

ADVENTURE_1

Nesta parte do projeto irá estender as funcionalidades do código em ADVENTURE_0 com dois novos módulos, nomeadamente Car e Tax. À semelhança dos módulos existentes, esses dois módulos devem ser criados utilizando a estrutura de Multiple Module Projects.

Para além de estender as funcionalidades do projeto, nesta parte irá também enriquecer o código em ADVENTURE_0 com testes de unidade (incluindo os novos módulos), usando a biblioteca JUnit. Em particular, deverá enriquecer as baterias de teste fornecidas para os vários módulos por forma a obter uma cobertura de pelo menos **70%** para o módulo Broker e **95%** para os restantes módulos. Repare no ficheiro pom.xml do projeto raiz que a cobertura mínima para que a build passe é, neste momento, de apenas 20%:

```
<coverage.class.ratio>0.2</coverage.class.ratio>
<coverage.instruction.ratio>0.2</coverage.instruction.ratio>
<coverage.method.ratio>0.2</coverage.method.ratio>
<coverage.branch.ratio>0.2</coverage.branch.ratio>
<coverage.complexity.ratio>0.2</coverage.complexity.ratio>
<coverage.line.ratio>0.2</coverage.line.ratio>
```

Não é necessário testar os dataobjects. Pode excluir essas classes incluindo o seguinte nas configurações do Jacoco no ficheiro pom.xml do projeto raiz (logo após os DataFile):

```
<excludes>
  <exclude>**/dataobjects/**</exclude>
  <exclude>**/exception/**</exclude>
</excludes>
```

A título de exemplo, depois de executar **mvn clean package**, é possível verificar no relatório de cobertura produzido pelo Jacoco (disponível em <modulo>/target/site/jacoco/index.html) que o método processPayment da classe Bank do módulo Bank não foi coberto por nenhum teste. Para aumentar a cobertura do módulo Bank, deve então adicionar testes que que exercitem esse método, seguindo uma cobertura baseada na especificação, o que inclui testar, sempre que possível, para os **valores de fronteira**.

Para cada um dos módulos, em caso de erro é lançada uma exceção customizada (por exemplo, HotelException). No módulo Broker, como orquestrador das funcionalidades entre os diversos módulos, podem acontecer ainda falhas na comunicação com os módulos que orquestra. Neste caso a exceção RemoteAccessException é lançada. Ainda no módulo Broker, recomenda-se que se analise com especial atenção a funcionalidade nas classes que estendem a classe AdventureState, responsável pela gestão do estado duma reserva. Nestas classes está incluído o comportamento quando ocorrem falhas na rede, RemoteAccessException, situação que poderá ocorrer uma vez que a comunicação entre o Broker e os restantes módulos seja distribuída.

Na implementação dos módulos Car e Tax deve seguir a mesma estrutura com que foram implementados os restantes módulos, pelo que aconselhamos a sua consulta primeiro. Tenha em atenção as novas dependências que surgem entre os módulos, dado que todos os módulos passam a depender do módulo Tax, ainda que atualmente não se esteja a pedir para implementar nenhum tipo de pedido ao módulo Tax.

Car

O módulo Car implementa todas as funcionalidades relativas ao aluguer de viaturas. As classes encontram-se descritas no diagrama de domínio disponibilizado no enunciado geral. Deve adotar uma abordagem de desenvolvimento guiado por testes (TDD) – i.e., começar por escrever os casos de teste e de seguida implementar o código. Os seguintes métodos devem ser testados:

- RentACar(name)
- RentACar.getRenting(reference)
- RentACar.getAllAvailableCars(begin, end)
- RentACar.getAllAvailableMotorcycles(begin, end)
- RentACar.getRentingData(reference) -> RentingData
- Car(plate, kilometers, rentACar)
- Motorcycle(plate, kilometers, rentACar)
- Vehicle.isFree(begin, end)
- Vehicle.rent(drivingLicense, begin, end)
- Renting.conflict(begin, end)
- Renting.checkout(kilometers)

RentingData consiste numa classe de data object com o seguinte construtor:

- RentingData(String reference, String plate, String drivingLicense, String rentACarCode, LocalDate begin, LocalDate end)

Os testes deverão cobrir também a integridade dos dados:

- Matrículas não podem ter valores nulos nem duplicados e têm o formato "XX-XX-XX".
- A reference de cada Renting é única.
- O número da carta de condução é composto por número ordinal precedido dos dígitos alfabéticos identificadores do serviço emissor da carta.
- Número de quilómetros de um veículo tem sempre um valor não negativo.
- Nome do RentACar não pode ser nulo nem vazio.
- A cada veículo está associado obrigatoriamente um RentACar.
- Não pode haver sobreposição de reservas num mesmo veículo.

