Tema 3 - Campanha 'Viaje e Poupe'

Comboios Portugal

Trabalho prático - Parte 1

Trabalho realizado por:
João Filipe Carvalho de Araújo - up201705577
Maria Helena Viegas Oliveira Ferreira - up201704508
Nuno Miguel Fernandes Marques - up201708997

no âmbito da disciplina de Algoritmos e Estruturas de Dados
do segundo ano da Faculdade de Engenharia da Universidade do Porto

Trabalho do grupo 5 da turma 2

Data de entrega de 23 de novembro de 2018

Índice

1.Tema do trabalho

O tema do trabalho que nos foi proposto foi o tema número 3. Este tema tem por base a campanha ‘Viaje e Poupe’ dos comboios Portugal. Esta campanha da CP pretende promover a utilização dos comboios, nomeadamente dos Alfas Pendulares e Intercidades. A CP decidiu portanto desenvolver uma aplicação para a realização de reservas de bilhetes bem como a disponibilização de cartões de desconto para clientes frequentes que queiram aderir.

A aplicação tem que ser capaz de lidar com os dois tipos de clientes possíveis, os passageiros passageiros sem cartão (que decidiram não aderir a campanha) e os passageiros com cartão (já registrados).

Os cartões podem ser de 3 tipos: cartão de viagem 25 (com o custo de 39€/mês), cartão de viagem 50 (com o custo de 69€/mês) e cartão de viagem 100 (com o custo de 149€/mês) . Os passageiros com cartão são passageiros que possuem um cartão associado, no qual está registada vária informação com nome, profissão, data de nascimento, viagens efetuadas, etc. Aquando da aquisição de um bilhete, o passageiro com cartão tem sempre uma redução de acordo com o tipo de cartão adquirido: Com um cartão de viagem 25 o cliente poupa 25% nas viagens de comboio efetuadas, com o cartão de viagem 50 o cliente poupa 50% e com o cartão de viagem 100 pode viajar em qualquer comboio sem bilhete durante a vigência da subscrição.

Ainda na recente estratégia de marketing, foi identificada a possibilidade de se um determinado comboio (AP ou IC) possuir menos de 50% dos lugares vendidos quando falta menos de 48h para o início da viagem, a CP pode decidir efetuar uma promoção, passando os bilhetes a ser vendidos com um desconto de 70% do valor original. Este desconto sobrepõem-se ao de 50% e 25% da adesão à campanha. O sistema deve incluir e gerir toda a informação relativa aos comboios

(quais os comboios AP e IC que existem), viagens (viagem e comboio associado), passageiros e reservas (compra e devolução de bilhetes).

2.Ficheiros

Para a implementação desta solução criaram-se doze ficheiros metade dos quais são header files.

2.1.Cartao.h e Cartao.cpp

Começando com os dois ficheiros designados cartão. No header file “Cartao.h” encontra-se declaradas e devidamente comentadas três classes, a classe Cartão, Registo e BaseClientes.

2.1.1. Classe Cartao

A classe Cartao é usada para criar cartões com modalidades de desconto diferentes. Os nomes da modalidades, que são as possibilidades de nome dos cartões, são “viagem 25”, “viagem 50” e “viagem 100” , e o preço mensal da subscrição é de 39, 69 e 149 euros, respetivamente. A cada preço está também associado um desconto, de 25%, 50% e 100%, de acordo com o nome do cartão. Nos membros função, encontramos o construtor, métodos de aceder aos membros dados e um método que retorna uma string com toda a informação relacionada com o cartão. Para além destes membros função, é também possível adicionar e alterar os cartões. Também definimos o operador de equivalência.

2.1.2. Classe Registo

A classe Registo recebe a informação do seu utilizador, como o seu cartão associado e dados pessoais, nomeadamente o nome, a profissão, a data de nascimento e o estado ativo ou desativo do registo. Tem ainda as viagens efetuadas pelo mesmo (num vetor de apontadores da classe compra que se encontra inicialmente vazio), vetor este designado de histórico. Nos membros função fizemos um

