SeriesRepeticoes

```
private int numeroRepeticoes;  
private int numerSeries;
```

Esta é a class representa Atividades do tipo SeriesRepeticoes
As variáveis de instância apresentadas tem o seguinte significado:

- numeroRepeticoes - é o número de repetições realizado naquela atividade
- numeroSeries - é o número de series realizado naquela atividade

4.1.6 SeriesRepeticoesPeso

```
private int peso;
```

Esta é a class representa Atividades do tipo SeriesRepeticoesPeso
As variáveis de instância apresentadas tem o seguinte significado:

- peso - é o peso utilizado nessa atividade

4.1.7 PlanodeTreino

```
private static int count;  
private int codigo;  
private String nome;  
private Set<Atividade> atividades;  
private Set<Utilizador> utilizadores;  
private Date data;  
private int nVezes;
```

Esta é a class representa o plano de atividades

As variáveis de instância apresentadas tem o seguinte significado:

- count - é o número de planos de treino
- codigo - é o código do plano de treino específico
- nome - é o nome do plano de treino
- atividades - são as atividades presentes nesse plano de treino
- utilizadores - são os utilizadores que estão a utilizar esse plano de treino
- data - é a data em que se está a utilizar o programa
- nVezes - é o numero de vezes que aquele treino será realizador

4.1.8 Aplicacion

```
private Map<String, Atividade> atividades;  
private Map<String, Utilizador> utilizadores;  
private Map<String, PlanoDeTreino> planosdetreino;  
private LocalDate data;
```

Esta é a class representa a todas as atividades, planos de atividades e utilizadores presentes num dado momento

As variáveis de instância apresentadas tem o seguinte significado:

- atividades - são as atividades que existem na aplicação que são procuradas utilizando o código da atividade
- utilizadores - são os utilizadores que existem na aplicação que vamos buscar utilizando o código de utilizador
- planosdetreino - são os planos de treino que existem na aplicação que são procurados utilizando o código
- data - é a data em que está a aplicação

4.1.9 DefaultCreator

A classe DefaultCreator cria os objetos necessários das variadas classes para popular a aplicação com uma quantidade de atividades, utilizadores e planos de treino

4.2 View

```
private Application application;
```

A classe view só possui uma variável de instancia que é a application que foi utilizada uma vez para confirmar a existencia de utilizadores além disso o seu propósito é servir de "interface" entre o utilizador e o programa.

4.3 Controller

```
private View view;  
private Application application;  
private String currentUser;  
private boolean run;  
private boolean logged ;  
private LocalDateTime data;
```

O controller terá apenas um objeto que é fazer a ligação entre a classe Application e View. As variaveis run e logged server para controlar quando o programa está a rodar(run) e se um utilizador está logged ou não.

5 View

A classe View é como o intermediário entre o usuário e o aplicativo. Ela cuida de mostrar os menus e informações para o usuário, além de receber qualquer entrada que o usuário faça. Além disso, quando o usuário interage, a classe View garante que apenas valores válidos e do tipo correto sejam aceitos. Por exemplo, se o usuário deve inserir um número, a classe View se certifica de que apenas números sejam aceitos e não letras. Isso ajuda a manter a integridade e a funcionalidade da aplicação.

6 Controller

O Controller é como o "maestro" que coordena a comunicação entre o Model e a View. Ele usa a View para interagir com o usuário, recebendo dados dele e enviando-os para serem armazenados ou processados pelo Model, se necessário. Além disso, ele pode enviar esses dados de volta para a View para que o usuário os veja.

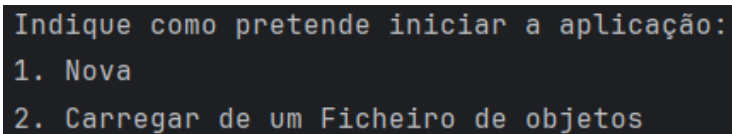
Basicamente, o Controller atua como um intermediário entre os aspectos funcionais e visuais do programa. Ele conhece todos os detalhes da implementação e contém métodos que controlam a exibição de vários menus. Ele também garante que o programa seja robusto, usando estruturas como switches e exceções para evitar encerramentos inesperados.

Além disso, o Controller mantém informações como o usuário que fez login, permitindo que vários usuários usem o programa ao mesmo tempo. Existem dois booleanos que indicam se alguém está logado e se o programa deve ser encerrado. Isso ajuda a simular uma experiência realista de uma aplicação e garante a segurança e a funcionalidade contínua do programa.

7 Execução

7.1 Carregamento de Dados

No arranque do programa encontram-se duas opções de carregamento de dados, carregar dados usando a classe DefaultCreator, que funciona como um script, ou o carregamento do estado a partir de um ficheiro de objetos recorrendo ao processo de serialização.



```
Indique como pretende iniciar a aplicação:  
1. Nova  
2. Carregar de um Ficheiro de objetos
```

Figura 2: Menu Carregamento de dados

7.2 LogIn/ SignIn

Depois dos dados serem carregados o utilizador pode fazer o login se já existir numero de utilizador nos dados da aplicação ou fazer sign in se ainda não tiver conta.

```
=====
1. Login/Sign-in
0. Terminar Programa
```

Figura 3: Menu de login

Se o utilizador escolher a primeira opção será então pedido o seu código de utilizador, caso o utilizador já exista então o login é efetuado, se não existir será então pedida mais informação ao utilizador para criar a sua conta.

```
==Utilizador==
Insira o código de utilizador:
```

