Lab XI.

Objetivos

Os objetivos deste trabalho são:

- Utilizar padrões de comportamento (i.e., *State, Strategy, Template Method, Visitor*) para resolver casos práticos.
- Aplicar boas práticas de programação por padrões
- Rever todos os padrões

Nota: Para além do código no github, inclua também um ficheiro PDF ou PNG com o diagrama de classes da solução final.

XI.1

Uma revista eletrónica quer fornecer aos seus clientes um conjunto de propriedades sobre diferentes telemóveis, como por exemplo, processador, preço, memória, câmara, etc. Os resultados devem ser apresentados numa lista, ordenada por qualquer um dos atributos. Por outro lado, existem vários algoritmos de ordenação com diferentes desempenhos, relativamente ao tempo de processamento e ao espaço ocupado. Assim, é necessário podermos selecionar facilmente o melhor algoritmo (por exemplo, em tempo de execução).

a) Que padrão (padrões?) pode ser aplicado para cumprir estes requisitos?
b) Desenhe um diagrama de classes para responder a este problema.
c) Construa o código necessário para demonstrar o princípio. Inclua 3 algoritmos de ordenação distintos (não é relevante para este exemplo a sua implementação).

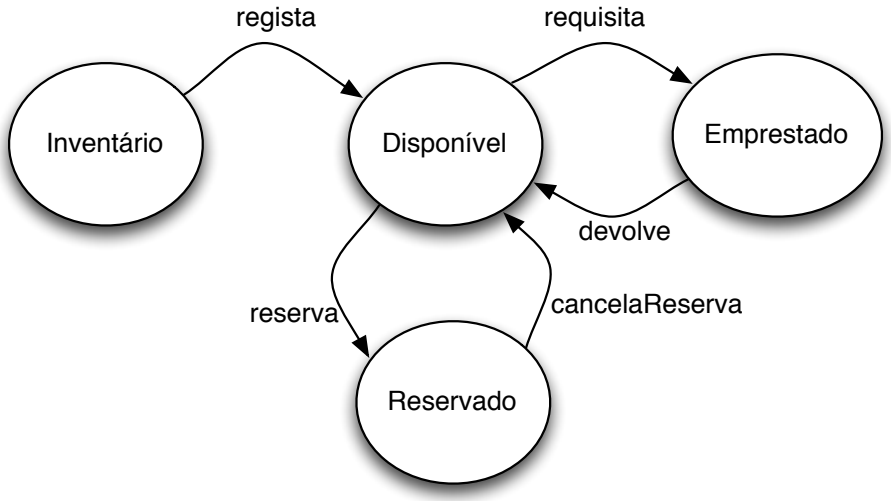
XI.2

O padrão *Template Method* é utilizado em várias classes da Java API. Por exemplo, nas classes abstratas *java.io.InputStream*, *java.io.OutputStream*, *java.io.Reader*, *java.io.Writer*, *java.util.AbstractList*, *java.util.AbstractSet* e *java.util.AbstractMap*.
Selecione 3 destas classes, analise o código fonte (e.g., em *http://openjdk.java.net*) e identifique todas as situações em que o padrão ocorre. Descreva as suas conclusões no ficheiro *Lab11_2.txt*.

XI.3

Numa biblioteca cada livro é caracterizado por um *título*, *ISBN*, *ano*, e o primeiro *autor*. A entidade livro permite operações tais como: regista, requisita, reserva, cancelaReserva, disponível, etc.). No entanto, cada uma destas operações depende da situação do livro na biblioteca: se está em situação de inventário, por exemplo, só permite a operação regista.

a) Considerando o diagrama de estados (círculos) e operações (setas) representados na figura seguinte, construa uma solução que represente adequadamente este problema. *Nota: os vários estados poderão não representar fielmente uma situação real.*



b) Teste a solução criando uma lista de 3 livros e permitindo ao funcionário da biblioteca simular a iteração com a lista da seguinte forma (*o texto escrito pelo utilizador está marcado a azul*):

```
*** Biblioteca ***
1    Java Anti-Stress       Omodionah      [Inventário]
2    A Guerra dos Padrões   Jorge Omel     [Inventário]
3    A Procura da Luz       Khumatkli      [Inventário]
>> <livro>, <operação: (1)regista; (2)requisita; (3)devolve; (4)reserva; (5)cancela

>> 1,1
*** Biblioteca ***
1    Java Anti-Stress       Omodionah      [Disponível]
2    A Guerra dos Padrões   Jorge Omel     [Inventário]
3    A Procura da Luz       Khumatkli      [Inventário]
>> <livro>, <operação: (1)regista; (2)requisita; (3)devolve; (4)reserva; (5)cancela

>> 1,3
Operação não disponível

>> 1,2
*** Biblioteca ***
1    Java Anti-Stress       Omodionah      [Emprestado]
2    A Guerra dos Padrões   Jorge Omel     [Inventário]
3    A Procura da Luz       Khumatkli      [Inventário]
>> <livro>, <operação: (1)regista; (2)requisita; (3)devolve; (4)reserva; (5)cancela

>> 3,1
*** Biblioteca ***
1    Java Anti-Stress       Omodionah      [Emprestado]
2    A Guerra dos Padrões   Jorge Omel     [Inventário]
3    A Procura da Luz       Khumatkli      [Disponível]
>> <livro>, <operação: (1)regista; (2)requisita; (3)devolve; (4)reserva; (5)cancela

>> ...
```

XI.4

Estude a estrutura e a finalidade de todos os padrões que foram apresentados em PDS. Responda ao seguinte questionário em linha, fazendo uma gestão da quantidade de acertos e falhas:

http://www.vincehuston.org/dp/patterns_quiz.html

XI.5

Um exemplo do padrão *Visitor* em Java visa mediar a interação com a estrutura de diretórios do sistema de ficheiros. Consulte a documentação do Java para perceber como usar as seguintes classes (*java.nio.file.FileVisitor*, *java.nio.file.SimpleFileVisitior*) e o método *java.nio.files.Files.walkFileTree*. Eles permitem aceder ao sistema de ficheiros utilizando o padrão *Visitor*.

a) Utilize o *grepcode.com* ou o *docjar.com* para ficar a conhecer a implementação destas classes.

b) Desenvolva um programa conceptual que devolva o tamanho de um diretório, i.e. a soma dos tamanhos de todos os ficheiros. Adicione a opção –r (recursiva), assim o programa contará com o tamanho dos subdiretórios dentro do diretório raiz.

```
$ java –jar sizeOf.jar root
A: 10 kB
C: 8  kB
Total: 18 kB
==================================
$ java –jar sizeOf.jar –r root
A: 10  kB
B: 100 kB
|->Test.file: 100kB
C: 8   kB
Total: 118 kB
```