overloading do construtor, com diferença em número de argumentos. Um possui mais um elemento, o booleano verdadeiro se os registos estão ativos, e falso se desativados. Por omissão, implementamos em “Cartao.cpp” que a subscrição está ativa se o booleano não for passado como argumento. Para além do construtor e o destrutor, possui métodos para dar acesso aos seus membros dados e métodos para alterar o cartão e o seu estado, isto é, ativar ou desativar registos, passando como argumento o novo estado da subscrição. Pode-se também visualizar o número, adicionar e eliminar compras de viagens no histórico. Para a remoção das compras do registo, no caso de devolução de bilhete, existe um método para devolver um vector de compras, só com as compras das viagem que ainda não foram efetuadas. Desta maneira ficamos a saber se as viagens do histórico já ocorreram ou ainda estão por ser realizadas, através do método getCompraAtiva e listCompraAtiva, ambas com o mesmo propósito, no entanto, a primeira devolve um vetor com as viagens que estão por realizar e a segunda uma string com as mesmas. Para aceder a um histórico completo numa string também existe um método conveniente.

2.1.3. Classe BaseClientes

A classe BaseClientes é responsável por guardar todos os cartões e os registos dos passageiros, é a que nos dá acesso às duas classes referidas anteriormente. Por isso, a classe tem nos seus membros dados dois vetores com apontadores, para cartões e para registos e ainda um valor id, que é usado para guardar qual o Registo que está a ser usado neste momento, permite não ter de perguntar constantemente ao utilizador o seu id para cada operação. O construtor inicializa os vetores vazios e o id com o valor de 0. O destrutor liberta a memória alocada pelo construtor, isto é, apaga todos os apontadores para cartões e registos. A esta classe estão associados muitas funções, como a getRegisto(), função constante que retorna a informação de um passageiro em específico, isto é retorna um apontador para um registo; as funções getInformacao() e getInfoCartao(), que retornam uma string a primeira com a data de nascimento de um passageiro e a segunda com um cartão de vetor de cartões. Podemos também saber o número de cartões e registos na base de dados, através das funções

getNumCartoes e getNumRegistos, funções constantes que retornam o número de cartões e de registos, respetivamente. De maneira a editar os dois vetores mais facilmente, criamos funções que adicionam e removem cartões e registos nos seus respectivos vetores. Pode-se também alterar o id, através da função setId, que tem como argumento um inteiro com o valor da id que se quer colocar. Por fim, implementamos métodos que fazem load e save de cartões e registos de e para um ficheiro, respetivamente, de maneira aos registos não ser apagados quando se termina a execução do programa.

2.2.Viagem.h e Viagem.cpp

Em viagem.h encontra-se a declaração da classe viagem, e no ficheiro viagem.cpp a implementação das funções declaradas nessa classe.

2.2.1. Classe Viagem

Esta classe cria e gere todas a viagens (reserva, devolução de bilhetes). À classe Viagem estão associadas duas strings, uma com a origem e outra com o destino da viagem. Estão associados dois valores não inteiros, o da distância e o preço base da viagem (calculado pela multiplicação do preço por km pela distância) e dois valores positivos inteiros, o das vagas e o número de compras às quais não estão associadas nenhum cartão, logo não tem os seu dados nos registos, designadas compras anônimas. É gerido o número de compras total, para tornar possível a devolução de bilhetes. Temos também três apontadores, para comboios, datas de partidas e horas de partidas. O construtor da classe possui um overloading em número de argumentos, em concreto, a classe tem 3 construtores. O primeiro é para ser usado na criação de uma viagem por argumentos, e os restantes utilizados no carregamento de viagens através de um ficheiro (dependendo se a viagem e carregada para venda ou para o histórico na classe compra). Para iniciar é sempre preciso fornecer como argumento as strings de origem e destino. No primeiro o construtor passa-se também como argumento a distância, e os três apontadores. No segundo, para além

de todos os argumentos já referidos, passa-se ainda o número de vagas e de compras anônimas. Ao terceiro passa-se como argumento, para além das strings, apontadores para as datas e as horas de partida o preço base da viagem, isto é, quanto o utilizador pagaria sem os descontos que pode ter direito.

O preço final da viagem, para além de se basear no preço base, é também influenciado pela data e hora de compra e pela lotação disponivel, sendo por isso que estes dados fazem parte da classe. Se um determinado comboio (AP ou IC) possuir menos de 50% dos lugares vendidos quando falta menos de 48h para o início da viagem, a CP pode decidir efetuar uma promoção, passando os bilhetes a ser vendidos com um desconto de 70% do valor original. O desconto aplicado é o mais alto, se o desconto anteriormente descrito estiver disponível sobrepõem-se aos descontos de 25% e 50 % da subscrição do serviço “viaje e poupe” da CP. Devido aos descontos possíveis, o método getPrecoFinal calcula o preço final a pagar, sem argumentos ou com um cartão como argumento, caso o utilizador esteja subscrito.

