Cauê Drummond Gaudie RA: 11012115
João Pedro de Abreu Martins RA: 21026515

Sistema de Reserva de Passagens Aéreas

Relatório apresentado à Universidade Federal do ABC como parte dos requisitos para aprovação na disciplina Programação Orientada a Objetos do Curso de Bacharelado em Ciência da Computação.

Prof.ª Prof. Vera Nagamuta
Prof. Saul de Castro Leite

Santo André - SP

2019

Sumário

1. Introdução

O projeto teve como objetivo a implementação um sistema para reserva de passagens aéreas para uma companhia aérea. A escolha do tema pode ser justificada pela proximidade dos integrantes do grupo com o sistema em questão, pelo desafio e as várias possibilidades que a implementação do sistema propõe aos desenvolvedores, como por exemplo, o sistema de logins. Inicialmente o grupo criou um documento com a especificação das funcionalidades do sistema, sendo que nem todas foram implementadas por questões abordadas ao longo do relatório. Após a especificação das funcionalidades do sistema, criou-se um diagrama em UML (Linguagem de Modelagem Unificada, em português) para auxílio na visualização do sistema e posteriormente

2. Usuários e Funcionalidades

Os tipos de usuários que podem utilizar as funcionalidades oferecidas pelo sistema são:

2.1. Cliente

2.1.1. Busca de Passagens: Essa funcionalidade exibe para o usuário uma lista com todas as passagens cadastradas no sistema e algumas informações, como: id, origem, destino, data, horário e o número de passagens disponíveis.

2.1.2. Busca Avançada de Passagens: Essa funcionalidade exibe para o usuário uma lista com as passagens cadastradas no sistema e suas informações utilizando um filtro. O filtro pode ser feito por origem e destino, data e horário.

2.1.3. Compra de Passagem: Essa funcionalidade permite que o cliente compre uma passagem para qualquer uma das viagens

disponíveis no sistema. O cliente passa o ID da passagem que deseja comprar e coloca o número e senha do cartão do banco, e caso ainda possuam passagens disponíveis para este voo e o banco autorize a compra, a passagem é reservada no nome daquele cliente.

2.2 Funcionários

Os funcionários são divididos em outras categorias, onde cada categoria possui as suas próprias funcionalidades, referentes aos cargos que esses funcionários exercem dentro da companhia aérea.

2.2.1. Administrador

2.2.1.1. Remove Cliente: Essa funcionalidade permite que o administrador do sistema remova o cadastro de um cliente, impedindo que esse cliente acesse o sistema novamente utilizando aquele usuário e senha.

2.2.1.2. Adiciona Funcionário: Essa funcionalidade permite que o administrador do sistema cadastre novos funcionários, podendo ser eles atendentes, pilotos e comissários, permitindo que eles acessem o sistema utilizando o usuário e senha informados no momento em o usuário foi cadastrado.

2.2.1.3. Remove Funcionário: Essa funcionalidade permite que o administrador do sistema remova o cadastro de um funcionário impedindo que esse funcionário acesse o sistema novamente utilizando aquele usuário e senha.

2.2.1.4. Adiciona Avião: Essa funcionalidade permite que o administrador do sistema cadastre novos aviões adquiridos pela companhia aérea.

2.2.1.5. Remove Avião: Essa funcionalidade permite que o administrador do sistema remova o cadastro de um avião.

2.2.2. Atendente

2.2.2.1. Adiciona Viagem: Essa funcionalidade permite que o atendente cadastre novas viagens no sistema.

2.2.2.2. Remove Viagem: Essa funcionalidade permite que o atendente remova o cadastro de uma viagem.

2.2.2.3. Relatório de Viagem: Essa funcionalidade permite que o atendente visualize um relatório da viagem que ele selecionar. Esse relatório exibe as especificações da viagem (origem, destino, data, horário, e passagens compradas), e os clientes e funcionários que estão neste voo e suas informações.

2.2.3. Comissário

2.2.3.1. Próximas Viagens: Essa funcionalidade permite que o comissário consiga visualizar as próximas viagens em que ele está incluído.

