Universidade do Minho
Escola de Engenharia
Licenciatura em Engenharia Informática

Unidade Curricular de Programação Orientada a Objetos

Ano Letivo de 2023/2024

Activity Planner

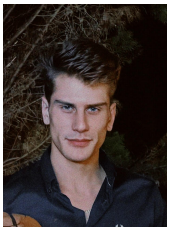
Flávia Alexandra Silva Araújo (A96587)
Miguel Torres Carvalho (A95485)

19 de abril de 2024

POO

Equipa de Trabalho:

Flávia Alexandra Silva Araújo (A96587)

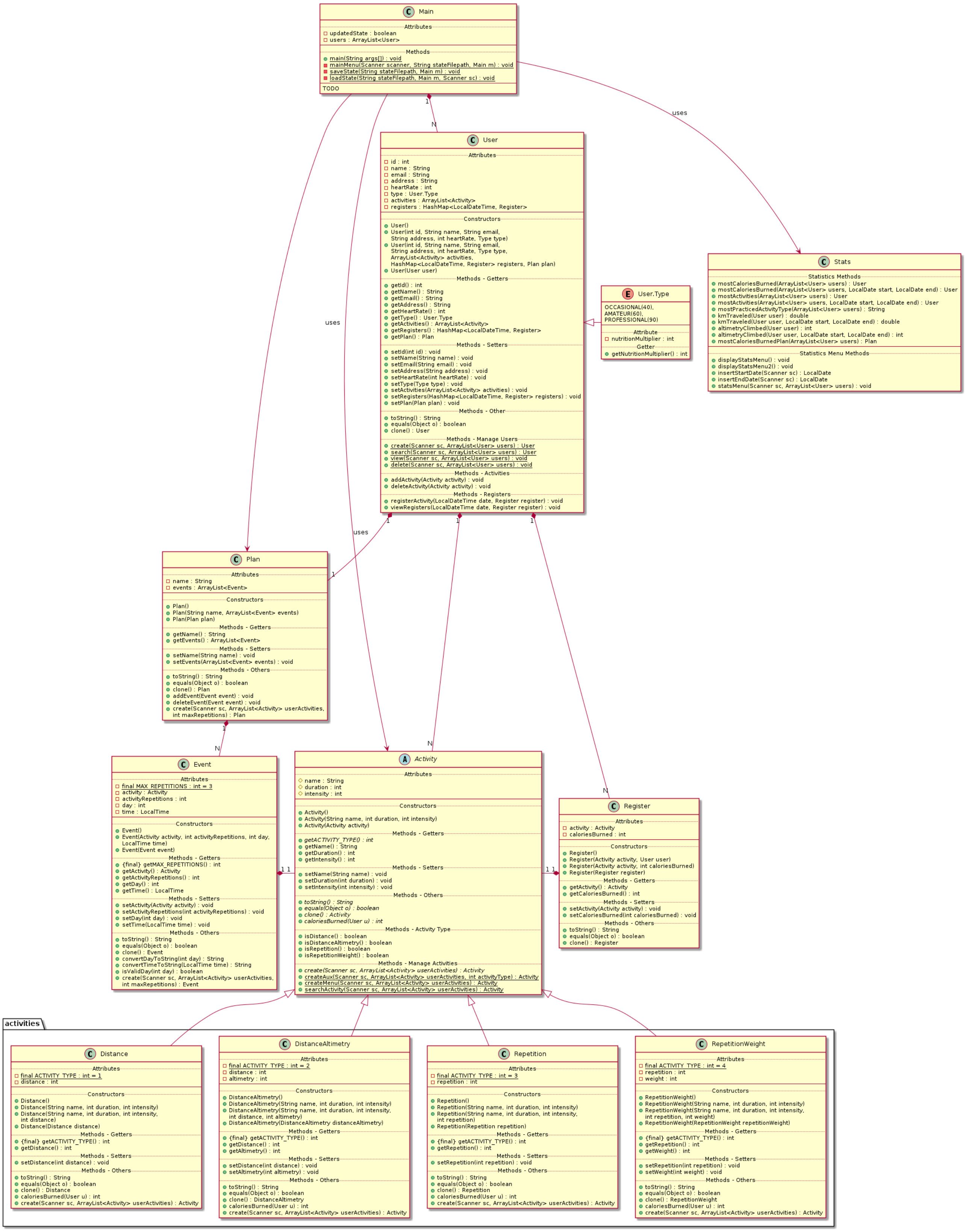


Miguel Torres Carvalho (A95485)

