Projeto Prático

Unidade Curricular de
Programação Orientada aos Objetos

Grupo 10:

Ana Sá Oliveira a104437

Inês Silva Marques a104263

José Rafael de Oliveira Vilas Boas a76350

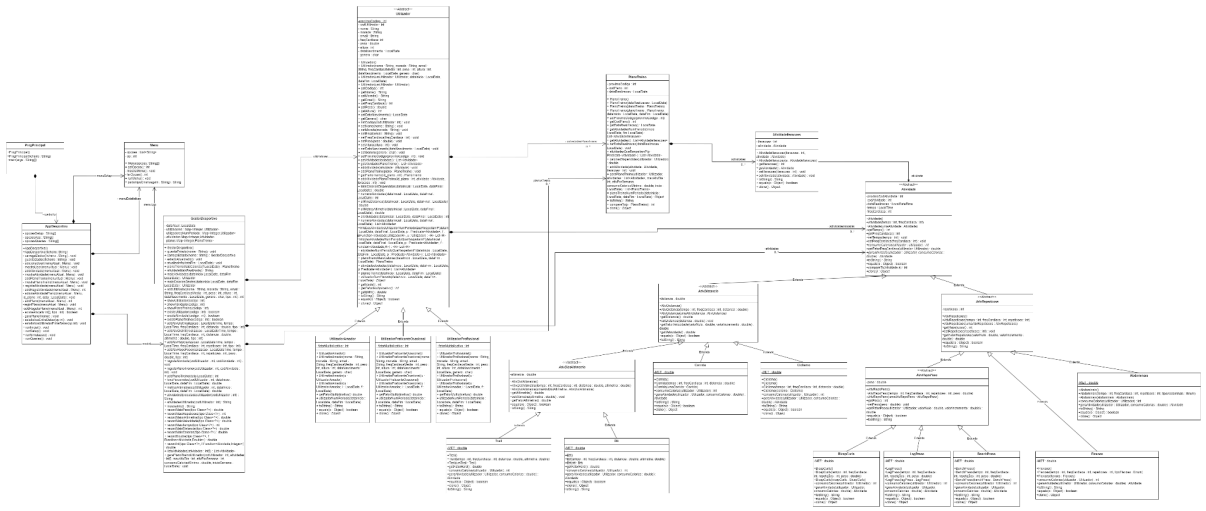
2023/2024

Índice

Introdução

Neste trabalho, desenvolvemos uma aplicação que faz a gestão das atividades e planos de treino, de praticantes de atividades físicas. Esta aplicação foi feita com a linguagem de programação Java. O objetivo deste trabalho era criar uma aplicação que, não só funcionasse como também respeitasse o paradigma da programação orientada aos objetos.

Arquitetura/diagrama de classes



Atividades

A classe Atividade serve como uma estrutura básica para representar diferentes tipos de atividades desportivas na aplicação. É uma classe abstrata, o que significa que não pode ser instanciada diretamente, mas fornece um molde para as suas subclasses. Contém atributos e métodos comuns a todos os tipos de atividades: o identificador da atividade, a data e hora da sua realização, a duração da atividade e a frequência cardíaca média durante a sua realização.

Como subclasses diretamente abaixo de Atividade temos os vários tipos de atividades, aquelas em que é relevante saber a distância percorrida na sua realização e aquelas em que devemos saber o número de repetições efetuadas. Estas classes também são abstratas e têm como variáveis de instância (para além das que herdam de Atividade) ‘distancia’ do tipo double, e ‘repeticoes’ do tipo inteiro, respetivamente.

Depois, como subclasse de AtivDistancia temos as classes (não abstratas) Corrida e Ciclismo, que são as atividades desportivas concretas que vão ser instanciadas, e a classe

abstrata AtivDistAltimetria que engloba as atividades onde é relevante saber os metros de altimetria acumulados na sua realização. AtivDistAltimetria tem como subclasses as atividades Btt e Trail, que já não são abstratas.

Algo semelhante acontece na classe AtivRepeticoes, com as suas subclasses: Flexoes e Abdominais, e a classe abstrata AtivRepsPeso, que adiciona a variável peso do tipo double, e tem como subclasses BenchPress, LegPress e BicepCurls, todos exercícios com um número de repetições e um peso utilizado.

Cálculo Calorias

O cálculo de calorias é feito da seguinte forma:
Calorias = MET * Soma(Fatores) * BMR * Tempo(min) / (24 * 60)

MET:
MET = Metabolic equivalent of task, é um multiplicador que reflete a razão entre a quantidade de energia utilizada enquanto se está parado e enquanto se faz algum tipo de atividade. 1 MET = energia gasta quando parado, um exercício com 2 MET gasta o dobro da energia que se gasta quando parado.
Também depende da intensidade a que se faz a atividade, por isso irá ser decomposto em duas partes, o MET da atividade com a intensidade mais baixa e vários multiplicadores conforme as características do utilizador, o que irá aumentar a intensidade da atividade, ficando MET(intensidade baixa) * (Soma(Fatores(utilizador)))

BMR:
BMR = Basal metabolic rate, é a energia (em calorias) gasta num dia enquanto parado e é calculada através da fórmula de Harris-Benedict:
BMR = (10*kg + 6.25*cm - 5*anos + s)
s = 5 para homens e s = -161 para mulheres

Atividades hard

Para distinguir atividades Hard daquelas que não são, criamos uma interface chamada Hard. Assim, podemos associar qualquer atividade a esta interface sem grandes dificuldades.