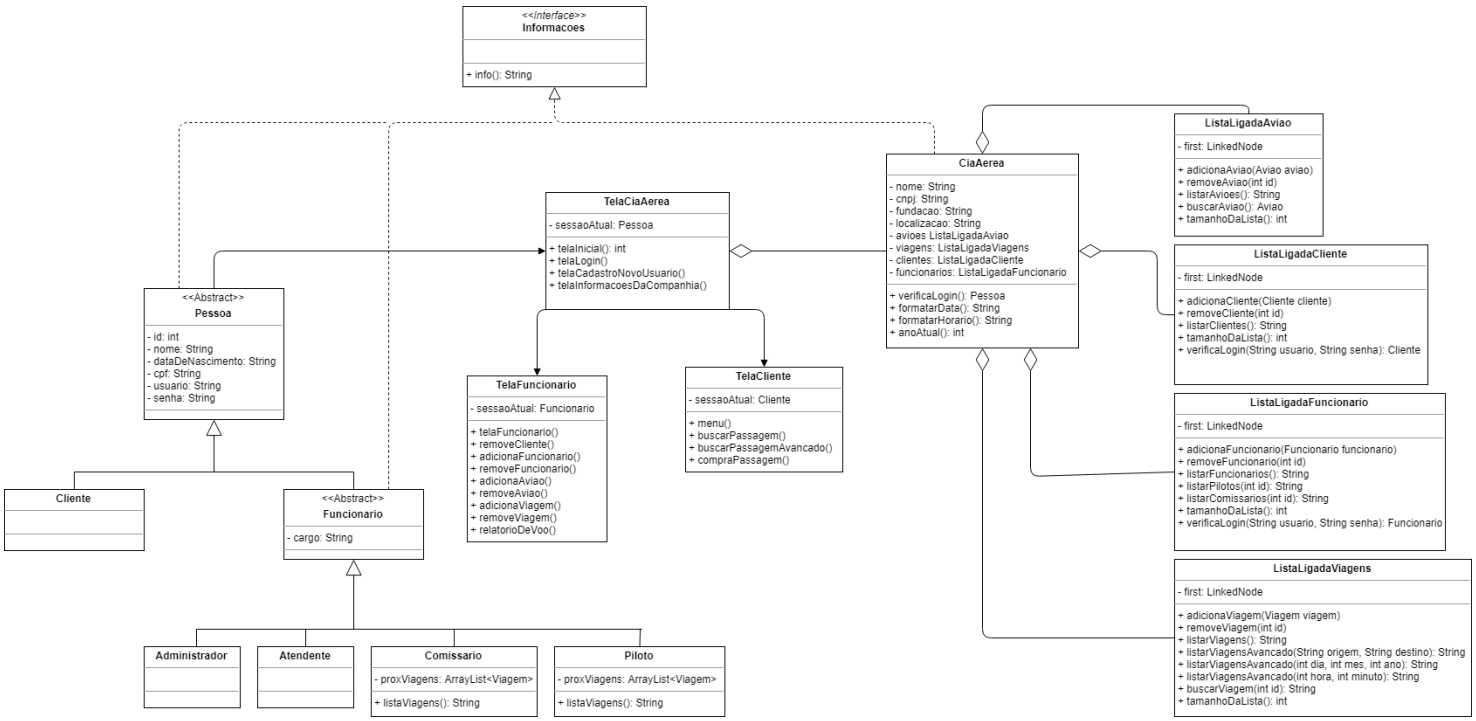
2.2.4. Piloto

2.2.4.1 Próximas Viagens: Essa funcionalidade permite que o comissário consiga visualizar as próximas viagens em que ele está incluído.

3. Implementação

Foi utilizado a classe JOptionPane da biblioteca Swing para a implementação da entrada e saída dos dados do sistema.