Tax

O módulo Tax é formado pelas classes descritas no enunciado geral. Deve começar por codificar os casos de teste e de seguida implementar o código que faz com que eles passem. Os métodos a testar são:

- Buyer(NIF, NAME, ADDRESS)
- Buyer.taxReturn(YEAR)
- Invoice(VALUE, DATE, ITEM_TYPE, SELLER, BUYER)
- IRS.getItemTypeByName(NAME)
- IRS.getTaxPayerByNIF(BUYER_NIF)
- IRS.submitInvoice(invoiceData)
- ItemType(ITEM_TYPE, TAX)
- Seller(NIF, NAME, ADDRESS)
- Seller.toPay(YEAR)
- Seller(TaxPayer).getInvoiceByReference(INVOICE_REFERENCE)

Em que invoiceData é de uma uma instância de um data object InvoiceData com construtor InvoiceData(String sellerNIF, String buyerNIF, String itemType, float value, LocalDate date)

Adicionalmente deve considerar os seguintes invariantes:

- Os NIFs possuem 9 dígitos e são únicos.
- Os ITEM_TYPE são string e únicos.
- Os INVOICE_REFERENCE são string e únicos.
- NAME, ADDRESS, VALUE, DATE, ITEM_TYPE, SELLER, BUYER, TAX, YEAR e INVOICE_REFERENCE não podem ser nulos nem vazios, no caso de serem do tipo String.
- TAX não pode ser negativo
- YEAR não pode ser anterior a 1970.
- DATE não pode ser anterior a 1970.
- O valor do IVA é igual à percentagem do valor de tax em ItemType. Note que se segue o modelo em que o valor do produto não contém o IVA, sendo este adicionado no ato da compra, quando o Invoice é criado.
- O valor de retorno do IVA é igual a 5% do valor de IVA cobrado ao produto comprado.




Cada grupo deve dividir-se em sub-grupos de 3 alunos, em que cada um deles, trabalha num dos novos módulo (Car ou Tax) incluindo as suítes de teste. Um dos sub-grupos deve trabalhar no módulo broker, e o outro nos restantes módulos. **É obrigatório indicar no README.md quais os elementos que trabalham em cada um dos módulos, indicando o número de aluno, o nome e o username no GitHub.**

Para a entrega deverão fazer após o último commit:

```
$ git checkout -b first-deliver
$ git tag ADVENTURE_1
$ git push origin --tags first-deliver:first-deliver
```

Esta tag colocada deve ter uma data anterior à data limite de entrega, dia **18 de Março pelas 20:00**.

Durante os laboratórios, os alunos dos grupos serão avaliados com base no trabalho desenvolvido durante cada uma das semanas. Para isso é necessária a criação de um sprint no GitHub, usando a interface de Project, em que cada tarefa deve ser representada por um Note. Neste caso o projeto a criar deve-se chamar Sprint One. Uma vez criado o projeto, escolhendo como template pré-definido "Kanban (automatic)", deve ser alterado o nome da coluna To Do para Backlog. Nesta coluna serão colocadas todas as tarefas a realizar, as quais serão definidas durante a reunião do grupo. Cada tarefa é criada como uma Note, contendo a descrição da tarefa a realizar, e deve ser convertida num Issue, por forma a que os commits possam ser-lhe associados. Cada aluno que vá realizar uma tarefa move o respetivo note da coluna de Backlog para a coluna In progress, e atribui-se essa tarefa no Issue respetivo. Quando a tarefa é terminada, o commit deve ser associado ao respetivo Issue, passando este para o estado fechado. Para se associar um commit a um Issue deve colocar-se na mensagem de commit closes #n, em que n é o número do Issue. Quando o Issue é fechado o respetivo Note é automaticamente passado da coluna In progress para a coluna Done. **Cada commit deve estar associado a uma única tarefa.**

Página Inicial	
Grupos	
Avaliação	
Bibliografia	
Horário	
Métodos de Avaliação	
Objectivos	
Planeamento	
Programa	
Turnos	
Anúncios	
Sumários	
Notas	
Resultado dos QUC	
Teóricas	
Laboratórios	
Horário de Dúvidas	
Documentação	
Projeto	

ADVENTURE_1
FAQ
ADVENTURE_2
ADVENTURE_3
ADVENTURE_4
ADVENTURE_5
Extra-Mile (Prémio Novabase)
Época Especial
Notas intercalares