Prática 4 - Cartas

Para iniciarmos, deverás fazer download do ficheiro .zip disponível nesta atividade.

Deverás modificar a classe `Molho` para que possa ordenar-se segundo os critérios distintos e algoritmos de ordenação.

Para isso:

- Cria uma classe abstrata AlgoritmoOrdenação que inclua um método abstrato `ordenar(List listaCartas)` que devolve a lista de cartas passadas por parâmetro ordenadas.

- Cria subclasses de `AlgorimoOrdenação` que ordenam as cartas por critérios diferentes:
  - Ordenação naipe-número-incremental: ordenar as cartas primeiro por naipe (por ordem alfabética de PAUS, COPAS, ESPADAS, OUROS) e logo por número (do menor ao maior).
  - Ordenação naipe-número-decremental: ordenar as cartas primeiro por naipe (por ordem alfabética de PAUS, COPAS, ESPADAS, OUROS) e logo por número (do maior ao menor).
  - Ordenação número-naipe. Ordenar as cartas primeiro por número (incremental) e logo por naipe (ordem alfabética).

- Pode-se procurar algoritmos de ordenação em: https://pt.wikipedia.org/wiki/Algoritmo_de_ordena%C3%A7%C3%A3o e irá precisar de especificar o critério de ordenação de cartas.
- Permite que se possa indicar o algoritmo de ordenação ao molho de cartas mediante um método setAlgoritmo.
- Ordenar o molho de cartas mediante o algoritmo de ordenação que foi indicado, o molho não sabe realmente como se está a ordenar.

Por exemplo: Nos ficheiros que fizeste download, já está criada a maior parte da estrutura do código, apenas há que introduzir o código correspondente à ordenação. Ou seja, os métodos setAlgoritmo e ordenação de molho e as classes OrdNaiNumDec, OrdNaiNumInc e OrdNaiIncPal. Em cada método de ordenação será útil criar um comparador de cartas utilizado nesse algoritmo.

No main é mostrado como se cria um molho e se ordena de várias formas distintas.