3.1. Diagrama UML



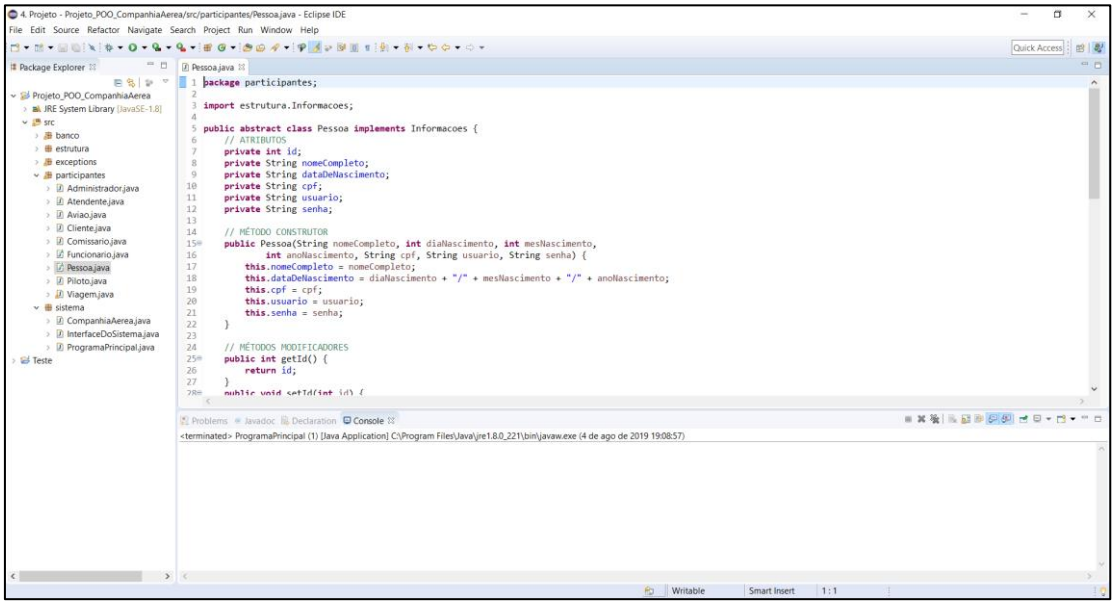
As classes referentes a implementação do banco foram ocultas do diagrama, visto que o objetivo do projeto é o desenvolvimento de um sistema de reserva de passagens aéreas. O mesmo vale para os métodos de inserção de usuários, voos e viagens assim que o sistema é executado toda vez, visto que esses métodos visam apenas facilitar a manipulação e execução de testes por parte do desenvolvedor, já que não há uma persistência dos dados inseridos, modificados e excluídos durante a execução do sistema, sendo necessário inseri-los a cada vez.

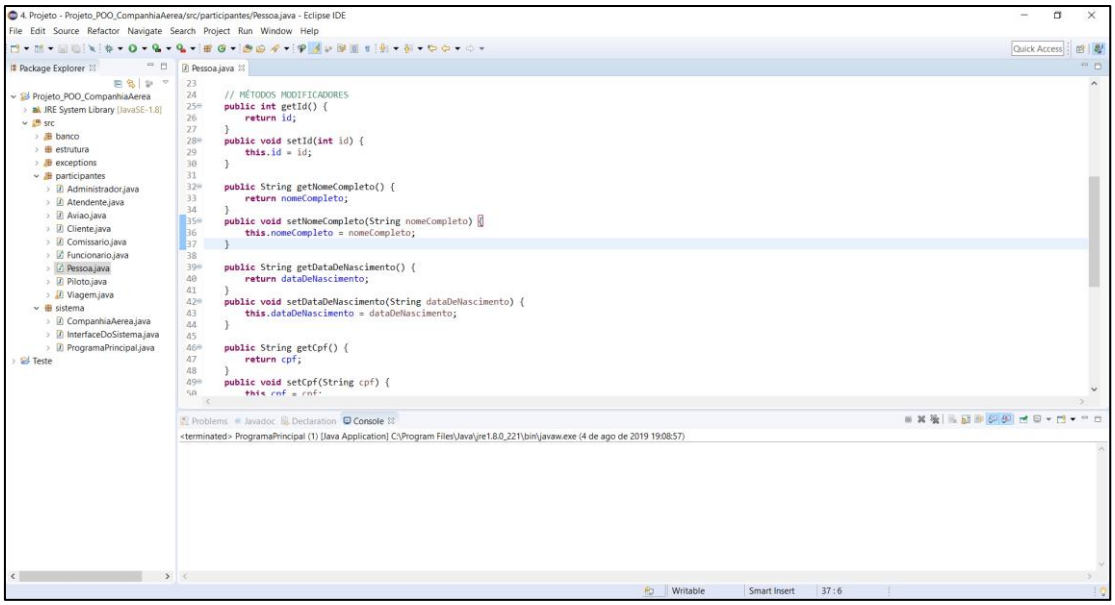
3.2. Conceitos da Disciplina

3.2.1. Herança: O conceito de Herança foi aplicado, como pode-se perceber olhando o diagrama UML, no caso dos usuários do sistema, havendo uma classe abstrata Pessoa que se estendeu para Cliente e Funcionario, sendo que Funcionario ainda estendeu para Administrador, Comissario e Piloto.

