Hierarquia

O nosso trabalho tem as seguintes classes

**Main**

Classe principal responsável pelo funcionamento da aplicação, i.e., todo o aspeto e fluxo de funcionamento da aplicação. Comunica diretamente com UMeR que fornece lhe dados essenciais ao funcionamento da aplicação.

**UMeR**

Classe da empresa UMeR. É a base de dados sobre clientes, condutores, veículos e viagens. A UMeR guarda também a informação sobre o utilizador a usar a aplicação atualmente. Fornece informação sobre os seus dados à classe Main para que esta os possa disponibilizar e agir de acordo com a informação guardada. Recebe também informação da Main quando é necessário guardar dados.

**User**

Classe abstrata dos utilizadores. É a classe onde estão os atributos e métodos comuns a todos os utilizadores. Isto é:

- email;

- password;

- nome,

- morada;

- data de nascimento;

- histórico de viagens;

- total gasto/faturado em viagens (gasto no caso de clientes, ganho no caso dos condutores.

**Client**

Subclasse da User. É a classe dos utilizadores que são apenas clientes. Para além da informação da subclasse User, esta classe guarda a localização deste cliente.

**Driver**

Subclasse de User. É a classe dos utilizadores que são condutores. Para além da informação da subclasse User, esta classe guarda também:

- a classificação do utilizador;

- o grau de cumprimento do condutor;

- total de quilómetros feitos;

- disponibilidade;

**Vehicle**

Classe abstrata dos veículos. É a classe onde se encontra os atributos e métodos comuns a todos os veículos. Isto é:

- matricula;

- velocidade média;

- total de quilómetros feitos;

- fiabilidade;

- lugares;

- localização;

**Car**

Subclasse de veículo. É a classe dos veículos que são carros

**Van**

Subclasse de veículo. É a classe dos veículos que são carrinhas

**Motorcycle**

Subclasse de veículo. É a classe de veículos que são motas.

**Coordinates**

Classe de coordenadas. Simplesmente contem uma coordenada no eixo x e e no eixo y. Usada como o tipo para a localização de clientes e veículos. Implementa também um método de calculo de distância entre duas coordenadas.

**Address**

Classe de morada. Usada para guardar dados sobre uma morada. Contem simplesmente uma Cidade e um Pais. Usada para guardar morada de clientes.

**Interface**

Interface que imprime alguns menus que se mantem iguais. Implementada pela classe Main

**EmailValidator**

Interface responsável por analisar o formato de um dado email, e verificar se tem um formato valido.

**Trip**

Classe da viagem. Tem toda a informação relativa a uma viagem efetuada. Contem informação sobre:

- id que identifica unicamente a viagem;

- cliente que pediu esta viagem;

- condutor que efetuou esta viagem;

- táxi utilizado na viagem;

- data e hora da viagem;

- localização inicial do taxi;

- localização inicial do cliente;

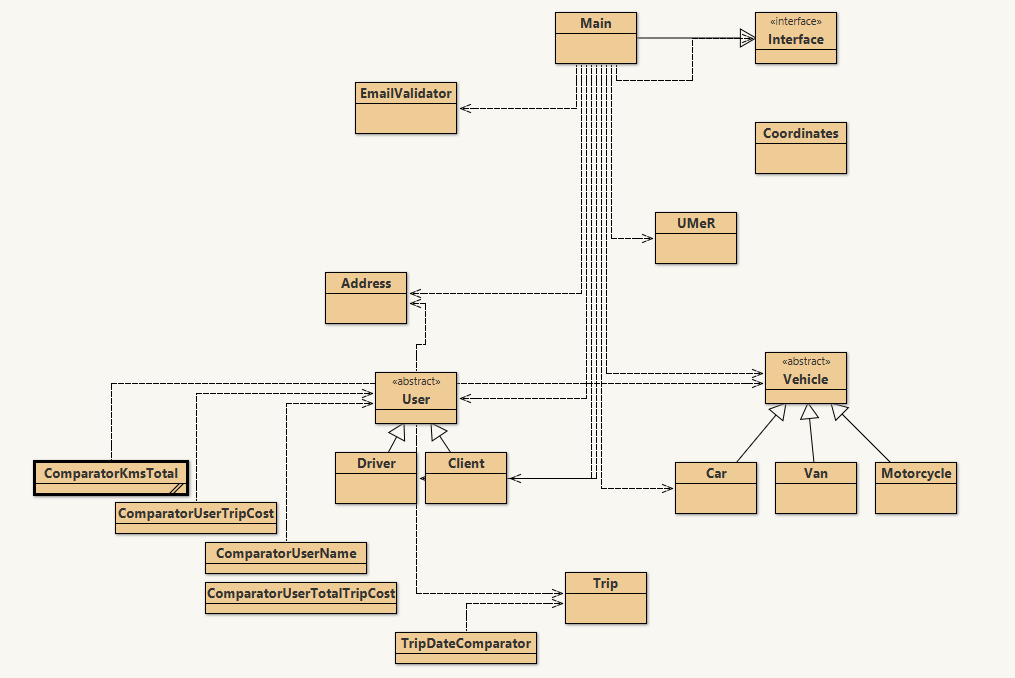
- destino;

