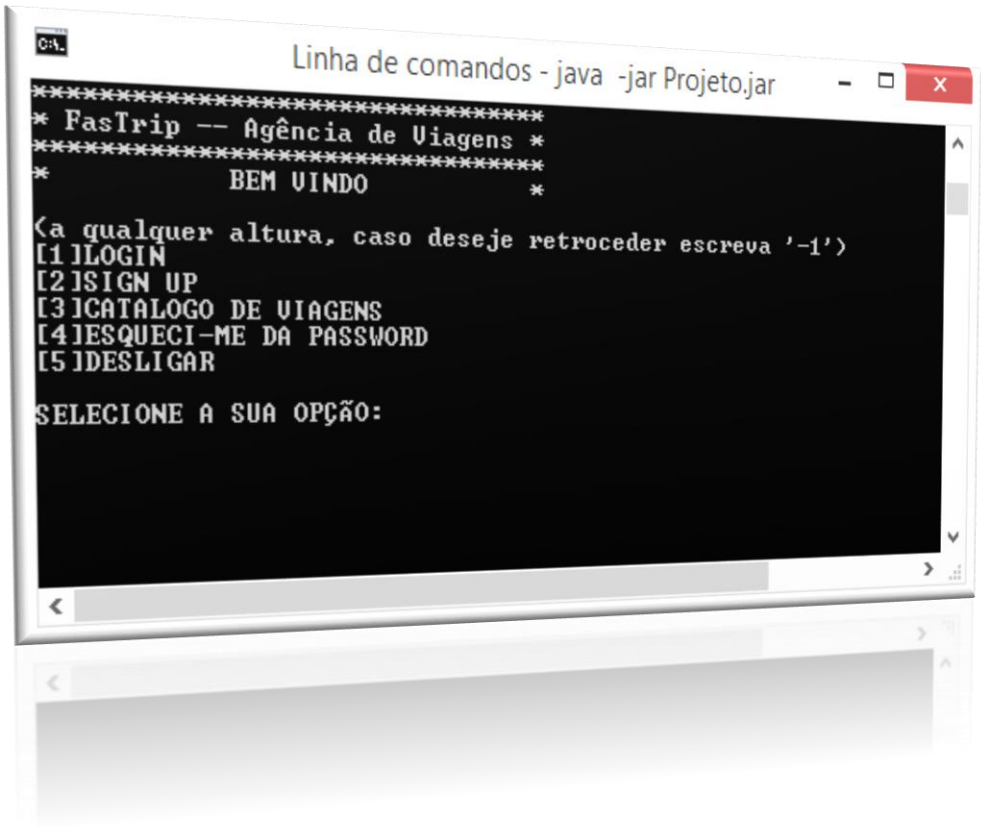
Universidade de Coimbra
Faculdade de Ciências e Tecnologia
DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

*PROGRAMA PARA A CADEIRA DE PROGRAMAÇÃO ORIENTADA *
A OBJETOS (POO)

* DOCENTE: PROFESSORA MARÍLIA PASCOAL CURADO *

* TURMA: TP4 *



Trabalho Realizado Por:

- Joel Filipe Rogão Pires Nº2014195242

- Francisco Nogueira Fernandes Nº2014200243

RELATÓRIO

Eis as nossas principais Estruturas, Ficheiros e Objetos:

- Temos **5 principais ArrayLists** que armazenam a informação toda do programa no que há manipulação e alteração de objetos diz respeito:
    - listaU -> lista com todos os utilizadores
    - listaV -> lista com todas as viagens (as que cuja data de partida já passou e as que ainda estão em vigor)
    - listaA -> lista com todas os autocarros que a agência de viagens possui
    - listaR -> lista com todas as reservas ativas
    - listaC -> lista com todas as reservas canceladas
    - listaE -> lista com todas as reservas em espera

- Temos **7 ficheiros** principais:
    - **listaUtilizadores.txt ->** permite guardar localmente os objetos da listaU
    - **listaViagens.txt ->** permite guardar localmente os objetos da listaV
    - **listaAutocarros.txt ->** permite guardar localmente os objetos da listaA
    - **listaReservas.txt ->** permite guardar localmente os objetos da listaR
    - **listaReservasC.txt ->** permite guardar localmente os objetos da listaC
    - **listaReservasE.txt ->** permite guardar localmente os objetos da listaE
    - **estatisticas.txt ->** permite guardar as informações para que durante o programa se tenham acesso e se guardem estatísticas.