3.2.2. Interface: O conceito de Interface foi aplicado, como pode-se perceber olhando o diagrama UML, na implementação da interface Informacoes por parte das classes CiaAerea, Pessoa, Viagem e Aviao.

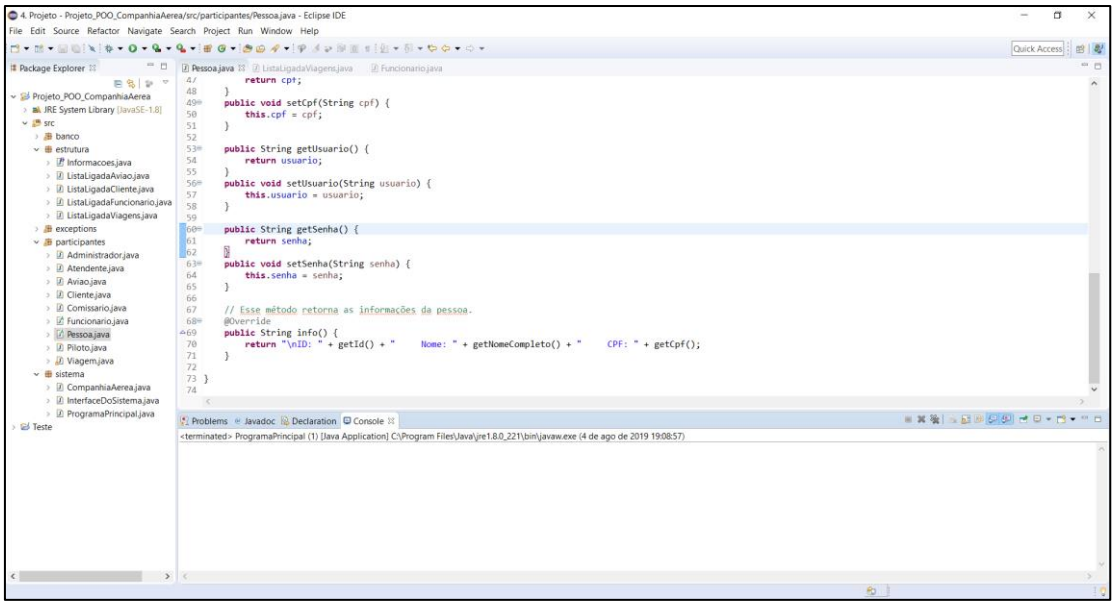
3.2.3. Encapsulamento: O conceito de Encapsulamento foi aplicado em praticamente todas as classes, como mostrado na imagem abaixo da implementação da classe Pessoa, colocando os atributos com o acesso *private* e criando os métodos modificadores (get e set) necessários para acessar e alterar esses atributos, protegendo-os para que eles não sejam acessados e alterados de forma indevida.

