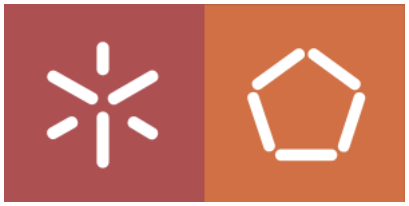


Universidade do Minho
Escola de Engenharia
Licenciatura em Engenharia Informática

Unidade Curricular de Programação Orientada a Objetos

Ano Letivo de 2023/2024

Activity Planner

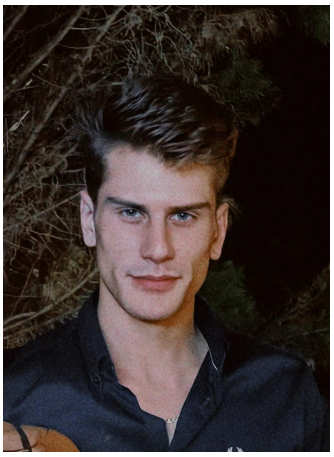
Flávia Alexandra Silva Araújo (A96587)
Miguel Torres Carvalho (A95485)

9 de maio de 2024

POO

Equipa de Trabalho:

Flávia Alexandra Silva Araújo (A96587)



Miguel Torres Carvalho (A95485)

Resumo

No âmbito da Unidade Curricular Programação Orientada a Objetos, foi-nos proposto o desenvolvimento de uma aplicação de gestão de atividades físicas, à qual desginamos de *Activity Planner*.

A aplicação desenvolvida permite a gestão de utilizadores, atividades, planos de treino, simulação de atividades e visualização de estatísticas. A aplicação foi desenvolvida em *Java*, utilizando o paradigma de programação orientada a objetos aprendido nas aulas.

Neste relatório, é apresentada a arquitetura de classes da aplicação, bem como as funcionalidades implementadas nesta e a forma como as mesmas foram desenvolvidas.