- Temos **17 classes** diferentes, das quais apenas **11** se podem instanciar em objetos, a saber:
    - Administrador
    - Autocarro
    - Cartão
    - Cheque
    - Dinheiro
    - Comentario
    - FicheiroDeObjetos
    - FicheiroDeTexto
    - Login
    - Premium
    - Regular
    - Reserva
    - Viagem

O Programa simula uma Agência de Viagens onde Utilizadores podem fazer diversos tipos de operações. Quando o programa arranca são feitas 2 coisas importantes:

- Carregamento do conteúdo que está nos ficheiros para listas que contêm os objetos necessários no decorrer programa;
- Atualização dessas mesmas listas por verificar se a data de alguma viagem já ocorreu, sinalizando isso nos objetos viagens e cancelando as suas respetivas reservas.

A partir daí o utilizador entra num ciclo (*mainLoop()*) que lhe permite aceder sempre ao menu das principais e mais básicas operações que pode fazer, podendo desligar o programa adequadamente sempre que precisar. Com desligar adequadamente queremos dizer que o utilizador insere a opção 5 e o programa executa a função *term()* que se assegura que a informação que constam nas listas são gravadas para os ficheiros respetivos.

A implementação do nosso programa segue os trâmites do que foi sugerido pelo enunciado, porém existem algumas particularidades na forma como o programa foi implementado e executado que gostaríamos de chamar à atenção:

- Sem se autenticar o Utilizador apenas pode consultar as Viagens disponibilizadas pela Agência de Viagens, mas está impedido de fazer quaisquer operações.

- Quando um utilizador se autentica ou se regista, o programa sabe se ele é Administrador ou Cliente, redirecionando-o para os menus específicos de cada um.

- É disponibilizada ao Utilizador a possibilidade de voltar atrás nas suas escolhas nos respetivos menus, bastando que, para isso, insira “-1”.

- A forma como os objetos são listados diferem de Administrador para Cliente, sendo que o Administrador consegue imprimir no ecrã toda, ou quase toda a informação relativa a um objeto.

- Cada reserva possui o código identificativo da Viagem a que diz respeito (em vez do objeto Viagem), e cada Viagem possui uma lista das Suas Reservas.

- A atribuição de lugares numa reserva segue uma ordem, a saber: os lugares dos autocarros são preenchidos do fim para o início, e só depois de o primeiro autocarro estiver cheio é que se passa para o subsequente.

- O Administrador gere reservas e comentários, mas não lhe cabe a ele criar um comentário ou reserva.

- Implementamos um determinado rating associado a uma viagem e que é calculado da seguinte forma: média das pontuações dos comentários feitos a essa viagem + 0.2 * número de reservas que essa viagem. Achamos que uma simples média não seria justo, pois as pontuações de viagens mais comentadas deveriam ser valorizadas.

- Implementamos 3 formas diferentes de um cliente poder pagar a sua reserva: com cartão, dinheiro ou cheque.

- Implementamos o método checkString() bem como outras proteções ao inserir o nif, telefone, etc, e que previnem a inserção de dados não reais.

- Tivemos em atenção o risco que seria o Administrador cancelar uma viagem que já possuía reservas, eliminando essas reservas adequadamente: enviando as respetivas notificações e dinheiro para os clientes.

- Viagens Canceladas não são removidas, fazem parte do histórico das viagens da agência, mas não do seu catálogo disponível. Não há então uma remoção da lista de Viagens, mas uma sinalização no atributo *isRealizada* de cada objeto Viagem.

- Reservas canceladas despoletam notificações para os utilizadores que possuíam reservas em espera, bem como a incrementação das vagas dessa viagem e dos lugares dos seus autocarros.