Índice

1 Arquitetura de Classes

1.1 Diagrama de Classes


Main
Attributes
updatedState : boolean
users : ArrayList<User>
Methods
main(String args[]) : void
mainMenu(Scanner scanner, String stateFilepath, Main m) : void
saveState(String stateFilepath, Main m) : void
loadState(String stateFilepath, Main m, Scanner sc) : void
TODO
uses
uses
uses
1
N
User
Attributes
id : int
name : String
email : String
address : String
heartRate : int
type : User.Type
activities : ArrayList<Activity>
registers : HashMap<LocalDateTime, Register>
Constructors
User()
User(int id, String name, String email,
String address, int heartRate, Type type)
User(int id, String name, String email,
String address, int heartRate, Type type,
ArrayList<Activity> activities,
HashMap<LocalDateTime, Register> registers, Plan plan)
User(User user)
Methods - Getters
getId() : int
getName() : String
getEmail() : String
getAddress() : String
getHeartRate() : int
getType() : User.Type
getActivities() : ArrayList<Activity>
getRegisters() : HashMap<LocalDateTime, Register>
getPlan() : Plan
Methods - Setters
setId(int id) : void
setName(String name) : void
setEmail(String email) : void
setAddress(String address) : void
setHeartRate(int heartRate) : void
setType(Type type) : void
setActivities(ArrayList<Activity> activities) : void
setRegisters(HashMap<LocalDateTime, Register> registers) : void
setPlan(Plan plan) : void
Methods - Other
toString() : String
equals(Object o) : boolean
clone() : User
Methods - Manage Users
create(Scanner sc, ArrayList<User> users) : User
search(Scanner sc, ArrayList<User> users) : User
view(Scanner sc, ArrayList<User> users) : void
delete(Scanner sc, ArrayList<User> users) : void
Methods - Activities
addActivity(Activity activity) : void
deleteActivity(Activity activity) : void
Methods - Registers
registerActivity(LocalDateTime date, Register register) : void
viewRegisters(LocalDateTime date, Register register) : void
User.Type
OCCASIONAL(40),
AMATEUR(60),
PROFESSIONAL(90)
Attribute
nutritionMultiplier : int
Getter
getNutritionMultiplier() : int
Stats
Statistics Methods
mostCaloriesBurned(ArrayList<User> users) : User
mostCaloriesBurned(ArrayList<User> users, LocalDate start, LocalDate end) : User
mostActivities(ArrayList<User> users) : User
mostActivities(ArrayList<User> users, LocalDate start, LocalDate end) : User
mostPracticedActivityType(ArrayList<User> users) : String
kmTraveled(User user) : double
kmTraveled(User user, LocalDate start, LocalDate end) : double
altimetryClimbed(User user) : int
altimetryClimbed(User user, LocalDate start, LocalDate end) : int
mostCaloriesBurnedPlan(ArrayList<User> users) : Plan
Statistics Menu Methods
displayStatsMenu() : void
displayStatsMenu2() : void
insertStartDate(Scanner sc) : LocalDate
insertEndDate(Scanner sc) : LocalDate
statsMenu(Scanner sc, ArrayList<User> users) : void
Plan
Attributes
name : String
events : ArrayList<Event>
Constructors
Plan()
Plan(String name, ArrayList<Event> events)
Plan(Plan plan)
Methods - Getters
getName() : String
getEvents() : ArrayList<Event>
Methods - Setters
setName(String name) : void
setEvents(ArrayList<Event> events) : void
Methods - Others
toString() : String
equals(Object o) : boolean
clone() : Plan
addEvent(Event event) : void
deleteEvent(Event event) : void
create(Scanner sc, ArrayList<Activity> userActivities,
int maxRepetitions) : Plan
Event
Attributes
final MAX_REPETITIONS : int = 3
activity : Activity
activityRepetitions : int
day : int
time : LocalTime
Constructors
Event()
Event(Activity activity, int activityRepetitions, int day,
LocalTime time)
Event(Event event)
Methods - Getters
{final} getMAX_REPETITIONS() : int
getActivity() : Activity
getActivityRepetitions() : int
getDay() : int
getTime() : LocalTime
Methods - Setters
setActivity(Activity activity) : void
setActivityRepetitions(int activityRepetitions) : void
setDay(int day) : void
setTime(LocalTime time) : void
Methods - Others
toString() : String
equals(Object o) : boolean
clone() : Event
convertDayToString(int day) : String
convertTimeToString(LocalTime time) : String
isValidDay(int day) : boolean
create(Scanner sc, ArrayList<Activity> userActivities,
int maxRepetitions) : Event
Activity
Attributes
name : String
duration : int
intensity : int
