Programação Orientada pelos Objectos

1º Teste (5/Maio/2021)

MIEI 2020/2021

Duração: 1h45

Instruções:

- Antes de começar a resolver, **leia o enunciado do princípio até ao fim**.
  - As interfaces e classes dos grupos I e II têm mais métodos do que os que deverá implementar na resolução deste teste. Em cada grupo, tenha o cuidado de **ver com muita atenção quais os métodos que deve implementar**, para **não desperdiçar o seu tempo a implementar métodos que não lhe são pedidos**.
  - Disponibilizamos a descrição sumária de todos os métodos, incluindo os que não tem de implementar, para que os possa usar na sua resolução.
- Pode usar caneta ou lápis.
- Não é permitido consultar quaisquer elementos para além deste enunciado.

Nos grupos I e II terá que implementar parcialmente algumas classes necessárias à construção de uma aplicação de gestão de leilões (e.g. OLX, eBay). Nesta aplicação existem dois tipos de leilões: *standard* e com *plafond*. No leilão *standard* o vendedor tem que explicitamente fechar o leilão, enquanto no leilão com *plafond* é definido um valor de plafond (valor pretendido pelo produto) e o leilão fecha quando é recebida uma proposta com valor superior ao plafond. Neste tipo de leilões (com plafond) o vendedor pode decidir terminar o leilão antes do plafond ter sido atingido, vencendo assim a melhor oferta recebida – em caso de empate é escolhida a oferta mais antiga. Na descrição do Grupo I são dados mais detalhes sobre o funcionamento de ambos os leilões.

Note que apenas é permitido acrescentar métodos privados às classes (não pode alterar ou acrescentar variáveis, nem métodos não privados).