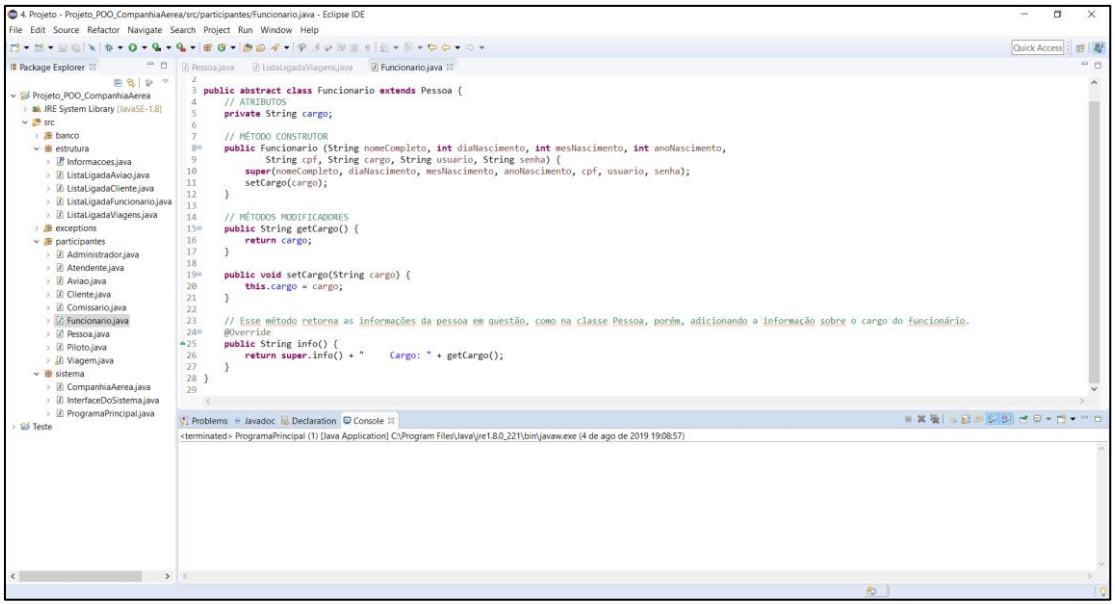




3.2.4. Polimorfismo: O conceito de polimorfismo foi aplicado em diversas áreas do código, destacando-se dois trechos, onde são aplicados os conceitos de polimorfismo por sobreposição e sobrecarga.

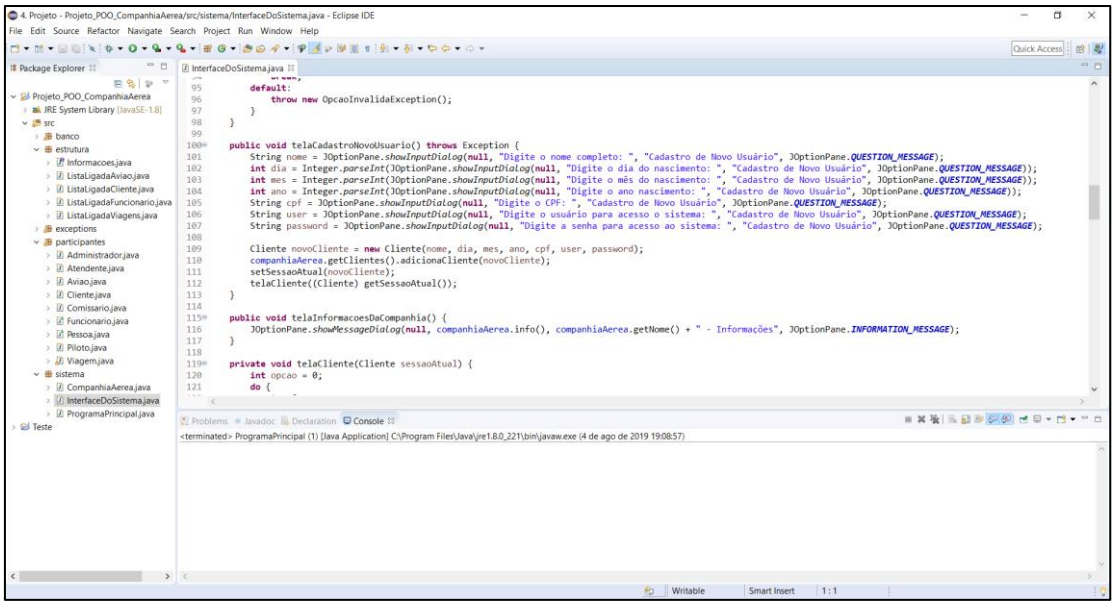
3.2.4.1. Polimorfismo por Sobreposição: Na primeira imagem abaixo, mostra um trecho da classe Pessoa, que implementa a interface Informacao, implementando o método *info()* da interface, que já por si só é um exemplo de sobreposição. Porém, na segunda imagem, vemos um trecho da classe Funcionario, que estende a classe Pessoa, onde novamente o método *info()* é sobreposto, chamando o mesmo método da superclasse, mas implementando mais uma informação antes de retornar a string para a parte do código em que esse método foi chamado.