Índice

Lista de Figuras

1 Arquitetura de Classes

1.1 Diagrama de Classes

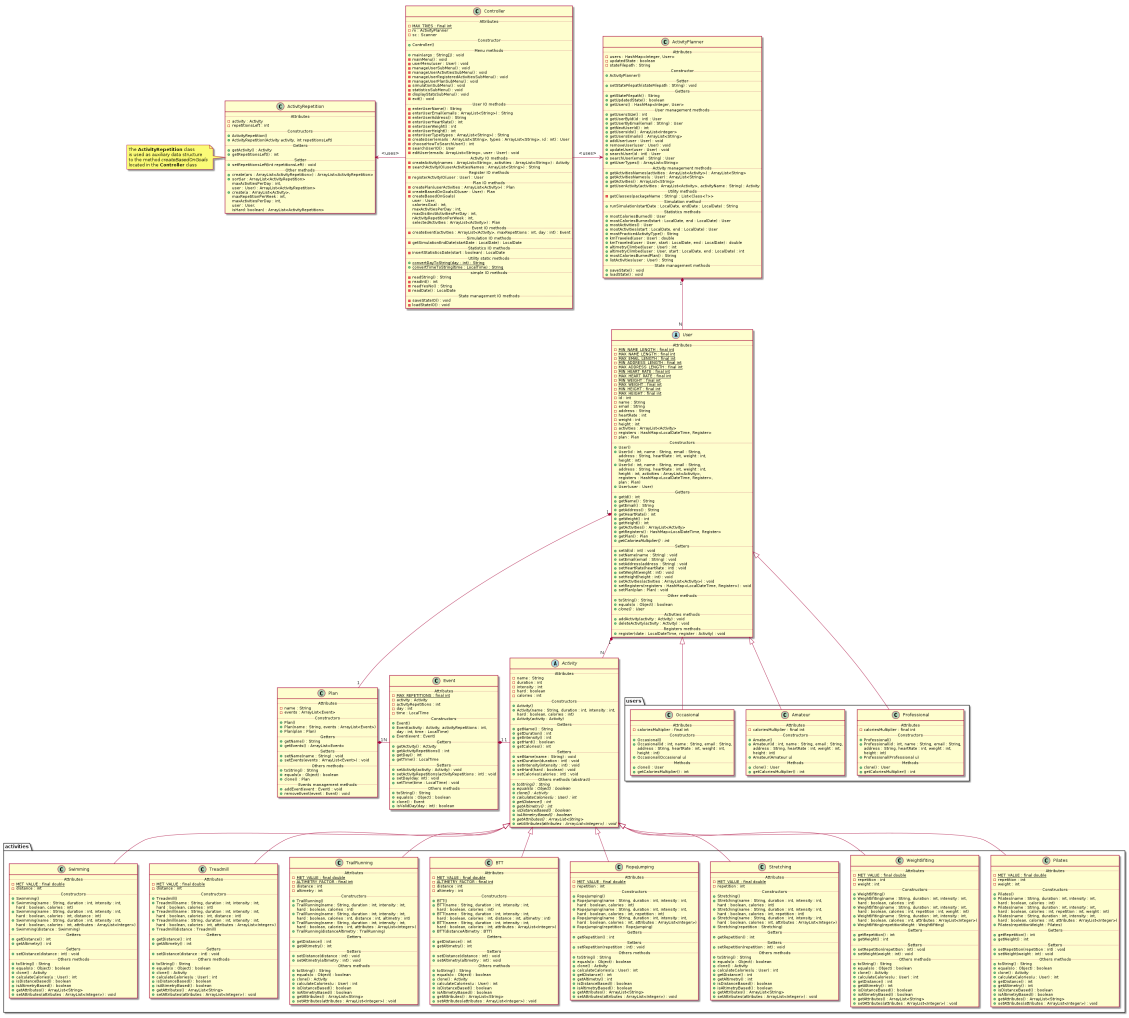


Figura 1.1: Diagrama de Classes

1.2 Classe *ActivityPlanner*

A classe *ActivityPlanner* funciona como o modelo e *facade* da aplicação, sendo responsável por guardar e carregar o estado da aplicação, bem como por fornecer métodos para a manipulação e acesso deste estado.

Por conseguinte, a *ActivityPlanner* é constituída pelos seguintes atributos:

- ***users : HashMap<Integer, User>*** - Lista de utilizadores carregados;
- ***updatedState : boolean*** - Indica se o estado da aplicação foi alterado;
- ***defaultStateFilepath : String*** - Caminho para o ficheiro binário que contém o estado da aplicação.

Providencia métodos para a gestão de utilizadores, acesso ao nome de atividades, simulação de atividades e cálculo de estatísticas, assim como para a salvaguarda do estado da aplicação e métodos que retornam a lista das subclasses de *User* e *Activity*, utilizados para a listagem de utilizadores e atividades oferecidas pela aplicação, ao criar um novo utilizador ou atividade iterativamente, na classe *Controller*.

1.3 Classe *Controller*

A classe *Controller* é a classe responsável pela execução do programa - faz o *parsing* dos argumentos da linha de comandos, carrega e salva o estado da aplicação através da classe *ActivityPlanner*, e executa o menu principal ou da perspetiva de um utilizador, dependendo dos argumentos passados.

É nesta classe que ocorrem as interações com o utilizador, através de menus interativos, e a execução das funcionalidades da aplicação.

Foram desenvolvidos os seguintes menus interativos, com os respetivos submenus e opções:

- Menu principal da aplicação:
  - Adicionar, editar, remover e visualizar utilizadores;
  - Adicionar, remover e visualizar atividades de interesse;
  - Registar e visualizar atividades completas;
  - Adicionar, remover e visualizar planos de treino;
  - Simular atividades;
  - Visualizar estatísticas;
  - Carregar e guardar o estado da aplicação;
  - Sair da aplicação.
- Menu com a perspetiva de um utilizador:
  - Editar o perfil do utilizador;
  - Adicionar e remover atividades de interesse;
  - Registar e visualizar atividades completas;
  - Adicionar, remover e visualizar planos de treino;
  - Visualizar estatísticas;
  - Carregar e guardar o estado da aplicação;
  - Sair da aplicação.

A classe *Controller* funciona como uma *interface* entre o utilizador e a aplicação, sendo responsável por chamar os métodos da classe *ActivityPlanner* e por apresentar os resultados.

No sistema MVC esta classe representa o *controller* assim como o *view*, uma vez que é responsável por apresentar os menus interativos, fazendo as respetivas execuções através dos métodos da classe *ActivityPlanner*.

1.4 Classe *User*

A superclasse *User* representa um utilizador da aplicação, sendo constituída pelos seguintes atributos:

- ***id : int*** - Identificador único do utilizador;
- ***name : String*** - Nome completo do utilizador;
- ***email : String*** - Endereço de correio eletrónico do utilizador, sendo este único em relação a todos os utilizadores da aplicação;
- ***address : String*** - Morada do utilizador;
- ***heartRate : int*** - Frequência cardíaca em repouso do utilizador, em batimentos por minuto;
- ***weight : int*** - Peso do utilizador, em kilogramas;
- ***height : int*** - Altura do utilizador, em centímetros;
- ***activities : ArrayList<Activity>*** - Conjunto de atividades de interesse de um utilizador, utlizada para a simplificação de um registo da atividade praticada ou da criação de um plano de treino;
- ***registers : HashMap<LocalDateTime, Activity>*** - Conjunto de actividades praticadas por um utilizador, com a respetiva data e hora de prática, atividade praticada e consumo calórico;
- ***plan : Plan*** - Plano de treino semanal de um utilizador.