A classe tem também funções que retornam os membros dados, um deles retorna uma string com toda a informação associada. Tem métodos para a reserva e devolução de bilhetes. Fizemos overloading desta função; se chamarmos a função com o argumento cartão, temos acesso aos descontos da subscrição.Por fim fizemos a redefinição do operador de comparação, sendo passado apenas um argumento, para comparar objectos Viagem. E ainda uma função que retorna um booleano com o valor se se pode ou não efetuar mais compras da viagem, em suma, se a lotação máxima já foi atingida.

2.3.Comboio.h e Comboio.cpp

No ficheiro comboios.h estão declaradas quatro classes. A classe Comboio é uma superclasse, e as classe AlfaPendular e Intercidades são derivadas desta classe base Comboio, sendo por isso subclasses e servem para distinguir Alfa Pendulares e Intercidades.

2.3.1. Classe Comboio

À classe Comboio está associado uma string, o nome, e quatro números, a capacidade do comboio, a velocidade, a identificação e o

preço por quilômetro. Este último valor é mais tarde usado para calcular o preço base de uma viagem, considerando que uma viagem e mais cara se o comboio for mais rápido e/ou mais luxuoso. Esta classe possui um construtor, um destrutor e um método virtual, pois cada subclasse deverá implementar o seu próprio método de getTipo, dado que os comboios são de diferentes tipos (AP ou IC). Há também funções que retornam os membros-dados, que redefinem a identificação e que a tornam visível, através de uma ostream, um método de saída (<<) que devolve um objecto Comboio no formato string “nome-tipo”.

2.3.2. Classe AlfaPendular e Intercidades

As classes derivadas AlfaPendular e Intercidades apenas implementam um construtor e destrutor e o seu próprio método de getTipo(), dando se assim o polimorfismo. No ficheiro comboios.cpp implementa-se estas classes descritas.

2.3.3. Classe Frota

Neste mesmo ficheiro encontramos a declaração da classe Frota, que armazena todos os comboios num vetor de apontadores para comboios, designado comboios. Encontram-se funções que devolvem a informação da frota de comboios, do número de comboios e a possibilidade de encontrar um comboio através da sua identificação. Permite ainda adicionar e remover comboios do vetor e ainda guardar e carregar toda a informação dos comboios em ficheiro.

2.4.Bilheteira.h e Bilheteira.cpp

O grupo achou também importante criar uma classe Bilheteira, adicionada ao ficheiro bilheteira.h, juntamente com a classe compra, dado que é a operação efetuada nas bilheteiras.

2.4.1. Classe Bilheteira

A classe Bilheteira, que dá nome ao ficheiro, tem associado dois membros-dados, um vetor com apontadores para todas as viagens ativas, e um apontador para a frota de comboios. A esta classe está associado a adição de viagem, load e save de viagem através de ficheiros externos. Cada vez que a classe é atualizada, com a função updateViagens, removem-se as viagens que já partiram, e que portanto não tem valor nesta classe de compra, através do tempo local do computador. Também se ordena o vetor por ordem da partida da viagem, usando um algoritmo selection sort. O vector de viagens é atualizado cada vez que é mostrado ou carregado um ficheiro de viagens No entanto, apesar do destrutor, não destrói o objeto se este for usado pela classe a seguir descrita.

2.4.2. Classe Compra

Na classe Compra, o construtor receberá uma viagem, o cartão usado para a compra, o total da compra, e a sua data e hora, que são os seus membros-dados. Esta classe tem como função criar o histórico de compras de um utilizador, mantendo sempre correto o preço e cartão usado para a compra mesmo que utilizador, posteriormente, mude de cartão e a viagem seja removida da bilheteira. Tem métodos que retornam os dados membros, em especial destaque para um que retorna uma string com a informação associada à compra e para a redefinição do operador de comparação.

2.5. Datashoras.h

As estruturas enumeradas anteriormente fazem uso de duas classes, que criam objetos com uma data e a hora, declaradas e definidas em datashoras.h. O grupo decidiu que dado que as estas só podem tomar valores específicos, contrariamente a valores como a velocidade e preço, seria melhor criar uma classe para as definir individualmente. Estas classes podem ser construídas com valores inteiros ou com uma string. Há também a redefinição dos operadores de comparação e de igualdade e a possibilidade de conversão do formato HH:MM e DD:MM:AA respectivamente para um formato de

horas totais, ex (10:30 -> 10,5). As classes possuem também excepções para lidar com entradas e formatos inválidos.