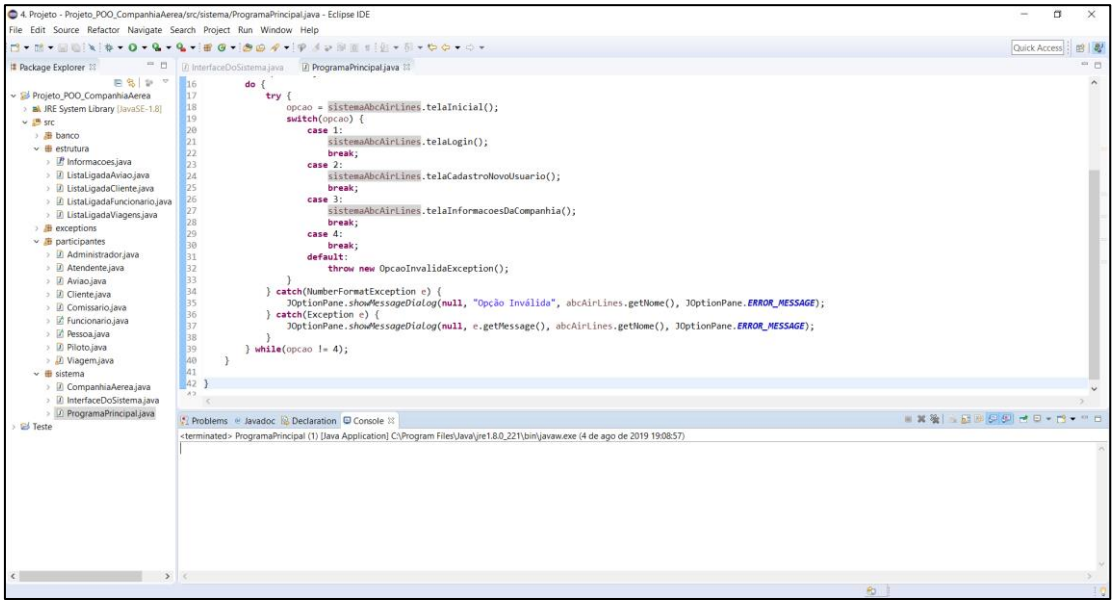


3.2.5. Tratamento de exceções: O conceito de tratamento de exceções foi aplicado em boa parte do código.

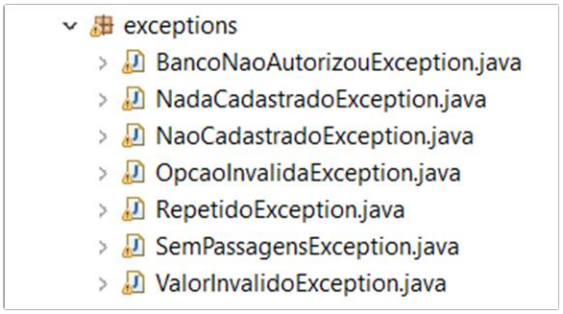
Na primeira imagem, podemos observar que o método *telaCadastroNovoUsuario()* “joga” a responsabilidade de ter que tratar qualquer exceção que ocorrer dentro do seu bloco de código para o trecho em que esse método foi chamado.



Nessa imagem podemos observar o tratamento de um certo bloco de código utilizando o try-catch, utilizando dois catch's, onde um trata o caso em que a exception capturada foi a NumberFormatException e outra que captura genericamente qualquer exception.



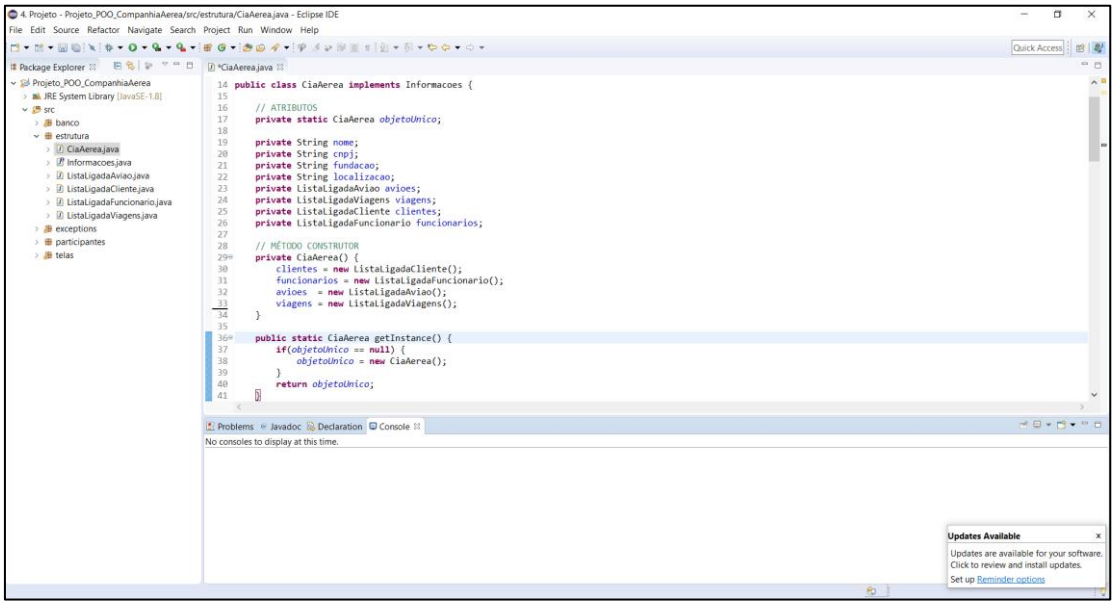
Nessa imagem podemos observar as exceptions que foram criadas, estendendo a classe Exception, para tratar algumas situações que poderiam ocorrer durante a execução do código.



