



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)

22 de abril de 2024

POO

Equipa de Trabalho:



Flávia Alexandra Silva Araújo (A96587)



Miguel Torres Carvalho (A95485)

Resumo

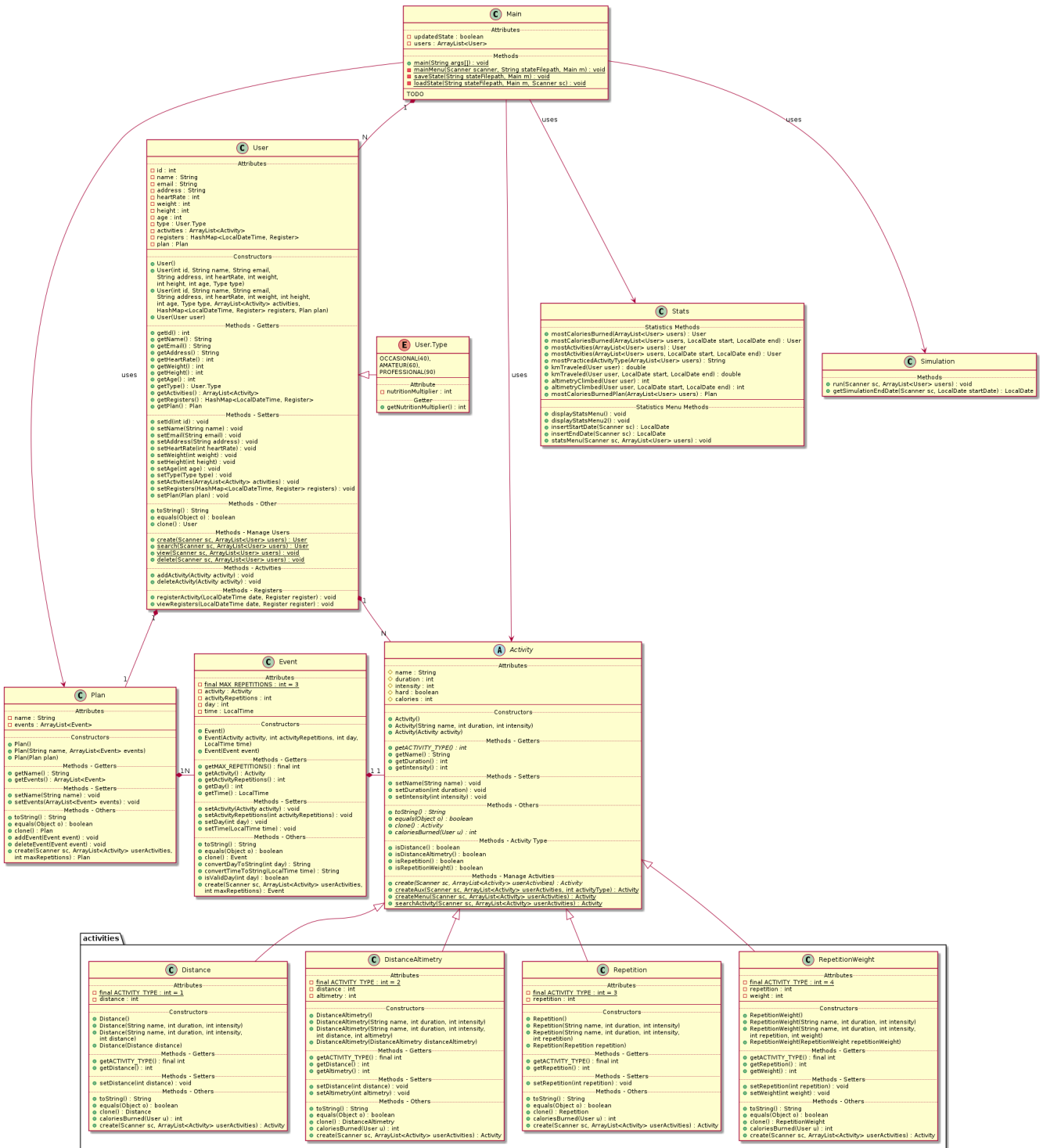
No âmbito da Unidade Curricular Programação Orientada a Objetos, foi-nos proposto o desenvolvimento de uma aplicação de gestão de atividades físicas, à qual chamámos *Activity Planner*. A aplicação desenvolvida permite a gestão de utilizadores, atividades, planos de treino, simulação de atividades e 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
1.1	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>Plan</i>	3
1.6	Classe <i>Event</i>	3
1.7	Classe <i>Stats</i>	3
1.8	Classe <i>Simulation</i>	3
2	Descrição de Funcionalidades da Aplicação	4
2.1	Gestão de Utilizadores	5
2.1.1	Adicionar Utilizador	5
2.1.2	Editar Utilizador	5
2.1.3	Remover Utilizador	5
2.1.4	Visualizar Utilizadores	5
2.2	Gestão de Atividades	5
2.2.1	Adicionar Atividade	5
2.2.2	Remover Atividade	5
2.2.3	Visualizar Atividades	5
2.3	Registo e Visualização de Atividades Completas	5
2.3.1	Registar Atividade	5
2.3.2	Visualizar Registos de Atividades	5
2.4	Gestão de Planos de Treino	5
2.4.1	Adicionar Plano de Treino Interativamente	5
2.4.2	Adicionar Plano de Treino Baseado em Objetivos	5
2.4.3	Remover Plano de Treino	5
2.4.4	Visualizar Planos de Treino	5
2.5	Simulação	5
2.6	Estatísticas	5
2.7	Salvaguarda do Estado da Aplicação	6
2.8	Argumentos de Linha de Comandos	7
3	Conclusões e Trabalho Futuro	8

1 Arquitetura de Classes

1.1 Diagrama de Classes



1.2 Classe *Main*

1.3 Classe *User*

1.4 Classe *Activity*

1.5 Classe *Plan*

1.6 Classe *Event*

1.7 Classe *Stats*

1.8 Classe *Simulation*

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 *true* 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