Os atributos *weight*, *height* e *heartRate* são utilizados para o cálculo do consumo calórico de uma atividade, sendo estes parâmetros mutáveis, uma vez que podem ser alterados ao longo do tempo.

Para além dos vários construtores, dos métodos tradicionais de acesso e modificação dos atributos (*getters* e *setters*), métodos para a cópia profunda dos objetos desta classe, conversão para *String* e igualdade entre objetos desta classe, foram implementados métodos para a adição/remoção de uma atividade de interesse, bem como métodos para o registo de uma atividade por parte de um utlizador.

A classe *User* foi desenvolvida de forma a permitir a extensão desta, através da implementação de novos tipos de utilizadores. Assim, foi definida uma hierarquia de classes, onde a classe *User* é a superclasse e as suas subclasses são *Occasional*, *Amateur* e *Professional*.

A aplicação foi desenvolvida de forma a permitir a adição de novos tipos de utilizadores, através da criação de uma nova subclasse de *User* com as características específicas deste tipo de utilizador, sem que fosse necessário alterar a classe *User* e outras componentes da aplicação, garantindo, deste modo, a modularidade, manutenção e extensabilidade da superclasse *User* e das suas subclasses.

1.5 Classe *Activity*

A superclasse *Activity* representa uma atividade física, sendo constituída pelos seguintes atributos:

- ***name : String*** - Nome da atividade;
- ***duration : int*** - Duração da atividade, em minutos;
- ***intensity : int*** - Intensidade da atividade (um valor de 1 a 100 para representar um valor percentual);
- ***hard : boolean*** - Indica, através de um valor booleano, se a atividade é considerada *hard* pelo utilizador;
- ***calories : int*** - Número de calorias queimadas durante a atividade. O registo deste atributo deve-se à mutação dos parâmetros do utilizador para o cálculo do consumo calórico, desde a frequência cardíaca, o peso, bem como a altura deste. Assim, este atributo foi utilizado de forma a garantir a salvaguarda do valor de calorias queimadas num registo de uma atividade.

A classe *Activity*, assim como a classe *User*, foi desenvolvida de forma a permitir a extensão desta, através da implementação de novos tipos de atividades, sem que fosse necessário alterar a classe *Activity* e outras componentes da aplicação.

As subclasses de *Activity* contêm atributos específicos que caracterizam a atividade em questão, como por exemple, a distância percorrida, o número de repetições, o peso levantado, entre outros.

Foram implementadas as seguintes subclasses de *Activity*:

- ***Swimming*** - Atividade de natação, compostas pelos atributos *distance : int*;
- ***Treadmill*** - Atividade de corrida em passadeira, compostas pelos atributos *distance : int*;
- ***TrailRunning*** - Atividade de corrida em trilhos, compostas pelos atributos *distance : int* e *altimetry : int*;
- ***BTT*** - Atividade de BTT, compostas pelos atributos *distance : int* e *altimetry : int*;
- ***RopeJumping*** - Atividade de saltar à corda, compostas pelos atributos *repetitions : int*;
- ***Stretching*** - Atividade de alongamentos, compostas pelos atributos *repetitions : int*;
- ***Weightlifting*** - Atividade de levantamento de pesos, compostas pelos atributos *repe-*

titions : int e *weight : int*;

- ***Pilates*** - Atividade de pilates, compostas pelos atributos *repetitions : int* e *weight : int*;

Para cada uma destas subclasses é definido o atributo MET (*Metabolic Equivalent of Task*) da atividade, que é utilizado para o cálculo do consumo calórico da atividade, este também definido como um método abstrato na superclasse *Activity*.

Nas subclasses de *Activity* foram implementados métodos para obter e definir os atributos específicos da atividade, estes permitem a criação destas atividades de forma iterativa, através da classe *Controller*.

A classe *Activity* foi desenvolvida tirando proveito do conceito de polimorfismo, permitindo a criação de atividades de forma genérica, através da superclasse *Activity*, e a sua manipulação de forma específica, através dos atributos únicos das respetivas subclasses.

1.6 Classe *Event*

A classe *Event* representa um evento correspondente a um plano de treino, sendo constituída pelos seguintes atributos:

- ***activity : Activity*** - Atividade praticada no evento;
- ***activityRepetitions : int*** - Número de vezes que a atividade será praticada;
- ***day : int*** - Dia da semana do evento, onde 1 corresponde a domingo e 7 a sábado;
- ***time : LocalTime*** - Hora do evento.

1.7 Classe *Plan*

A classe *Plan* representa um plano de treino, por definição, semanal, que um utilizador pode criar, vizualizar e remover.

Esta é composta pelos seguintes atributos:

- ***name : String*** - Nome do plano de treino;
- ***events : ArrayList<Event>*** - Lista de eventos que compõem o plano de treino.