Grupo I

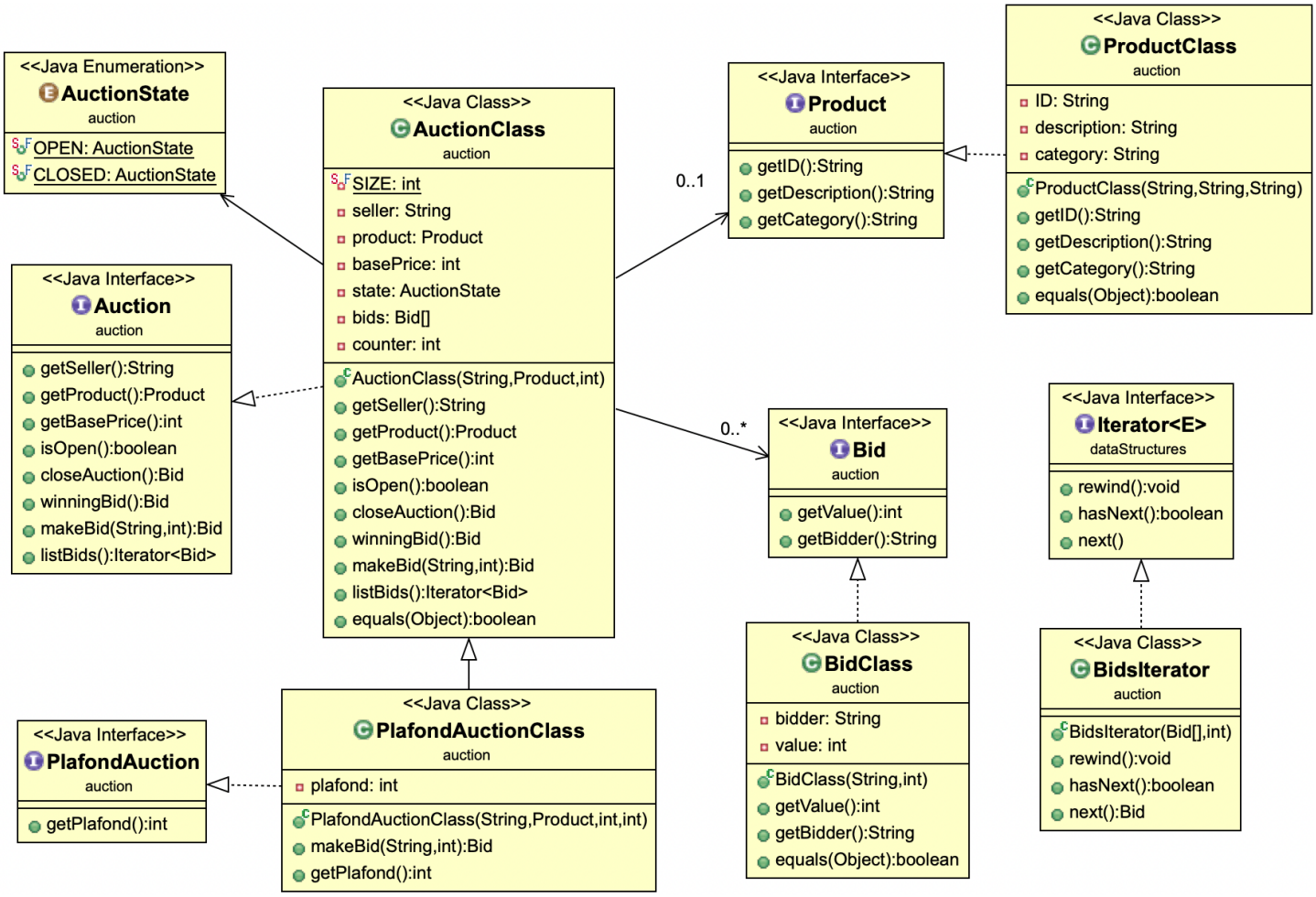

Figura 1 – Leilões standard e com *plafond*.

A interface `Auction` representa um leilão standard (Fig. 1). Esta interface tem os seguintes métodos:

- `getSeller(), getProduct(),getBasePrice()` devolvem, respectivamente, o endereço de email do vendedor, o produto a ser leiloado e o preço base de produto;
- `isOpen()` devolve `true` caso o leilão ainda esteja aberto e `false` caso contrário;
- `closeAuction()` muda o leilão para o estado fechado e devolve a licitação vencedora. Caso não exista nenhuma licitação ou todas as licitações sejam inferiores ao preço base é devolvido `null`. Se o leilão já estiver fechado o método apenas devolve a licitação vencedora.
- `winningBid()` devolve a licitação vencedora caso o leilão esteja fechado. A licitação vencedora é a maior licitação superior ao preço base, em caso de empate devolve a licitação mais antiga. Caso não exista nenhuma licitação ou todas as licitações sejam inferiores ao preço base é devolvido `null`. Se o leilão ainda estiver aberto devolve `null`;
- `makeBid(String buyer, int amount)` regista uma licitação no leilão dado o endereço de email do (potencial) comprador e o valor de licitação. Esta operação só será executada se a pré-condição `isOpen()` for verdadeira. O construtor da `BidClass` recebe como argumentos o comprador (`buyer`) e o valor de licitação (`amount`). Pode assumir que o valor de `counter` é menor do que `SIZE`;
- `listBids()` devolve um iterador para todos as licitações recebidas.

A classe `AuctionClass` implementa a interface `Auction`. Nesta classe temos um construtor que recebe o endereço de email do vendedor, o produto e o preço base. O construtor inicializa todas as variáveis de instância, sendo que inicialmente não existem licitações e o estado do leilão é aberto. A constante `SIZE` define a dimensão inicial do vector `bids`. Os métodos desta classe têm o comportamento já descrito na explicação dada sobre a interface `Auction`. Esta classe implementa também o método `equals` que devolve `true` caso o objecto recebido como argumento tenha o mesmo vendedor e o mesmo produto.

A classe `PlafondAuctionClass` representa um leilão com plafond (Fig. 1) já descrito anteriormente. Esta classe implementa a interface `PlafondAuction` e estende a classe `AuctionClass`. Esta subclasse inclui o novo método `getPlafond` que devolve o plafond do leilão e redefine o método `makeBid` para que o leilão seja fechado se for recebida uma proposta com valor superior ao plafond. O construtor da classe `PlafondAuctionClass`, para além dos parâmetros da superclasse, recebe o plafond do leilão.

Implemente os seguintes métodos das classes `AuctionClass` e `PlafondAuctionClass`.

a) O constructor `PlafondAuctionClass(String seller, Product product, int basePrice, int plafond)`.

b) O método `Bid winningBid()` da classe `AuctionClass`.

c) O método `Bid closeAuction()` da classe `AuctionClass`.

d) O método `Bid makeBid(String buyer, int amount)` da classe `PlafondAuctionClass`.

Tenha em atenção que as variáveis de instância da classe `AuctionClass` são privadas.

