



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)

25 de abril de 2024

P00

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.

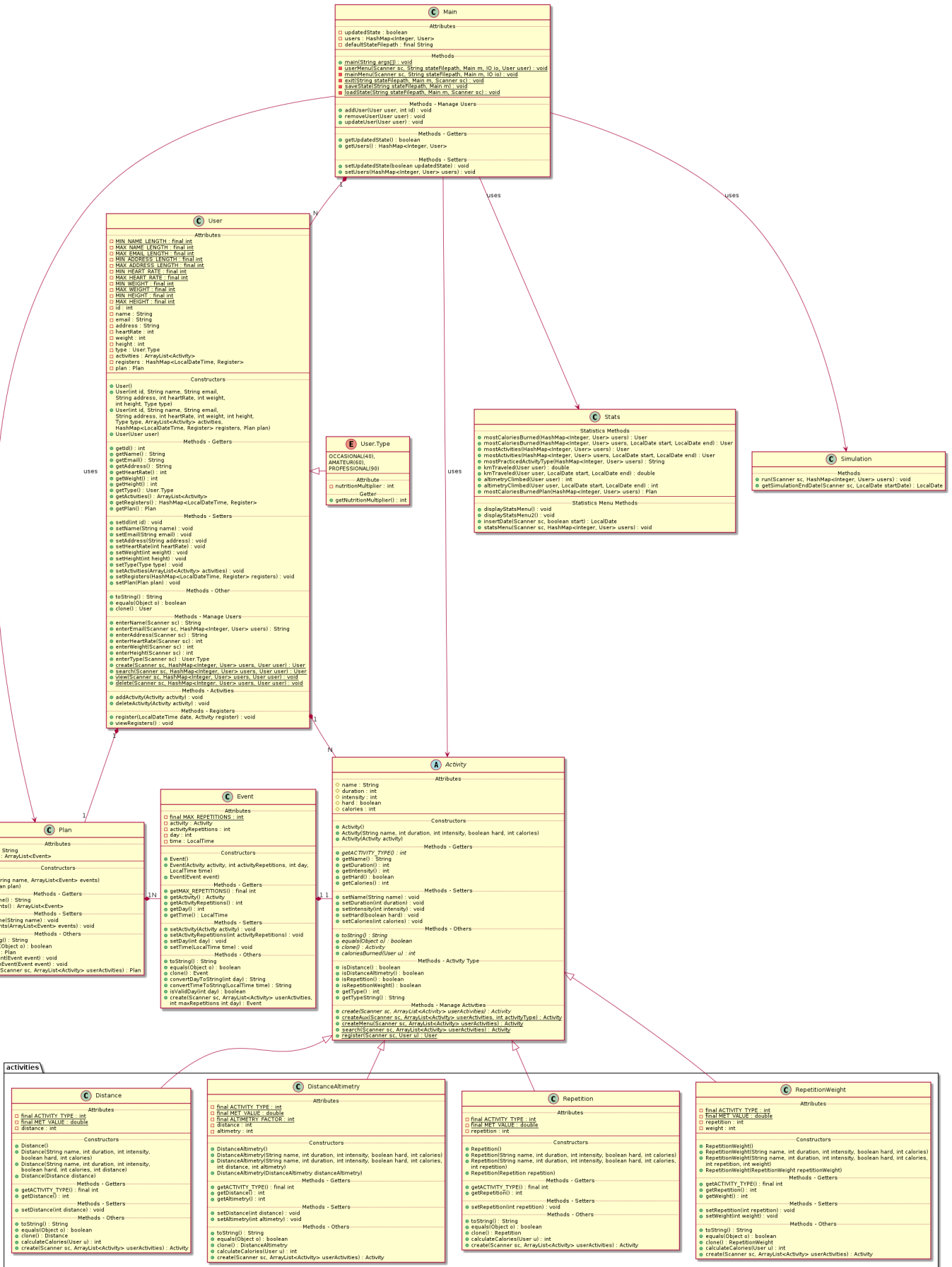
Índice

1	Arquitetura de Classes	1
1.1	Diagrama de Classes	1
1.2	Classe <i>Main</i>	3
1.3	Classe <i>User</i>	3
1.4	Classe <i>Activity</i>	3
1.5	Classe <i>Event</i>	3
1.6	Classe <i>Plan</i>	3
1.7	Classe <i>Stats</i>	4
1.8	Classe <i>Simulation</i>	4
1.9	Classe <i>IO</i>	4
2	Descrição de Funcionalidades da Aplicação	5
2.1	Gestão de Utilizadores	6
2.1.1	Adicionar Utilizador	6
2.1.2	Editar Utilizador	6
2.1.3	Remover Utilizador	6
2.1.4	Visualizar Utilizadores	6
2.2	Gestão de Atividades	6
2.2.1	Adicionar Atividade	6
2.2.2	Remover Atividade	6
2.2.3	Visualizar Atividades	6
2.3	Registo e Visualização de Atividades Completas	6
2.3.1	Registar Atividade	6
2.3.2	Visualizar Registos de Atividades	6
2.4	Gestão de Planos de Treino	6
2.4.1	Adicionar Plano de Treino Interativamente	6
2.4.2	Adicionar Plano de Treino Baseado em Objetivos	6
2.4.3	Remover Plano de Treino	6
2.4.4	Visualizar Planos de Treino	6
2.5	Simulação	6
2.6	Estatísticas	6
2.7	Salvaguarda do Estado da Aplicação	7
2.8	Argumentos de Linha de Comandos	8
3	Conclusões e Trabalho Futuro	9

1 Arquitetura de Classes

1.1 Diagrama de Classes

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



1.2 Classe *Main*

1.3 Classe *User*

1.4 Classe *Activity*

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.

Na classe *Event* são guardadas as informações relativas a um evento em um plano de treino - como a atividade a realizar, a quantidade de repetições, o dia da semana, representado por um inteiro, onde 1 corresponde a domingo, gradualmente, até 7, que corresponde a sábado, e a respetiva hora em que a atividade será realizada.

A aplicação permite ao utilizador criar um evento da seguinte maneira: O utilizador escolhe uma atividade da lista de atividades que este introduziu previamente na sua conta, e o número de repetições que fará (ou fez) desta, e escolhe o dia da semana (de domingo a sábado), bem como a hora a que pretende realizar a atividade (ou quando a realizou). O evento é então criado.

Estes Eventos são utilizados para criar um Plano de Treino Interativo pelo utilizador. [CONTINUAR]

1.6 Classe *Plan*

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

As funcionalidades proporcionadas por esta classe serão detalhadas no capítulo *Gestão de Planos de Treino*.

1.7 Classe *Stats*

1.8 Classe *Simulation*

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

2.1.2 Editar Utilizador

2.1.3 Remover Utilizador

2.1.4 Visualizar Utilizadores

2.2 Gestão de Atividades

2.2.1 Adicionar Atividade

2.2.2 Remover Atividade

2.2.3 Visualizar Atividades

2.3 Registo e Visualização de Atividades Completas

2.3.1 Registrar Atividade

2.3.2 Visualizar Registos de Atividades

2.4 Gestão de Planos de Treino

2.4.1 Adicionar Plano de Treino Interativamente

2.4.2 Adicionar Plano de Treino Baseado em Objetivos

2.4.3 Remover Plano de Treino

2.4.4 Visualizar Planos de Treino

2.5 Simulação

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 opções 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 inicializado a *false* no início 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 utilizador em específico.

2.8 Argumentos de Linha de Comandos

3 Conclusões e Trabalho Futuro

Maybe