Exercício 1

Altere a sua implementação da classe: `ContainerOfObjects` de forma a incluir um método que suporte a ordenação do vetor. Para suportar o processo, deverá implementar uma interface com o nome: `Comparator` que defina um método `compareTo` com a seguinte assinatura:

```
public int compareTo(Comparator obj);
```

O método deverá retornar a seguinte gama de valores:

- >0 se o objeto a comparar for maior que o objeto recebido;
- <0 se o objeto a comparar for menor que o objeto recebido;
- 0 se o objeto a comparar for igual ao objeto recebido;

Deverá realizar as alterações necessárias para que a implementação da interface permita a ordenação decrescente por número de dias dos eventos contidos na rota de eventos da ficha prática anterior. Para isso, a classe Evento deverá implementar a interface `Comparator` e a classe `ContainerOfObjects` deverá implementar um método de ordenação de forma a tirar partido o método `compareTo`;
O método deverá ainda identificar as situações em que o objeto recebido não é uma instância de `Comparator` e quando o vetor não tem elementos armazenados, utilizando para isso excepções específicas (por exemplo: `NotAnInstanceOfComparable` *exception* e `EmptyArray` *exception*.

Exercicio 2

Complemente a sua classe: `ContainerOfObjects` de forma a realizar a gestão de eventos de uma rota de eventos (`RotaEventos`). Implemente a interface `GestorEventos` (Figura 1) e uma classe: `GestaoEventos` que deverá ser uma subclasse de `ContainerOfObjects` e implementar a interface `GestorEventos` (Figura 1).

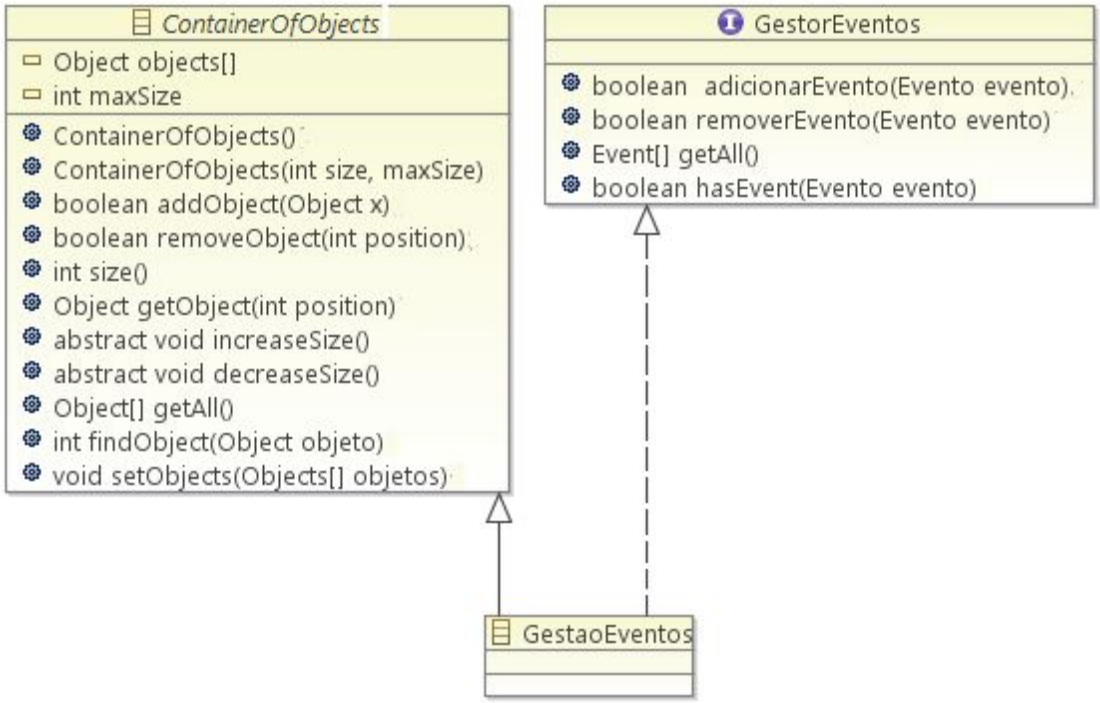

Figura 1

A classe `ContainerOfObjects` contém um array de objetos que poderá ser inicializado sem elementos ou com um número pré-definido de elementos. Adicionalmente poderá ser definido um **limite máximo** de elementos que o array deverá suportar.

