LEI_ESII2021_TP1_GRUPOU

TEST DESIGN SPECIFICATION

Version 1.0

07/12/2020

Histórico de Versões

Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
1.0	*8160207 – Nuno Matos;* *8190305 – Rafael Dias;*	*07/12/2020*	*8160207 – Nuno Matos;* *8190305 – Rafael Dias;*	*07/12/2020*	First and final draft;

Tabela de Conteúdos

1. Introdução

1.1. Identificador do documento

TCS_DOC

1.2. Âmbito

Todos os casos de uso, testes, e a devida documentação foram realizados para projeto PackingAPI, em prol da unidade curricular de Engenharia de Software II, licenciatura de Engenharia Informática.

1.3. Referências

Ao longo deste documento aparece uma tabela de "Features" que contém o identificador dos casos de uso que são abordados. Estes mesmo podem ser consultados no documento intitulado "Casos de Uso".

1.4 Glossário

Exemplo – palavras sublinhadas correspondem a instâncias de classes;

Exemplo – palavras em itálico correspondem a variáveis de instância;

Exemplo() – palavras em itálico seguidas de parênteses correspondem a métodos de classes;

2. Features/Itens a testar

Item a testar	Descrição	Casos de Uso	Responsabilidade
Método *setX()* de Position	Alterar o valor da coordenada X de uma Position.	UC0001	8190305 - Rafael Dias
Método *setY()* de Position	Alterar o valor da coordenada Y de uma Position.	UC0002	8190305 - Rafael Dias
Método *setZ()* de Position	Alterar o valor da coordenada Z de uma Position.	UC0003	8190305 - Rafael Dias
Método *addItem()* de Container	Adicionar um ItemPacked a um Container já existente.	UC0004	8190305 - Rafael Dias
Método *removeItem()* de Container	Remover um ItemPacked específico de dentro de um Container se este se encontrar lá inserido.	UC0005	8190305 - Rafael Dias
Método *close()* de Container	Fechar um Container já existente.	UC0006	8190305 - Rafael Dias
Método *getItem()* de Container	Através da sua referência, obter uma instância de um Item.	UC0007	8190305 - Rafael Dias
Método *validate()* de Container	Verificar se um Container se encontra devidamente estruturado.	UC0008	8190305 - Rafael Dias
Método *addContainer()* de ShippingOrder	Adicionar um novo Container válido a uma ShippingOrder já existente.	UC0009	8160207 - Nuno Matos
Método *existsContainer()* de ShippingOrder	Verificar se um Container específico já se encontra dentro de uma ShippingOrder ou não.	UC0010	8160207 - Nuno Matos
Método *findContainers()* de ShippingOrder	Usando a sua referência, procurar um Container específico numa ShippingOrder, e, se encontrado, devolver o número correspondente à sua posição na ShippingOrder.	UC0011	8160207 - Nuno Matos
Método *validate()* de ShippingOrder	Verificar se todos os Containers dentro de uma ShippingOrder são válidos ou não.	UC0012	8160207 - Nuno Matos
Método *removeContainer()* de ShippingOrder	Remover um Container específico de dentro de uma ShippingOrder se este se encontrar lá inserido.	UC0013	8160207 - Nuno Matos
Método *setStatus()* de ShippingOrder	Alterar o estado de uma ShippingOrder existente.	UC0014	8160207 - Nuno Matos
Método *validate()* de PackingGUI	Verificar se um ficheiro JSON correspondente a uma ShippingOrder se encontra devidamente formado ou não.	UC0015	8160207-Nuno Matos

3. Detalhes da abordagem aos testes

Os testes foram efetuados sobre o código foram criados tendo em consideração os inputs e os outputs esperados. Antes da sua elaboração, foi analisado o ficheiro "test.json", de onde foram retirados uma série de inputs válidos para a diferentes elementos. Depois realizamos uma análise, através da aplicação NetBeans e IntelliJ Idea, do ficheiro main.java, que correspondia a um exemplo de uma utilização da libraria que iriamos testar. Através deste ficheiro podemos verificar como se encontravam estruturados os métodos construtores de cada classe. É de salientar que verificamos que a classe Position não tinha o método construtor devidamente estruturado. Na documentação da classe indica-se que o construtor está feito de modo a aceitar as variáveis da seguinte forma: (x, y, z). Porém, o que realmente acontece é que a variável x se encontra no ultimo parâmetro, e os dois primeiros parâmetros são irrelevantes para a inicialização das outras duas variáveis, pois independentemente dos valores lá colocados, as instâncias de Position começam sempre com valores de Y e Z a 0.

Em seguida foi analisado o Javadoc a nós proporcionado. Verificamos os inputs, outputs e exceções de cada método das classes lá presentes, e escolhemos quais deveríamos testar. Os métodos *gets()* e *sets()* que não atiravam exceções, nem justificavam os testes dos inputs/outputs foram colocados de parte, e os restantes métodos foram adicionados á lista dos quais iriamos realizar testes.

Em todos os métodos é utilizado o método **Equivalence Class Partitioning** para realizar os devidos testes. Utilizamos vários inputs diferentes consoante o número de outputs possíveis, quer estes sejam válidos ou inválidos.

Utilizamos também o método **Boundary Value Analysis** nos seguintes métodos: *setX()*, *setY()*, *setZ()* de Position; *addItem()*, *close()* e *validate()* de Container. No caso dos *setX()*, *setY()* e *setZ()* verificamos se os valores que se encontram no limite ou fora do limite mínimo devolvam os outputs corretos. Para *addItem()*, *close()* e *validate()* é verificado se os Items que se encontram em posições válidas ou inválidas devolvem os outputs corretos.

4. Identificação dos Testes

Para cada método foram definidos os testes para casos válidos e caso inválidos, avaliando se retornam o esperado. Nos casos válidos verifica-se o fluxo normal do método, já nos casos inválidos analiza-se o fluxo alternativo dos métodos.

As restantes especificações dos mesmos encontram-se delineados no documento **TCO_DOC**.

5. Critérios de passagem ou falha das features

Os testes são considerados como passados quando, consoante o input, devolvem o output esperado, realizam as alterações esperadas, atiram as exceções corretas, ou simplesmente acabam sem serem atiradas exceções. Os testes são considerados falhados caso devolvam outputs incorretos, ou diferentes do esperado, não atirem as exceções quando devido, ou atirem exceções inesperadas.