Universidade do Minho
Escola de Engenharia
Licenciatura em Engenharia Informática

Unidade Curricular de Programação Orientada a Objetos

Ano Letivo de 2023/2024

Activity Planner

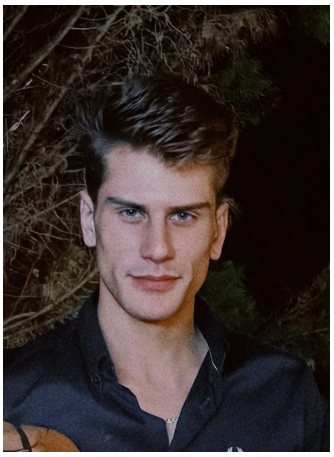
Flávia Alexandra Silva Araújo (A96587)
Miguel Torres Carvalho (A95485)

26 de abril 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 chamámos *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

1 Arquitetura de Classes

1.1 Diagrama de Classes

Descrever o diagrama de classes da aplicação, apresentando as classes e as suas relações.

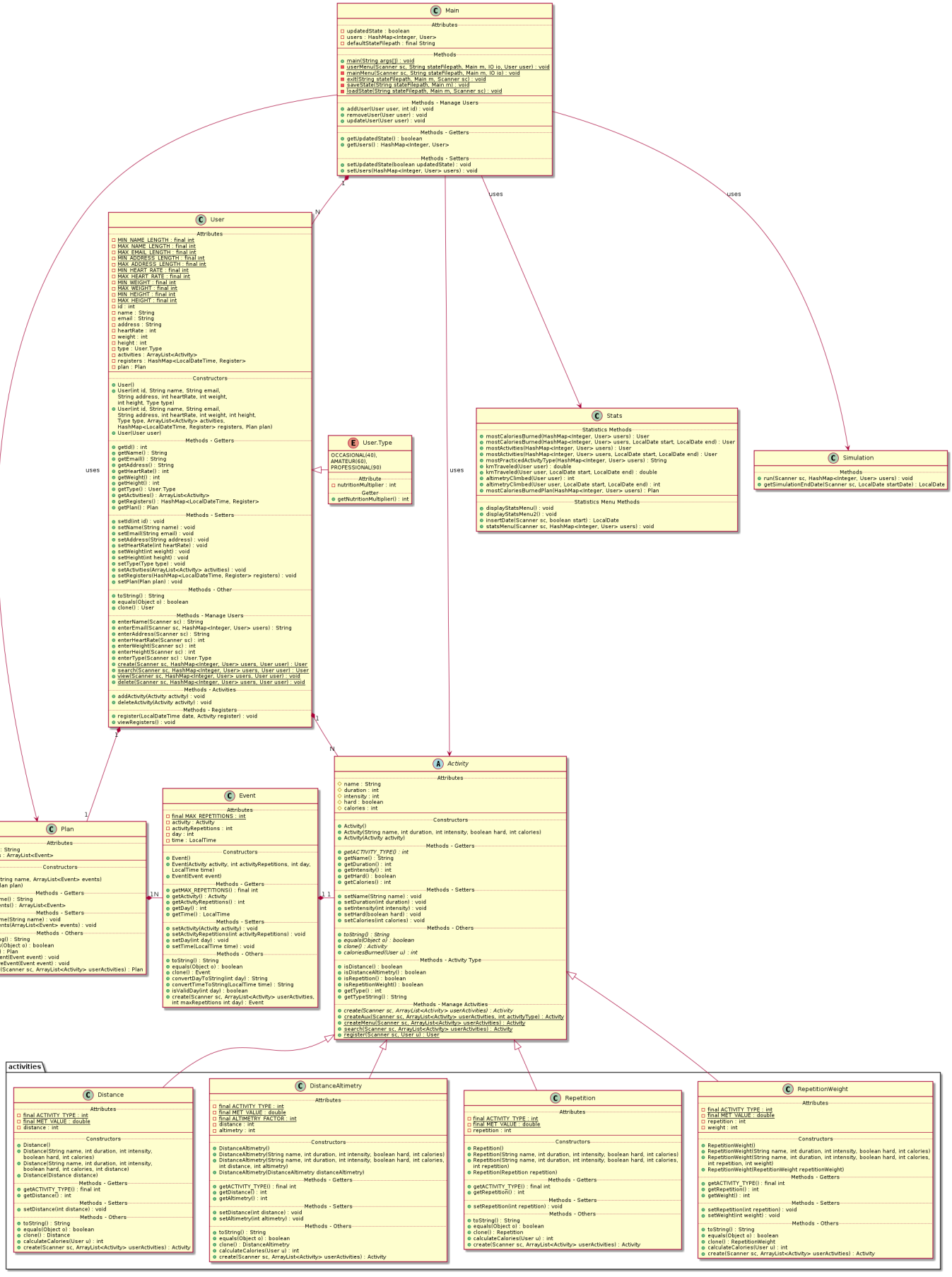


Figura 1.1: Diagrama de Classes

1.2 Classe *Main*

A classe *Main* é a classe principal da aplicação, sendo responsável pela execução desta - faz o *parsing* dos argumentos da linha de comandos, carrega o estado da aplicação, caso este tenha sido guardado anteriormente, e executa a aplicação. Por conseguinte, a *Main* é constituída pelos seguintes atributos:

- ***updatedState : boolean*** - Indica se o estado da aplicação foi alterado desde a última vez que foi guardado;
- ***users : HashMap<Integer, User>*** - Lista de utilizadores carregados;
- ***defaultStateFilepath : String*** - Caminho para o ficheiro binário que contém o estado da aplicação, caso este tenha sido omitido na linha de comandos.

Para além de alguns métodos tradicionais de acesso e modificação dos atributos (*getters* e *setters*) a nível da atualização do estado da aplicação, esta classe contém diversos métodos para a execução desta, nomeadamente:

- Menu para o utilizador, caso ele tenha entrado com a sua conta;
- Menu principal da aplicação;
- Adicionar e remover utilizadores;
- Guardar e carregar o estado da aplicação;
- Sair da aplicação.

Para além destes métodos, a classe *Main* contém o método *main*, que é o ponto de entrada da aplicação. As funcionalidades proporcionadas por esta classe serão detalhadas no subcapítulo *Salvaguarda do Estado da Aplicação*.

1.3 Classe *User*

A classe *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 na 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;
- ***type : Type*** - Tipo de utilizador, podendo ser ocasional, amador ou professional.
- ***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.

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 gestão de utilizadores (criação, vizualização, edição e remoção), bem como métodos para a adição/remoção de uma atividade de interesse e métodos para o registo de uma atividade e vizualição destes registos de um utlizador.

As funcionalidades proporcionadas por esta classe serão detalhadas nos subcapítulos *Gestão de Utilizadores* e *Registo e Visualização de Atividades Completas*.

