IPS Instituto Politécnico de Setúbal
Escola Superior de Tecnologia de Setúbal

PA 2021/22

Programação Avançada

Enunciado de Laboratório

7

Padrões de Software – Memento

Objetivos:

- Aplicação do padrão de software Memento.
- Desenvolvimento de testes unitários.

Introdução:

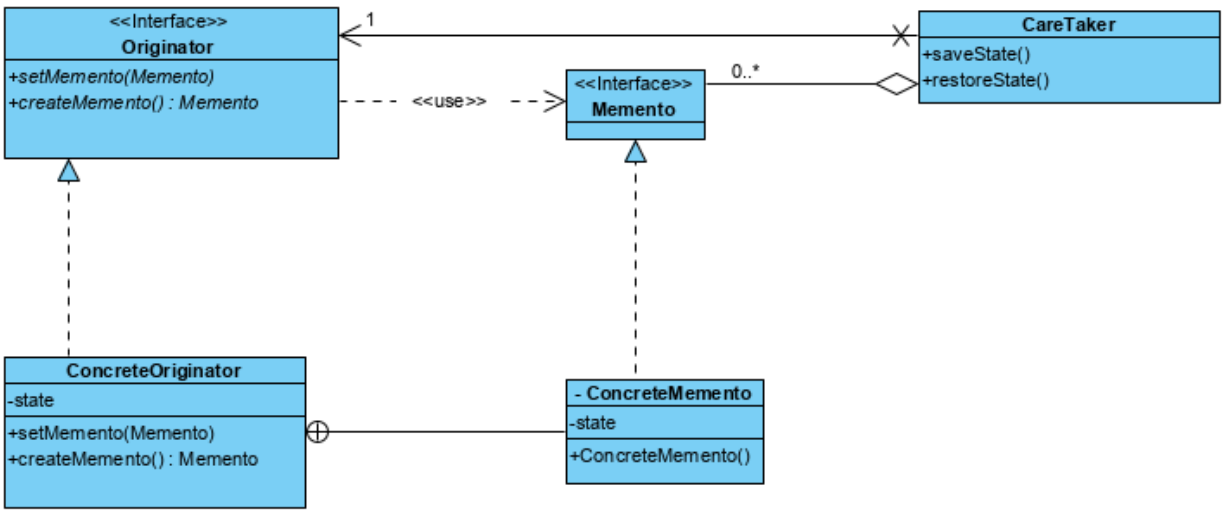
O padrão Memento permite guardar o estado atual de um objeto para que possa ser recuperado após alterações, i.e., *undo*.

Neste laboratório pretende-se oferecer suporte para a operação undo num carrinho de compras. De forma a que se possam adicionar produtos ao carrinho e fazer undo.

Comece por clonar o projeto existente no GitHub no seguinte endereço:
https://github.com/estsetubal-pa-geral/Lab7Template.git

O padrão determina que existe um objeto **`Originator`** com um estado que será guardado através de um objeto **`Memento`**, gerido por um **`CareTaker`**. A responsabilidade do **`CareTaker`** é simplesmente guardar estados do **`Originator`**, de forma a que os possa recuperar.

Para promover o encapsulamento na utilização do padrão, propõe-se uma versão adaptada do padrão original **`Memento`** que define interfaces **`Originator`** e **`Memento`**, implementando o estado (**`ShoppingCartMemento`**) como classe privada interna à instância de **`Originator`**(ShoppingCart).



Nesta implementação do padrão teremos as seguintes correspondências:

- `Concrete Originator – ShoppingCart`
- `ConcreteMemento - ShoppingCartMemento`

Nível I

1. Comece por executar o programa, familiarizando-se com a GUI e as funcionalidades disponibilizadas. Repare que o botão Undo não está a funcionar.
2. Altere a classe **`ShoppingCart`** para que esta passe a implementar a interface **`Originator`**.
3. Implemente a inner classe **`ShoppingCartMemento`**, que implementa a interface Memento
   - Nota: Esta classe deverá ter como atributo a **`List<Product> produtos.`**
4. Implemente os métodos:
   - **`public void setMemento(Memento savedState)`**
   - **`public Memento createMemento()`**

Nível II

1. Crie a classe Caretaker, que terá dois atributos:
   - **`Memento memento`**
   - **`Originator originator`**

   Crie um construtor que coloque o **`Memento`** a null e inicialize o atributo **`Originator`** com o valor passado por parâmetro.
2. Implemente o método **`void saveState()`** deve obter e guardar o Memento atual.
3. Implemente o método **`void restoreState()`** deve reestabelecer o estado do **`originator`** com base no **`Memento`** guardado. Se não existir **`Memento`** guardado deve lançar a exceção **`NoMementoException`**.

Nível III

1. Adicione um atributo **`Caretaker`** ao **`ShoppingCartController`**.
2. Modifique o método **`void addProduct(String name, double cost)`** para que passe a guardar o estado anterior a cada modificação do carrinho de compras.
3. Modifique o método **`void reset()`** para que salve o estado antes de limpar o carrinho.
4. Modifique o método **`void removeProduct(String name)`** para que salve o estado antes de remover um produto.
5. Implemente o método **`void undo()`** que executa a operação undo que utiliza a instância **`caretaker`** para recuperar o estado anterior do carrinho de compras.

6. Modifique a classe **`ShoppingCartPannel`** de forma a adicionar o comportamento do botão buttonUndo de forma a permitir um undo após inserção de um produto na lista de compras.

Nível IV

1. Adicione o botão **`buttonReset`** na classe **`ShoppingCartPannel`** e adicione-o ao lado do botão **`Undo`**.
2. Adicionar o comportamento do botão **`buttonReset`** de forma a permitir um reset à lista de compras.
3. Verifique se a funcionalidade undo continua a funcionar como o esperado.

Nível V

Altere a classe **`Caretaker`** de forma a implementar um multi-undo.

1. Substitua o atributo **`Memento memento`**, por **`Stack<Memento> mementos`**.
2. Altere o construtor para inicializar a stack.
3. Altere os métodos **`saveState`** e **`restoreState`**, para passarem a manipular uma stack de mementos e não só um memento.
4. Corra o **`main`** e verifique que tudo corre como planeado.