Exercício 3

Altere a sua classe relativa à representação de Feiras Gastronómicas desenvolvida na ficha prática anterior de forma a representar o tipo de cozinhas (Vegetariana, Alentejana, Sobremesas, etc) que cada evento pode suportar. De forma a testar as classes implementadas, realize o povoamento das cozinhas que cada evento pode suportar tendo por base os dados disponibilizados pela API disponibilizada pela Zomato (https://developers.zomato.com/?lang=pt). Particularmente, deverá utilizar o serviço /categories de forma a retornar todas as cozinhas disponíveis numa cidade à sua escolha. Para realizar o pedido ao serviço utilize a API `PPwebServiceManager` disponibilizada como suporte à ficha prática. Os dados retornados pelos serviços encontram-se estruturados em formato JSON (JavaScript Object Notation) - http://www.json.org. Poderá utilizar uma API à sua escolha para o processamento de objetos JSON.

Exercício 4

Na **inserção de um elemento** poderá ser necessário a "**alocação" de memória** para a adição desse elemento. Da mesma forma terá de ser **"libertada" memória caso um elemento seja eliminado**. Para a alocação/desalocação de memória deverá instanciar um novo vetor com um **tamanho apropriado à operação a realizar** (adição ou remoção) e copiar os elementos do anterior vetor para o novo vetor que reflete o novo número de objetos existente. Para isso deverá implementar os métodos abstratos: `increaseSize` e `decreaseSize` na classe ContainerOfObjects, de forma a respectivamente aumentar e diminuir a memória do array.

Teste o código implementado na classe `EventoDemo`.

Para a realização deste exercício deverá recorrer à utilização de excepções JAVA para a implementação dos métodos de seguida referidos:

- A implementação do método `addObject` (classe `ContainerOfObjects`) apenas considera a adição de elementos no array, não considerando a necessidade de alocação de memória. Deverá codificar no método `adicionarEvento` o código necessário para permitir captar as situações em que o vetor não tem espaço para a adição de novos elementos - `ArrayIndexOutOfBoundsException`. No caso dessa excepção ocorrer deverá invocar o método `increaseSize` de forma a aumentar o tamanho do array e de seguida proporcionar a inserção do novo elemento.

- De seguida, implemente o código necessário para que o método `adicionarEvento` (classe `GestaoEventos`) apresente uma mensagem ao utilizador, antes do elemento ser inserido no array. Por exemplo: *O Evento "XPTO" está a ser inserido no vetor*, em que **XPTO** é a representação natural em String do evento a ser adicionado.

Teste o código implementado na classe `EventoDemo` de modo a que consiga provocar todas as exceptions que poderão surgir.

Exercício 5

Ainda no método `adicionarEvento` (classe `GestaoEventos` e interface `GestorEventos`), crie uma excepção: `EventoJaExisteException` que deverá ser lançada nesse mesmo método, **caso o evento a inserir já exista no array**. No entanto, o tratamento da excepção deverá ser realizado na classe de teste (`EventoDemo`). Teste o método `getStackTrace` herdado da classe `Throwable`.

Teste o código implementado na classe `EventoDemo`.

Exercício 6

A implementação do método `removeObject` da classe `containerOfObjects` apenas contempla a remoção de objetos com base numa determinada posição deste no array, não realizando verificações sobre a existência do objecto ou sobre a sua instanciação. Se estas operações forem implementadas no método `removerEvento`, `aquando` da **remoção de um elemento,** poderão ocorrer dois tipos de Exception's, uma para o caso de o não se encontrar o elemento que se procura (`EventoNaoExisteException`) e outra para o caso de o evento recebido ser nulo (`EventoNuloException`).

Crie novas classes para estas `Exception's` que podem ocorrer, de modo a que seja possível utilizar a interface `GestorEventos` que lhe é fornecida. Estas novas `Exception's` devem ser lançadas mas não devem ser tratadas no método `removerEvento` da classe `GestaoEventos`.

Teste o código implementado na classe `EventoDemo` de modo a que consiga provocar todas as exceções que poderão surgir.