2.5.1. Classe Data

Começando pela classe “Data”, que obviamente tem três inteiros positivos como membros-dado o ano, o mês e o dia; que podem ser acessados por uma função que os retorna.

2.5.2. Classe Horas

Temos ainda uma classe designada de “Horas”, à qual estão associados dois números naturais, as horas e os minutos, que podem ser acedidos através de getHora e getMin.

2.6. Geral.h e Geral.cpp

Por fim, temos ficheiros que vão ser usados ao longo de todo o projeto, que obtém a data e a hora atual do computador, lidam com inputs inválidos e os diferentes menus, dos utilizadores anónimos, dos registados e da administração.

3.Interface

No ficheiro main.cpp, geral.cpp e geral.h encontra-se a interface do trabalho.

3.1.Menu Principal

```
---MENU INICIAL---

0 - Informacao
1 - Passageiro sem cartao
2 - Passageiro com cartao
3 - Administracao
4 - Sair
```

3.2.Menu Informação

```
0 - Informacao
1 - Passageiro sem cartao
2 - Passageiro com cartao
3 - Administracao
4 - Sair
0

---Informacao---


0 - Lista de Comboios
1 - Lista de Viagens
2 - Sair
0

Lista de Comboios

Nome   Tipo  Lotacao  Velocidade   Preco por Km
c1     IC       50     200 km/h         0.2€
c2     AP       30     300 km/h         0.3€
c3     IC      100     140 km/h         0.1€
c4     AP       40     250 km/h        0.02€

---Informacao---


0 - Lista de Comboios
1 - Lista de Viagens
2 - Sair
1

Lista de Viagens

id  Origem    Destino   Distancia(KM)  Comboio  Data         Hora    Preco base(€)   Vagas
0   Porto     Coimbra   130            c2-AP    25-11-2018   10:40   39              45
1   Porto     Lisboa    300            c1-IC    20-12-2018   00:25   60              50

---Informacao---


0 - Lista de Comboios
1 - Lista de Viagens
2 - Sair
```

3.3.Menu sem cartão

```
---MENU INICIAL---

0 - Informacao
1 - Passageiro sem cartao
2 - Passageiro com cartao
3 - Administracao
4 - Sair
1

---Passageiro sem Cartao---


0 - Comprar Bilhete
1 - Devolver Bilhete
2 - Subscrever a um cartao
3 - Sair
```

3.4.Menu com cartão

```
---MENU INICIAL---

0 - Informacao
1 - Passageiro sem cartao
2 - Passageiro com cartao
3 - Administracao
4 - Sair
2
ID do seu cartao: 0

---Passageiro Com Cartao---

ID: 0

Nome: Nuno -- Cartao: "Viagem 25" -- Profissao: Est -- Data de Nascimento: 05-10-2222
0 - Comprar Bilhete
1 - Devolver Bilhete
2 - Alterar cartao subscrito
3 - Remover subscricao
4 - Historico de Viagens
5 - Sair

4

Historico de Viagens

Data da compra  Hora da Compra  Origem   Destino  Data da viagem  Hora da viagem  Preco Base(€)   Total(€)
17-11-2018      23:09           Porto    Lisboa   20-12-2018      00:25           600             450
18-11-2018      13:31           Porto    Lisboa   20-11-2018      00:25           600             450
```