As funcionalidades proporcionadas por esta classe, bem como a classe *Event*, serão detalhadas no subcapítulo *Gestão de Planos de Treino*.

1.8 Classe *ActivityRepetition*

A classe *ActivityRepetition* é utlizada como um objeto auxiliar para a criação de um plano de treino baseado nos objetivos de um utilizador.

O processo de geração de planos de treino baseados em objetivos será aprofundado no subcapítulo *Gestão de Planos de Treino*, a qual será detalhada a forma como esta classe é utilizada.

1.9 Package *exceptions*

Foi também criado um *package exceptions*, onde foram definidas as seguintes exceções:

- ***ActivityIsRegisteredException*** - Exceção lançada quando uma atividade com o mesmo nome já foi registada;
- ***ActivityNotFoundException*** - Exceção lançada quando uma atividade não é encontrada;
- ***UserNotFoundException*** - Exceção lançada quando um utilizador não é encontrado;
- ***StateNotSavedException*** - Exceção lançada quando o estado da aplicação não pode ser guardado;
- ***StateNotLoadedException*** - Exceção lançada quando o estado da aplicação não pode ser carregado;

Estas exceções foram criadas de forma a garantir a robustez da aplicação, permitindo a deteção de situações de erro e a sua correta gestão.

2 Descrição de Funcionalidades da Aplicação

2.1 Gestão de Utilizadores

2.1.1 Adicionar Utilizador

TODO

2.1.2 Editar Utilizador

TODO

2.1.3 Remover Utilizador

TODO

2.1.4 Visualizar Utilizadores

TODO

2.2 Gestão de Atividades

2.2.1 Adicionar Atividade

TODO

2.2.2 Remover Atividade

TODO

2.2.3 Visualizar Atividades

TODO

2.3 Registo e Visualização de Atividades Completas

2.3.1 Registar Atividade

TODO

2.3.2 Visualizar Registos de Atividades

TODO

2.4 Gestão de Planos de Treino

2.4.1 Adicionar Plano de Treino Interativamente

TODO

2.4.2 Adicionar Plano de Treino Baseado em Objetivos

TODO

2.4.3 Remover Plano de Treino

TODO

2.4.4 Visualizar Planos de Treino

TODO

2.5 Simulação

TODO

2.6 Estatísticas

TODO

2.7 Salvaguarda do Estado da Aplicação

Para garantir que o estado da aplicação é preservado entre execuções, esta permite guardar e carregar o estado atual através de um ficheiro binário. As opcões de guardar e carregar o estado do programa estão disponíveis no menu principal da aplicação. Adicionalmente o ficheiro binário pode ser carregado diretamente no início da execução do programa através da passagem da localização deste na linha de comandos. Este ficheiro, por definição, é guardado na diretoria *data* e tem o nome *state.ser*, havendo a opção de carregar diferentes estados através da funcionalidade da linha de comandos supramencionada.

Para a implementação desta funcionalidade foram definidos dois métodos na classe *Main*:

- *saveState* - Método que guarda o estado atual da aplicação num ficheiro binário, passado como argumento. Este método deteta se alguma mudança foi feita no estado do programa antes de a guardar, de forma evitar salvar o mesmo estado.
- *loadState* - Método que carrega o estado da aplicação a partir de um ficheiro binário, passado como argumento.

Como os objetos da classe *User* contêm referências para todos os objetos relevantes de serem guardados/carregados - lista de Atividades, conjunto de registos de atividades, Plano de treino semanal com os respetivos Eventos - foi necessário garantir que estes e a própria classe referente ao Utilizador implementassem a interface *Serializable*, de forma a que fossem possíveis de ser guardados, e futuramente carregados, num ficheiro binário.

A aplicação também dispõe de uma capacidade inteligente de detetar mudanças no seu estado, através do atributo booleano *updatedState* na classe *Main*, o que permitiu a implementação das seguintes funcionalidades:

- Notificar o utilizador de que o estado atual não foi guardado, caso este tente sair da aplicação, dando a opção de o guardar, caso o utilizador o deseje fazer.
- Notificar o utilizador que, ao carregar um novo estado, o estado atual será perdido, se houver alterações, dando a opção de retornar atrás se este não quiser perder o estado atual.

O valor do atributo *updatedState* é incializado a *false* no ínicio da execução do programa e é alterado para *false* sempre que o estado da aplicação é guardado, no método *saveState*, ou carregado, no método *loadState*, referidos anteriormente.

Este valor booleano é alterado para *true* sempre que o estado da aplicação é alterado, seja através da adição, edição ou remoção de um utilizador, de uma atualização de uma atividade de um utilizador, do registo de uma nova atividade ou da criação/remoção de um plano de treino para um utlizador em específico.

2.8 Argumentos de Linha de Comandos

3 Conclusão

Maybe