- Para evitar conflitos, o Administrador só poderá alterar autocarros associados a uma viagem por outros cuja lotação em conjunto é superior.

- O caso em que o Administrador altera características de uma viagem que já possui reservas também é cuidadosamente tratado.

- Sempre que uma reserva fica em lista de espera, o pagamento é igual a null (não existe).

- O Cliente pode consultar a qualquer momento as diversas notificações que lhe são enviadas e armazenadas num ArrayList que é atributo da sua classe.

- Os Utilizadores podem a qualquer momento alterar os seus dados, mas não informações essenciais como por exemplo o seu username.

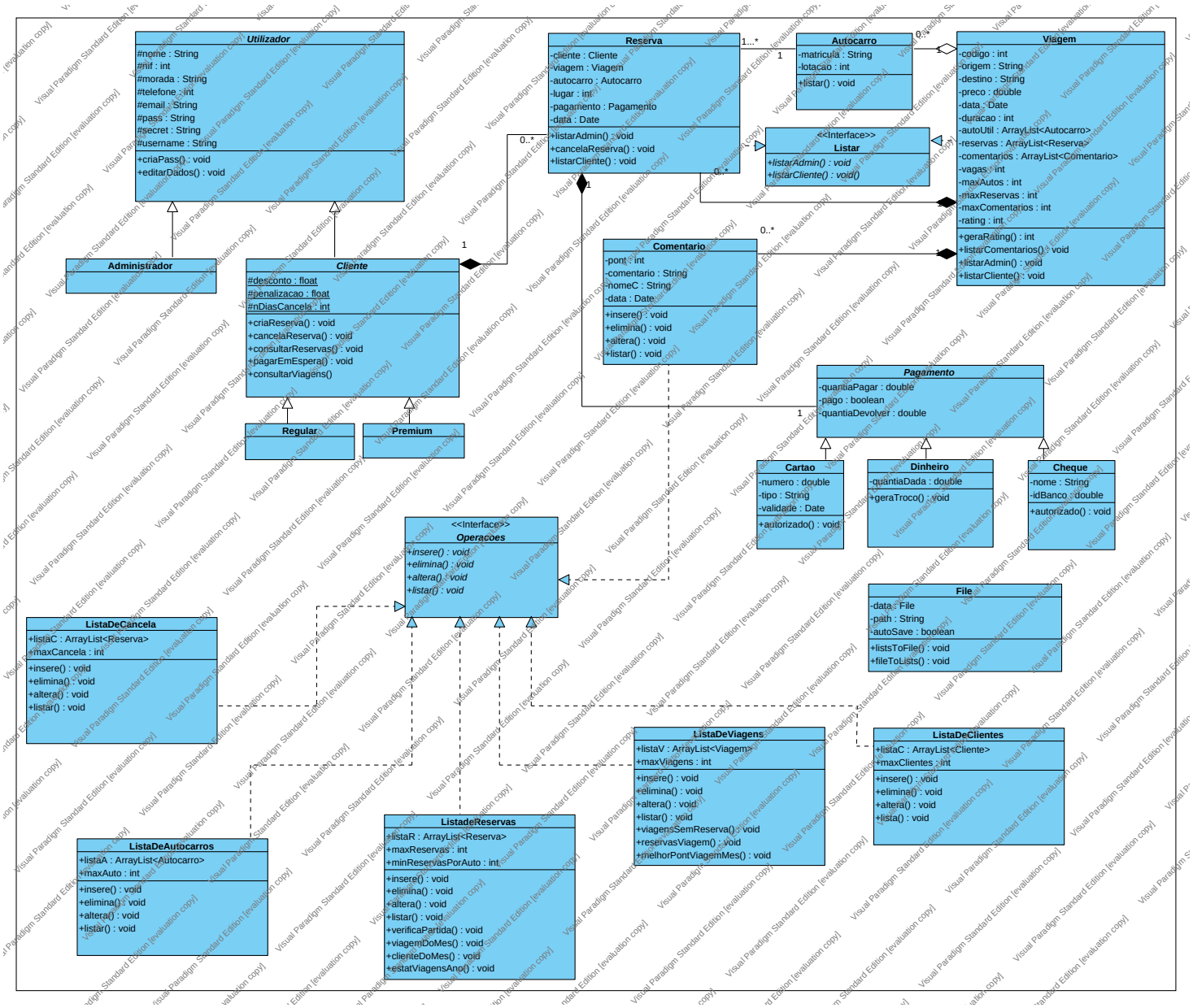
- Cada utilizador possui uma password secreta alternativa que lhe permite recuperar a sua password original quando assim o precisar.