No caso da nossa aplicação, as classes que implementam Hard são a classe Btt e Trail.

Assim, quando é calculado o gasto de calorias destas atividades é adicionado um fator multiplicativo que aumenta esse gasto, por serem atividades mais difíceis.

Utilizadores

A classe Utilizador serve como uma estrutura básica para representar diferentes tipos de utilizadores na aplicação. É uma classe abstrata, o que significa que não pode ser instanciada diretamente, mas fornece um molde para as suas subclasses. Contém variáveis e métodos comuns a todos os tipos de utilizadores. Guarda dados de contacto dos utilizadores (morada, email) e informações que impactam o cálculo das calorias das atividades que realizam (frequência cardíaca média, peso, entre outras).

A classe Utilizador tem 3 subclasses, que representam os vários tipos de utilizador na aplicação, os utilizadores amadores, os praticantes ocasionais e os utilizadores profissionais. Esta distinção de tipo de utilizador impacta o cálculo das calorias de uma atividade, como explicado acima.

Os utilizadores também têm armazenado um conjunto dos seus planos de treino e um das suas atividades fora de planos de treino, o que nos permite calcular, por exemplo, todos os kilómetros que já percorreu nas suas atividades, entre outras estatísticas relevantes.

Planos de treino

A classe PlanoTreino contém os métodos e atributos relacionados aos planos de treino da aplicação. Contém um conjunto de atividades que o utilizador deverá realizar e iterações de cada, caso seja necessário.

Estatísticas e recordes

Um dos objetivos desta aplicação é, tendo registos de utilizadores, atividades e planos de treino, poder consultar os recordes de cada atividade que consideramos, por exemplo, selecionando Corrida, temos acesso ao maior número quilómetros percorridos numa corrida ou a maior velocidade média, dados estes que são atualizados de acordo com a data que consideramos como atual na simulação, que pode ser alterada na opção ‘Avançar tempo’.

Também dependente da data atual da simulação temos as estatísticas, como consultar o plano de treino mais exigente em termos de calorias gastas, ou o total de quilómetros percorridos por um dado utilizador em todas as atividades que realizou.

MVC

Nesta aplicação implementamos a interação com o utilizador com o padrão model/view/controller da seguinte forma:
A classe ProgPrincipal tem apenas o método main e uma instância de AppDesportiva, o controller.

A AppDesportiva implementa o controller, que processa as opções escolhidas pelo utilizador enviando os pedidos correspondentes para o model/facade GestorDesportivo. A AppDesportiva também tem várias instâncias de Menu, a view, que mostra e lê as opções do utilizador. A classe AppDesportiva também imprime informação para o ecrã, podendo assim também ser considerada parte da view.

Utilização do programa

Podemos inicializar o programa de várias maneiras. Se o método main receber como argumento um ficheiro com um estado que possa ser lido não passa pelo menu inicial, se não receber argumentos ou receber um ficheiro que não possa ser carregado então um menu inicial irá aparecer no terminal. Este dá-nos várias opções:
sair do programa, carregar um estado de aplicação já existente ou criar um novo estado.

Após escolhermos a nossa opção e darmos a informação necessária para prosseguir vamos para outro menu. Aqui podemos adicionar utilizadores, atividades ou planos de treino, registar a execução de uma atividade ou plano de treino, ou visualizar um destes objetos em específico. Neste menu ainda podemos sair da aplicação e/ou guardar o estado atual. Por fim, podemos fazer uma simulação das execuções das atividades. Se o decidirmos fazer, vamos para outro menu, e vai ser pedida uma data para a simulação. Neste próximo menu podemos ver as estatísticas, recordes, avançar no tempo ou ver toda a informação junta.

Conclusão

Concluindo, conseguimos fazer uma aplicação de gestão de atividades físicas que funcionasse corretamente e consideramos que respeita as normas da programação orientada aos objetos, nomeadamente na definição das suas classes e em termos de encapsulamento.

Não conseguimos acabar a implementação da geração de um plano de treino, logo, não temos essa opção na nossa aplicação.
Também consideramos que existem várias partes que poderiam ser melhores em termos de reutilização do código e utilização de métodos mais genéricos. Mas no geral, achamos que conseguimos implementar estes princípios de reutilização em vários pontos, como o acesso a informações de um utilizador de acordo com um determinado critério e o cálculo dos recordes, por exemplo.