UNIVERSIDADE da MADEIRA

# 2021/2022
# Programação avançada – Ficha prática 1

**Exercício 1:**

Considere que foi contratado para trabalhar, enquanto Engenheiro Informático, numa companhia aérea. A companhia área já possui uma aplicação de gestão de voos e de passageiros. Todavia, existem algumas melhorias a serem feitas, nomeadamente testes unitários e validação de dados.

A aplicação atualmente implementada especifica dois tipos de passageiros (regular e VIP) e dois tipos de voos (económico e executivo). A lógica de negócio implementada especifica que passageiros VIP podem ser adicionados a qualquer tipo de voo; contudo, passageiros regulares só poderão ser adicionados a voos do tipo económico. Além disso, em caso de *overbooking*, a política da empresa prevê que apenas os passageiros da classe económica é que poderão ser removidos dos voos.

A aplicação atual da empresa encontra-se no GitHub da disciplina. Considere que hoje é o seu primeiro dia de trabalho.

1. Comece por fazer *fork* do repositório para a sua conta GitHub.
2. Explore a aplicação da companhia aérea, no seu computador, usando o IntelliJ IDEA. (Terá de fazer *clone* do seu repositório.)
3. Uma vez que irá fazer alterações na aplicação atual, crie um *branch* e dê um nome à sua escolha. (Não se esqueça de fazer *checkout* para o *branch* agora criado.)
4. Implemente os testes unitários que considere suficientes para testar a aplicação atualmente usada pela companhia aérea. Execute os testes implementados e certifique-se que todos passam. Tente implementar, pelo menos, um teste de cada tipo: `@Test` e `@RepeatedTest`.
5. Execute os testes com *coverage* e verifique se todas as linhas da aplicação foram testadas. Caso alguma linha não tenha sido testada, altere/implemente o(s) teste(s) de modo a testar(em) essa linha.
6. Uma vez que os testes foram desenvolvidos e passaram, faça *commit* da nova versão da aplicação e, de seguida, *push*. (Note que está a trabalhar no *branch* criado na alínea 3.)
7. Um dos gestores da companhia aérea comunicou-lhe que, por engano das agências de viagens, alguns passageiros estão a ser adicionados por mais do que uma vez ao mesmo voo. Comece por desenvolver e executar o teste unitário que teste esta situação. De seguida, corrija a lógica atual da aplicação de modo a evitar que tais erros voltem a acontecer. Execute de novo o teste unitário desenvolvido e certifique-se que o mesmo, desta vez, passa.
8. Uma vez que a correção foi feita e testada, faça *commit* da nova versão da aplicação e, de seguida, *push*. (Note que está a trabalhar no *branch* criado na alínea 3.)
9. Usando o GitHub, faça *pull-request* do *branch* criado na alínea 3 para o *branch* principal e aceite-o. Verifique que o código desenvolvido já se encontra no *branch* principal.

**Exercício 2:**

Considere que foi contratado para trabalhar, enquanto Engenheiro Informático, num banco recém-criado. Os administradores desse banco querem começar a desenvolver, em pequenos passos, o sistema bancário informático que dará suporte a todas as operações do banco.

1. Crie um projeto Maven utilizando o IntelliJ IDEA e crie o ficheiro *.gitignore* adequado. Partilhe o projeto no GitHub. Utilize esse projeto para resolver as alíneas seguintes.
2. A sua primeira tarefa consiste em permitir que um cliente abra uma conta à ordem, e que a consiga movimentá-la (isto é, depositar e retirar dinheiro). Um cliente possui um nome, idade associada, quantidade de dinheiro devida e número de filhos.

Utilizando o processo TDD (*test-driven development*), comece por desenvolver os testes, escrever o código (mais simples possível) que faz os testes passarem, e fazer *refactoring*. Tente implementar, pelo menos, um teste de cada tipo: `@Test`, `@ParameterizedTest` e `@RepeatedTest`. Itere o número de vezes necessário até ficar satisfeito com a arquitetura e funcionalidades da sua aplicação.

3. A sua segunda tarefa consiste em implementar um sistema que verifica se o cliente pode pedir ou não um empréstimo. Se o cliente tiver 18 anos ou mais, 25 % é adicionado à certeza de pagamento de crédito. Se o cliente não tiver dívidas, 45 % é adicionado. Finalmente, se o cliente tiver menos de o que 3 filhos, 30 % é adicionado. Um cliente pode pedir empréstimo se a certeza de pagamento for superior ou igual a 60 %.
   Utilizando o processo TDD, comece por desenvolver os testes, escrever o código (mais simples possível) que faz os testes passarem, e fazer *refactoring*. Tente implementar, pelo menos, um teste de cada tipo: `@Test`, `@ParameterizedTest`. Itere o número de vezes necessário, até ficar satisfeito com a arquitetura e funcionalidades da sua aplicação.
4. Faça *commit* da nova versão da aplicação e, de seguida, *push*.