Constructors
Activity()
Activity(String name, int duration, int intensity)
Activity(Activity activity)
Methods - Getters
getACTIVITY_TYPE() : int
getName() : String
getDuration() : int
getIntensity() : int
Methods - Setters
setName(String name) : void
setDuration(int duration) : void
setIntensity(int intensity) : void
Methods - Others
toString() : String
equals(Object o) : boolean
clone() : Activity
caloriesBurned(User u) : int
Methods - Activity Type
isDistance() : boolean
isDistanceAltimetry() : boolean
isRepetition() : boolean
isRepetitionWeight() : boolean
Methods - Manage Activities
create(Scanner sc, ArrayList<Activity> userActivities) : Activity
createAux(Scanner sc, ArrayList<Activity> userActivities, int activityType) : Activity
createMenu(Scanner sc, ArrayList<Activity> userActivities) : Activity
searchActivity(Scanner sc, ArrayList<Activity> userActivities) : Activity
Register
Attributes
activity : Activity
caloriesBurned : int
Constructors
Register()
Register(Activity activity, User user)
Register(Activity activity, int caloriesBurned)
Register(Register register)
Methods - Getters
getActivity() : Activity
getCaloriesBurned() : int
Methods - Setters
setActivity(Activity activity) : void
setCaloriesBurned(int caloriesBurned) : void
Methods - Others
toString() : String
equals(Object o) : boolean
clone() : Register
activities
Distance
Attributes
final ACTIVITY_TYPE : int = 1
distance : int
Constructors
Distance()
Distance(String name, int duration, int intensity)
Distance(String name, int duration, int intensity,
int distance)
Distance(Distance distance)
Methods - Getters
{final} getACTIVITY_TYPE() : int
getDistance() : int
Methods - Setters
setDistance(int distance) : void
Methods - Others
toString() : String
equals(Object o) : boolean
clone() : Distance
caloriesBurned(User u) : int
create(Scanner sc, ArrayList<Activity> userActivities) : Activity
DistanceAltimetry
Attributes
final ACTIVITY_TYPE : int = 2
distance : int
altimetry : int
Constructors
DistanceAltimetry()
DistanceAltimetry(String name, int duration, int intensity)
DistanceAltimetry(String name, int duration, int intensity,
int distance, int altimetry)
DistanceAltimetry(DistanceAltimetry distanceAltimetry)
Methods - Getters
{final} getACTIVITY_TYPE() : int
getDistance() : int
getAltimetry() : int
Methods - Setters
setDistance(int distance) : void
setAltimetry(int altimetry) : void
Methods - Others
toString() : String
equals(Object o) : boolean
clone() : DistanceAltimetry
caloriesBurned(User u) : int
create(Scanner sc, ArrayList<Activity> userActivities) : Activity
Repetition
Attributes
final ACTIVITY_TYPE : int = 3
repetition : int
Constructors
Repetition()
Repetition(String name, int duration, int intensity)
Repetition(String name, int duration, int intensity,
int repetition)
Repetition(Repetition repetition)
Methods - Getters
{final} getACTIVITY_TYPE() : int
getRepetition() : int
Methods - Setters
setRepetition(int repetition) : void
Methods - Others
toString() : String
equals(Object o) : boolean
clone() : Repetition
caloriesBurned(User u) : int
create(Scanner sc, ArrayList<Activity> userActivities) : Activity
RepetitionWeight
Attributes
final ACTIVITY_TYPE : int = 4
repetition : int
weight : int
Constructors
RepetitionWeight()
RepetitionWeight(String name, int duration, int intensity)
RepetitionWeight(String name, int duration, int intensity,
int repetition, int weight)
RepetitionWeight(RepetitionWeight repetitionWeight)
Methods - Getters
{final} getACTIVITY_TYPE() : int
getRepetition() : int
getWeight() : int
Methods - Setters
setRepetition(int repetition) : void
setWeight(int weight) : void
Methods - Others
toString() : String
equals(Object o) : boolean
clone() : RepetitionWeight
caloriesBurned(User u) : int
create(Scanner sc, ArrayList<Activity> userActivities) : Activity

1.2 Classe *Main*

1.3 Classe *User*

1.4 Classe *Activity*

1.5 Classe *Plan*

1.6 Classe *Register*

1.7 Classe *Event*

1.8 Classe *Stats*

1.9 Classe *Simulation*

2 Descrição de Funcionalidades da Aplicação

2.1 Gestão de Utilizadores

2.2 Gestão de Atividades

2.3 Registo e Visualização de Atividades Completas

2.4 Gestão de Planos de Treino

2.5 Simulação

2.6 Estatísticas

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