3.2.6. Padrões de Projeto:

Foram utilizados dois padrões de projeto durante o desenvolvimento do sistema em questão:

3.2.6.1. Singleton: esse padrão de projeto foi utilizado para que as classes CiaAerea, Banco e TelaCiaAerea pudessem ser instanciadas apenas uma única vez. Abaixo podemos ver a implementação deste padrão na classe CiaAerea:



3.2.6.2. Façade: esse padrão de projeto foi utilizado criando-se uma classe TelaCiaAerea, onde todas as outras operações e acesso às outras classes pudessem ser feitas apenas a partir dele, sendo uma espécie de interface do sistema.

3.3. Bugs

3.3.1. É possível cadastrar dois voos para o mesmo dia e colocar neles os mesmos pilotos, copilotos e funcionários, o que seria fisicamente impossível, principalmente se o destino da primeira viagem a ser realizada não coincidir com a origem da viagem que vem a seguir.

3.3.2. É possível cadastrar um usuário e atribuir letras ao seu nome. Isso não poderia acontecer, visto que quando uma passagem aérea é comprada, o nome cadastrado pelo usuário é o nome que aparecerá na passagem comprada pela pessoa e no relatório de voo emitido pelo atendente.

3.3.3. Não é possível alterar as informações de um usuário, avião ou viagem. Para fazer alterações é necessário excluir o usuário vigente e cadastrá-lo novamente com as alterações necessárias.

Em uma situação específica, caso um funcionário ou o avião seja removido e eles estejam presentes em algum voo cadastrado no sistema, não é possível alterar as informações desse voo, sendo necessário removê-lo do sistema e cadastrá-lo novamente, e como isso as informações de quem comprou passagem são perdidas.

4. Conclusão

Foi possível aplicar vários dos conceitos aprendidos na disciplina de Programação Estruturada à Objetos, ficando faltando apenas a utilização de Padrões de Projeto.

Algumas das funcionalidades pensadas inicialmente e incluídas no documento das especificações do projeto não foram implementadas por questões de tempo, organização e complexidade. Inicialmente havíamos planejado que o cliente pudesse comprar passagens por tipo, havendo a possibilidade de compra na classe econômica e primeira classe e que haveria

um sistema de milhas, onde o cliente acumulava milhas quando comprasse passagens e usaria elas para obter desconto em futuras compras.

Por tratar-se da implementação um sistema de nível acadêmico e para fins de obtenção de conceito na disciplina de Programação Orientada à Objetos, concluímos que os objetivos para tal foram obtidos, porém, caso esse sistema continue a ser desenvolvido após o término da disciplina, pode-se pensar na inclusão de um banco de dados e a conexão do sistema com esse banco, para realizar a persistência das operações realizadas nele durante o seu funcionamento, a implementação da compra de passagens por classe, o sistema de milhas e a compra de mais passagens utilizando uma única conta, além de outras ideias que podem surgir posteriormente.

5. Referências Bibliográficas

GUANABARA, Gustavo. Curso de POO Java (Programação Orientada à Objetos). 2016. Link: https://www.youtube.com/playlist?list=PLHz_AreHm4dkqe2aR0tQK74m8SFe-aGsY. Acesso em: agosto de 2019.

GRONER. Loiane. Curso de Java Básico. Link: https://www.youtube.com/playlist?list=PLGxZ4Rq3BOBq0KXHsp5J3PxyFaBIXVs3r. Acesso em: agosto de 2019.

RAMO, Ricardo Ramos. Título: Treinamento Prático em UML. Subtítulo: Desenvolva e gerencie seus projetos com essa sensacional ferramenta. São Paulo. Digerati Books, 2006.