- tempo estimado para a duração da viagem;

- tempo efetivo da duração da viagem;

- custo estimado da viagem;

- custo real/valor pago pela viagem;



**Análise aprofundada das classes**

**User**

Classe abstrata. Todos os atores do sistema, sejam clientes ou condutores, tem como super-classe a classe User. A esta classe faz parte a seguinte informação:

Variáveis de instância:

|  |  |  |
| --- | --- | --- |
| **Nome de atributo** | **Tipo** | **Função** |
| Email | String | Email do utilizador. Este atributo será único para cada utilizador |
| Name | String | Nome de utilizador |
| Password | String | Password do utilizador. Permite autenticar o utilizador que tente entrar na aplicação com o email dele. A password não se deve encontrar em forma visível, mas passará por um processo de hashing antes de ser guardada. |
| Address | Address | Morada do utilizador. Cidade e País. |
| Birthday | LocalDate | Data de nascimento do utilizador. |
| TripHistory | List<Integer> | Histórico de viagens efetuadas pelo utilizador. Para manter consistência de dados, apenas é guardado na lista o numero identificador da viagem em questão. Dentro da classe esta lista será sempre uma LinkedList<Integer> |
| TotaTripCost | Double | Dinheiro total gasto em viagens. |

Contrutores:

|  |  |  |  |
| --- | --- | --- | --- |
| Declaração | Nome | Parametros | Descrição |
| Private | User | - | Contrutor vazio. Este construtor pois não é suposte ser usado. Inicializa strings com “N/A”, inteiros e dubles a 0 e outros dados a NULL |
| Public | User | String nome  Address address  LocalDate birthday  String email  String password | Contrutor que cria um User com nome, morada, data de nascimento, email e password passados como parametros. O histórico de viagens é inicializado como LinkedList<Integer> vazia. TotalTripCost é inicializado a 0. |
| Private | User | String nome  Address address  LocalDate birthday  String email  String password  List<Integer>  Double totalTripCost | Construtor que cria User com todos os atributos de utilizador passados por parâmetros. Este construtor está criado como privado pois não achamos que faria sentido criar um novo utilizador com um histórico já preenchido nem criar com um valor total gasto em diferetente de 0. No entanto, no contexto da classe é util ter este construtor para o metodo de clone. |
| Public | User | User user | Construtor que cria um utilizador a partir de outro, copiando todos valores nos atributos do utilizador passado como argumento. |

Metodos de instância:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Declaração** | **Return** | **Nome** | **Parametros** | **Descrição** |
| Public abstract | User | Clone | - |  |
| Public | String | toString | - |  |
| Public | Boolean | Equals | Object o |  |
| Public | String | getNome | - |  |
| Public | Address | getAddress | - |  |
| Public | LocalDate | getBirthday | - |  |
| Public | String | getEmail | - |  |
| Public | String | getPassword | - |  |
| Public | Double | getTotalTripCost | - |  |
| Public | List<Integer> | getTripHistory | - |  |
| Public | Void | setName | String name |  |
| Public | Void | setTripHistory | Address address |  |
| Public | Void | setAddress | List<Integer> tripHistory |  |
| Public | Void | setBirthday | Int year  Int month  Int day |  |
| Public | Void | setBirthday | String email |  |
| Public | Void | setEmail | String password |  |
| Public | Void | setPassword | List<Integer> tripHistory |  |
| Public | Void | setTripHistory | Double cost |  |
| Public | Void | setTotalTripCost | Trip trip |  |