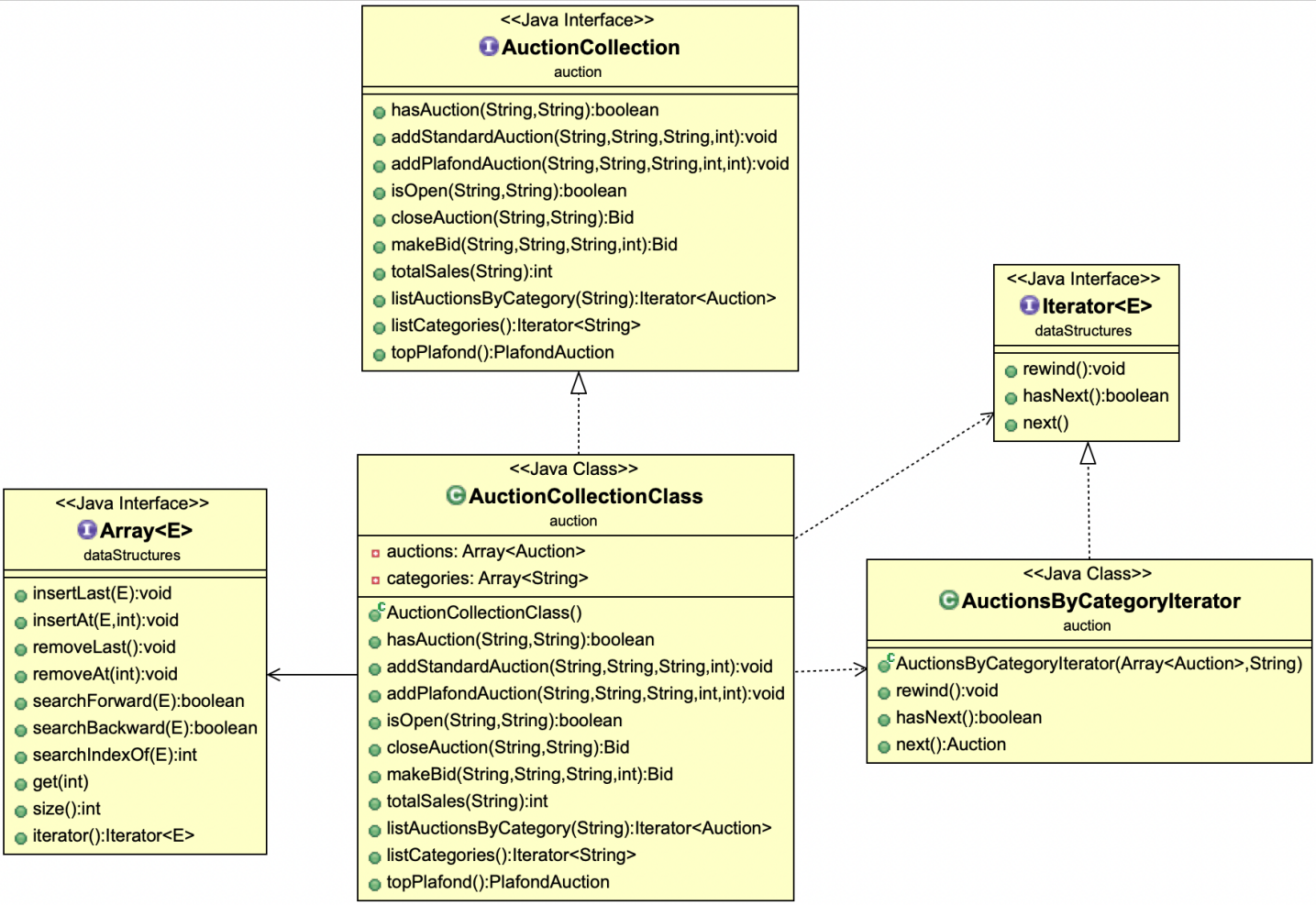

Figura 2 – Aplicação para gestão de leilões.

A interface `AuctionCollection` representa uma aplicação para a gestão de leilões (Fig. 2).

- `hasAuction(String seller, String productID)` devolve `true` se existir um leilão para o vendedor `seller` com o identificador do produto `productID` na aplicação e `false` caso contrário;
- `addStandardAuction(String seller, String productID, String description, String category, int basePrice)` recebe o email do vendedor, a informação do produto, o preço base e adiciona um novo leilão de tipo standard. Este método só é executado se a pré-condição `hasAuction(seller,productID)` for falsa;
- `addPlafondAuction(String seller, String productID, String description, String category, int basePrice, int plafond)` recebe o email do vendedor, a informação do produto, o preço base e o plafond e adiciona um novo leilão de tipo plafond. Este método só é executado se a pré-condição `hasAuction(seller,productID)` for falsa;
- Os métodos `isOpen`, `closeAuction` e `makeBid` são similares aos métodos da interface `Auction`, sendo que recebem como argumentos adicionais o email do vendedor (`seller`) e identificador de produto (`productID`). Estas operações só serão executadas se a pré-condição `hasAuction(seller,productID)` for verdadeira. O método `makeBid` tem a pré-condição adicional que a invocação do método `isOpen()` devolve `true`.
- `totalSales(String category)` devolve o valor total das vendas dos produtos dos leilões com a com a categoria `category`. Considere apenas leilões fechados;
- `listAuctionsByCategory(String category)` devolve um iterador que percorre os leilões com a categoria `category`;
- `listCategories()` devolve um iterador que percorre todas as categorias;
- `topPlafond()` devolve o leilão de tipo plafond com o valor de plafond máximo, em caso de empate devolve o leilão mais recente. Caso não exista nenhum leilão de tipo plafond o método devolve `null`;

Note que a classe `AuctionCollectionClass` tem as variáveis de instância `auctions` e `categories` que mantêm os leilões e as categorias de produtos em leilão. A variável `categories` não deve conter repetições de categorias. Deve considerar a actualização das variáveis nos métodos que achar adequados.