3.5.Menu Administração

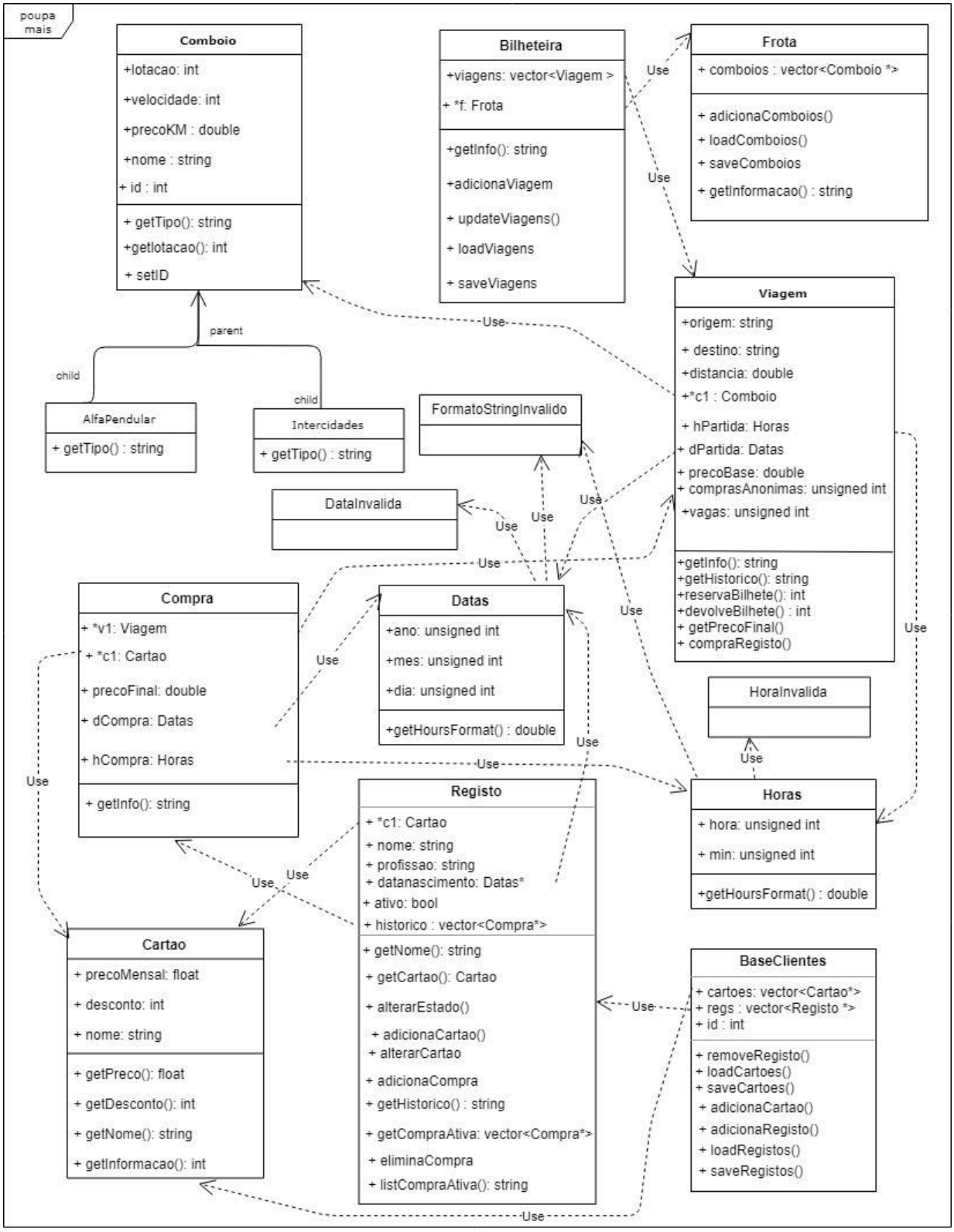
```
---MENU INICIAL---

0 - Informacao
1 - Passageiro sem cartao
2 - Passageiro com cartao
3 - Administracao
4 - Sair
3

---Administracao---


0 - Guardar Dados
1 - Carregar Dados
2 - Adicionar Comboios
3 - Adicionar Cartoes
4 - Adicionar Viagens
5 - Sair
```

4.UML



5.Conclusões

Ao longo do trabalho usamos, nos doze ficheiros, classes adequadas para representação das entidades envolvidas. Os atributos foram escolhidos de forma a caracterizar os conceitos mais importantes das entidades a representar. Utilizamos o conceito de herança e polimorfismo na classe comboios e nas derivadas alfa pendular e intercidades. Os dados usados pela aplicação são guardados em ficheiros, para utilização em futuras execuções. Todas as possíveis exceções são tratadas convenientemente. Também usamos algoritmos de pesquisa e ordenação.

6.Divisão do trabalho

O trabalho em C++ foi feito por todos, porém o maior contribuidor foi o Nuno. O relatório foi feito pela Helena e o doxygen foi feito pelo Filipe, e obviamente relido e editado por todos. Diríamos que a divisão seria que o Nuno contribui 40 %, o Filipe 25% e a Helena 35%.