- Existe uma password geral de administradores que permite que apenas reais administradores se possam registar no programa.

- O Administrador pode alterar quase tudo numa viagem, à exceções de informações essenciais como o código identificativo, Origem ou Destino da Viagem, pois isso seria semelhante a criar uma Viagem nova.

- Como um cenário real seria um cliente fazer uma reserva para si e para a sua família, nós permitimos no nosso programa que mais que uma reserva esteja associada ao mesmo cliente.

Implementamos diversas proteções ao longo do programa que permitem que os dados sejam coerentes com a realidade, bem como tratamento de todas as exceções para que o programa não colapse.


Utilizador
#nome : String
#nif : int
#morada : String
#telefone : int
#email : String
#pass : String
#secret : String
#username : String
+criaPass() : void
+editarDados() : void
Administrador
Cliente
#desconto : float
#penalizacao : float
#nDiasCancela : int
+criaReserva() : void
+cancelaReserva() : void
+consultarReservas() : void
+pagarEmEspera() : void
+consultarViagens()
Regular
Premium
Reserva
-cliente : Cliente
-viagem : Viagem
-autocarro : Autocarro
-lugar : int
-pagamento : Pagamento
-data : Date
+listarAdmin() : void
+cancelaReserva() : void
+listarCliente() : void
Autocarro
-matricula : String
-lotacao : int
+listar() : void
<<Interface>>
Listar
+listarAdmin() : void
+listarCliente() : void()
Viagem
-codigo : int
-origem : String
-destino : String
-preco : double
-data : Date
-duracao : int
-autoUtil : ArrayList<Autocarro>
-reservas : ArrayList<Reserva>
-comentarios : ArrayList<Comentario>
-vagas : int
-maxAutos : int
-maxReservas : int
-maxComentarios : int
-rating : int
+geraRating() : int
+listarComentarios() : void
+listarAdmin() : void
+listarCliente() : void
Comentario
-pont : int
-comentario : String
-nomeC : String
-data : Date
+insere() : void
+elimina() : void
+altera() : void
+listar() : void
Pagamento
-quantiaPagar : double
-pago : boolean
-quantiaDevolver : double
Cartao
-numero : double
-tipo : String
-validade : Date
+autorizado() : void
Dinheiro
-quantiaDada : double
+geraTroco() : void
Cheque
-nome : String
-idBanco : double
+autorizado() : void
<<Interface>>
Operacoes
+insere() : void
+elimina() : void
+altera() : void
+listar() : void
File
-data : File
-path : String
-autoSave : boolean
+listsToFile() : void
+fileToLists() : void
ListaDeCancela
+listaC : ArrayList<Reserva>
+maxCancela : int
+insere() : void
+elimina() : void
+altera() : void
+listar() : void
ListaDeViagens
+listaV : ArrayList<Viagem>
+maxViagens : int
+insere() : void
+elimina() : void
+altera() : void
+listar() : void
+viagensSemReserva() : void
+reservasViagem() : void
+melhorPontViagemMes() : void
ListaDeClientes
+listaC : ArrayList<Cliente>
+maxClientes : int
+insere() : void
+elimina() : void
+altera() : void
+lista() : void
ListaDeAutocarros
+listaA : ArrayList<Autocarro>
+maxAuto : int
+insere() : void
+elimina() : void
+altera() : void
+listar() : void
ListadeReservas
+listaR : ArrayList<Reserva>
+maxReservas : int
+minReservasPorAuto : int
+insere() : void
+elimina() : void
+altera() : void
+listar() : void
+verificaPartida() : void
+viagemDoMes() : void
+clienteDoMes() : void
+estatViagensAno() : void