Implemente os seguintes métodos da classe `AuctionCollectionClass`:

a) O método `void addPlafondAuction(String seller, String productID, String description, String category, int basePrice, int plafond)`. O constructor da classe `ProdutClass` recebe como argumentos o identificador do produto (`productID`), a descrição (`description`) e a categoria (`category`).

b) O método `Bid makeBid(String seller, String productID, String buyer, int amount)`.

c) O método `int totalSales(String category)`.

d) O método `PlafondAuction topPlafond()`.

Grupo III

Pretendemos estender a interface `Auction` e a classe `AuctionClass` (Fig. 1) com o seguinte método:

```
boolean subBids(Iterator<Bid> itSubBids)
```

que verifica se o vector `bids` inclui a sequência de licitações do iterador `itSubBids`. Por exemplo, vamos considerar que o iterador `itSubBids` itera as licitações `("alice", 12)` e `("bob", 7)` e o vector `bids` inclui as licitações `("eve", 17)`, `("alice", 12)`, `("bob", 7)` e `("marie", 19)` nas posições 0, 1, 2 e 3 (i.e. `counter` é 4). Nesta situação o método deve devolver `true`. No entanto, se o iterador `itSubBids` iterar as licitações `("alice", 12)` e `("marie", 19)` o método deve devolver `false`, porque no vector `bids` as licitações de `alice` e `marie` não são adjacentes. No caso do iterador não iterar qualquer elemento o método `subBids` devolver `true`.

Implemente o método `subBids`.

Grupo IV

Queremos modelar uma aplicação para gerir uma biblioteca pessoal. Esta biblioteca contém livros. Todos os livros têm um título, um autor, um género literário e uma editora. Actualmente a biblioteca suporta três tipos de livros. Os livros tradicionais, os audiobooks e os ebooks. Os livros tradicionais têm um determinado número de páginas e um peso. Os audiobooks não têm a noção de página, ou peso, mas podem ser tocados com a função play, têm uma dimensão em kB, e pelo menos um autor. Os ebooks têm páginas, mas o número de páginas varia consoante a configuração do tamanho da fonte usada na sua visualização. Podemos alterar a fonte usada, resultando isso numa variação correspondente no número de páginas necessárias para apresentar o livro ao leitor. A aplicação deve suportar as seguintes funcionalidades:

- Pesquisa de livro dado um título, e o seu autor, ou apenas um deles, se forem vários. Note que é perfeitamente possível que exista mais que uma cópia do mesmo livro na biblioteca. Por exemplo, podemos ter diferentes edições de um livro de que gostamos muito, sejam num mesmo formato sejam em formatos distintos. A funcionalidade deve devolver uma listagem dos livros encontrados, com informações diferenciadas consoante o tipo de livro.
- Pesquisa de livros de um mesmo autor.
- Pesquisa de livros com a noção de página (neste momento, quer os livros tradicionais quer os ebooks, mas quem sabe se no futuro outros se juntarão). Por outro lado, os audiobooks não tem a noção de página.
- Novo livro, para adicionar um novo livro à coleção.
- Pesquisa de livros com um determinado tipo, por exemplo, livros tradicionais.

Apresente a sua proposta de modelação para o programa, através de um **diagrama de classes e interfaces**, tendo em atenção que deve incluir na sua resposta:

- A interface de topo com a qual o programa principal irá interagir. Esta interface deve ser completamente especificada.
- As variáveis de instância da classe que implementa a interface de topo.
- Para as restantes componentes do diagrama, isto é, para as interfaces e classes que não a interface de topo, omita a indicação das operações e das variáveis de instância.

<u>Nota 1</u>: Não é necessário implementar nenhuma das operações.

<u>Nota 2</u>: Apresente de forma distinta classes e interfaces, identifique caso existam classes abstractas e indique a visibilidade das variáveis de instância.

<u>Nota 3</u>: Para efeitos de clareza da apresentação represente as relações com uma seta etiquetada, por exemplo $A \xrightarrow{extends} B$ significa A extends B (analogamente para implements, eventualmente com linha tracejada, mas sempre etiquetada com implements).