Figura 4: Código de utilizador

Caso não exista esse utilizador:

```
Vamos criar uma nova conta!
Informação sobre a sua nova conta!
==Utilizador==
Insira o código de utilizador:
10
Insira o seu nome: teste
Insira a sua morada: rua
Insira o seu email: gmail
Insira o seu tipo:
1 - Amador
2 - Praticante Ocasional
3 - Profissional
```

Figura 5: Criar Conta

7.3 Menu Principal

Após o login efetuado é apresentado ao utilizador todas as operações que pode realizar na aplicação, sendo elas as opções representadas na Fig 6:

```
====Fitness HUB====  
1 - Criar Atividade  
2 - Criar Utilizador  
3 - Criar Plano de Treino  
4 - Guardar num ficheiro de objetos  
5 - Calcular Estatísticas  
6 - Mudar Data  
7 - Escreve ficheiro de txt  
0 - Terminar Sessão  
Indique a opcao:
```

Figura 6: Hub

Vamos explicar nos próximos pontos as opções funcionais na nossa aplicação.

7.4 Criar Atividade

Primeiro perguntamos ao utilizador que tipo de atividade pretende criar

```
=====  
Que tipo de Atividade queres criar?  
1. Distancia  
2. Distancia e Alimetria  
3. Series de repeticoes  
4. Series de repeticoes com peso  
0 - Voltar ao Menu Inicial
```

Figura 7: Tipo de atividade

7.4.1 Distancia

Caso seja escolhida a opção distancia(1) vai para este menu:

```
Vamos criar uma nova atividade!!  
Qual o nome da atividade:  
corrida  
Qual a duração da atividade(minutos):  
10  
Quantas calorias pretende queimar(kcal):  
10  
Qual a distancia que será percorrida(km):  
10  
Esta atividade é do tipo HARD?  
1 - Sim  
2 - Não  
1  
Atividade Criada
```

Figura 8: Criar atividade Distancia

7.4.2 Distancia e Altimetria

Caso seja escolhida a opção distancia e altimetria(2) vai para este menu:

```
Vamos criar uma nova atividade!!  
Qual o nome da atividade:  
b  
Qual a duração da atividade(minutos):  
10  
Quantas calorias pretende queimar(kcal):  
10  
Qual a distancia que será percorrida(km):  
10  
Qual a altura percorrida:  
20  
Esta atividade é do tipo HARD?  
1 - Sim  
2 - Não  
2  
Atividade Criada
```

Figura 9: Criar atividade Distancia e Altimetria

7.4.3 Series de Repetições

Caso seja escolhida a opção series de repetições(3) vai para este menu:

```
Vamos criar uma nova atividade!!  
Qual o nome da atividade:  
s  
Qual a duração da atividade(minutos):  
10  
Quantas calorias pretende queimar(kcal):  
10  
Esta atividade é do tipo HARD?  
1 - Sim  
2 - Não  
2  
Quantas series vão ser realizadas:  
10  
Número de repetições em cada serie:  
10  
Atividade Criada
```

Figura 10: Criar atividade Series de repetições

7.4.4 Series de Repetições com Peso

Caso seja escolhida a opção series de repetições com peso(4) vai para este menu:

```
Vamos criar uma nova atividade!!  
Qual o nome da atividade:  
f  
Qual a duração da atividade(minutos):  
10  
Quantas calorias pretende queimar(kcal):  
10  
Qual o Peso usado(kg):  
10  
Esta atividade é do tipo HARD?  
1 - Sim  
2 - Não  
1  
Quantas series vão ser realizadas:  
10  
Número de repetições em cada serie:  
10  
Atividade Criada
```

Figura 11: Criar atividade Series de Repetições com peso

7.5 Criar Utilizador

Primeiro pedimos o código de utilizador e caso exista dá este "erro":

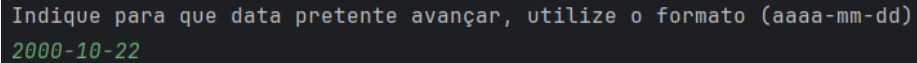
```
==Utilizador==  
Insira o código de utilizador:  
2  
Utilizador com esse código já existe!
```

Figura 12: Utilizador já existente

Caso não exista irá mostrar como na Figura 5

7.6 Mudança de Data

Muda a data da aplicação para a data que o utilizador decidir:



```
Indique para que data pretende avançar, utilize o formato (aaaa-mm-dd)  
2000-10-22
```

Figura 13: Mudança de data

7.7 Guardar num ficheiro de Objetos

Ele simplesmente guarda o estado na aplicação num ficheiro de objetos que pode ser utilizado mais tarde (como na opção 2 da Figura 2).

8 Conclusão

Este projeto nos ajudou a aprofundar vários conceitos da programação orientada a objetos. Uma técnica importante que aprendemos foi o encapsulamento, que significa não acessar diretamente os dados, mas sim usar métodos para isso.

Além disso, exploramos o conceito de herança, que nos permitiu economizar tempo e evitar a reescrita extensiva de código. Com a herança, podemos fazer com que objetos herdem características e métodos de outros, o que torna o código mais eficiente e fácil de manter.

Com este projeto, conseguimos identificar quais métodos da programação orientada a objetos são mais adequados para diferentes situações e quais devemos evitar. Isso ajudou nos a desenvolver uma compreensão mais sólida desses conceitos e a aplicá-los de forma mais eficaz em projetos futuros.