1.4 Classe *Activity*

A classe abstrata *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 a percentagem da intensidade desta);
- ***hard : boolean*** - Indica, através de um valor booleano, se a atividade é considerada ou não *hard*.
- ***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.

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 os seguintes métodos:

- Obtenção do tipo da atividade (*Distance*, *DistanceAltimetry*, *Repetition* ou *RepetitionWeight*);
- Criação de uma atividade;
- Procurar uma atividade de um utilizador através do nome atribuído a esta;
- Registar uma atividade praticada por um utilizador;

As funcionalidades proporcionadas por esta classe serão detalhadas no subcapítulo *Gestão de Atividades*.

1.5 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.

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, foi implementado um método para a criação interativa de um evento.

1.6 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.

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 de um evento a um plano de treino, para a criação interativa de um plano e para a criação de um plano baseado nos objetivos de um utlizador.

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

1.7 Classe *Stats*

A classe *Stats* foi criada com o intuito de proporcionar métodos para o cálculo de estatísticas sobre o estado da aplicação. Esta é unicamente constituída por métodos - um para o menu de estatisticas, e os restantes para o cálculo destas.

As estatísticas e respetivas formas de cálculo, determinadas nesta classe, serão detalhadas no subcapítulo *Estatísticas*.

1.8 Classe *Simulation*

A classe *Simulation* contém um método para a simulação da prática de atividades por parte de todos os utilizadores da aplicação baseado nos seus planos de treino.

O método de simulação definido nesta classe será aprofundado no subcapítulo *Simulação*.

1.9 Classe *IO*

No âmbito de agilizar a coleta de *input's* por parte de um utilizador, foi criada a classe *IO*, com os seguintes métodos:

- ***readString(sc : Scanner) : String*** - Método que lê uma *String* introduzida pelo utilizador;
- ***readInt(sc : Scanner) : Int*** - Método que lê um inteiro introduzido pelo utilizador;

- ***readYesNo(sc : Scanner) : String*** - Método que lê um caracter introduzido pelo utilizador, que deverá ser 'y' ou 'n', independentemente deste ser maiúsculo ou minúsculo.

Nestes métodos, é feita a validação do *input* através da verificação das exceções lançadas pela classe *Scanner*.

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 Conclusões e Trabalho Futuro

Maybe