Pontifícia Universidade Católica do Rio Grande do Sul
Programação Orientada a Objetos
Prof. Marcelo H. Yamaguti
2023/2

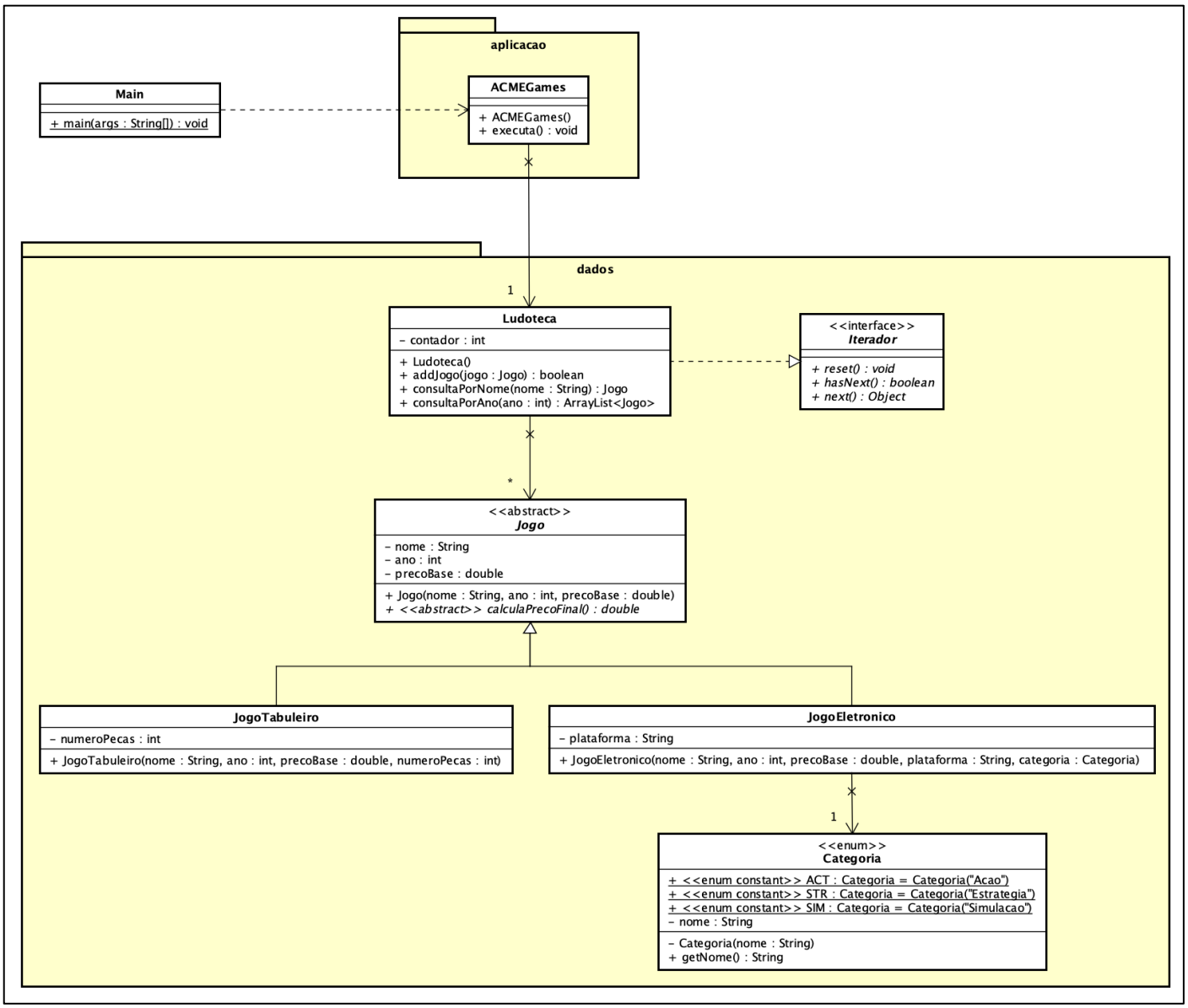
Exercício de Avaliação 2

1. **Enunciado geral:**

A ACMEGames comercializa jogos (eletrônicos e de tabuleiro) e deseja uma aplicação que faça o processamento de dados de seus jogos.

Você será responsável pelo desenvolvimento da aplicação.

O analista de sistemas gerou um diagrama de classes inicial, com alguns atributos, operações e relacionamentos apresentados a seguir.



O analista definiu as seguintes operações da interface **Iterador**:

- **reset()**: reinicia a iteração na coleção.
- **hasNext()**: retorna *true* se ainda há elementos para a iteração, ou *false* em caso contrário.
- **next():** retorna o próximo elemento da iteração.

Sabe-se que será necessário haver subclasses da classe abstrata ***Jogo***. Cada subclasse possui informações adicionais específicas:

- ***JogoTabuleiro***: possui o número de peças.
- ***JogoEletronico***: possui uma plataforma e uma categoria (que pode ser: ACT (jogo de ação), SIM (jogo de simulação) ou STR (jogo de estratégia)).

O método **calculaPrecoFinal()** depende da subclasse:

- ***JogoTabuleiro***: é o preço base acrescido de 1% para cada peça.
- ***JogoEletronico***: é o preço base acrescido de um percentual sobre o preço base conforme a categoria: 10% se for ação; 30% se for simulação; 70% se for estratégia.

O método **executa()** da classe ACMEGames deve realizar a sequência de passos:

1. **Cadastrar jogos eletrônicos**: lê todos os dados de cada jogo eletrônico e, se o nome não for repetido no sistema, cadastra-o no sistema. Se o nome do jogo for repetido mostra a mensagem no formato: **`1:Erro-jogo com nome repetido: nome do jogo`**
   Para cada jogo eletrônico cadastrado com sucesso no sistema, mostra os dados do jogo no formato: **`1:nome do jogo,preço final`**
2. **Cadastrar jogos de tabuleiro**: lê todos os dados de cada jogo de tabuleiro e, se o nome não for repetido no sistema, cadastra-o no sistema. Se o nome do jogo for repetido mostra a mensagem no formato: **`2:Erro-jogo com nome repetido: nome do jogo`**
   Para cada jogo de tabuleiro cadastrado com sucesso no sistema, mostra os dados do jogo no formato: **`2:nome do jogo,preço final`**
3. **Mostrar os dados de um determinado jogo**: lê o nome de um jogo. Se não existir um jogo com o nome indicado, mostra a mensagem de erro: **`3:Nome inexistente`**.
   Se existir, mostra os dados do jogo no formato: **`3:atributo1,atributo2,atributo3,...,preço final`**
4. **Mostrar os dados de jogo(s) de um determinado ano**: lê o ano de um jogo. Se não existir um jogo com o ano indicado, mostra a mensagem de erro: **`4:Nenhum jogo encontrado`**.
   Se existir, mostra os dados do(s) jogo(s) no formato: **`4:atributo1,atributo2,atributo3,...,preço final`**
5. **Mostrar os dados de jogo(s) eletrônico(s) de uma determinada categoria:** lê a categoria de jogo. Se não existir a categoria indicada, mostra a mensagem de erro: **`5:Categoria inexistente`**.
   Se não existir um jogo com a categoria indicada, mostra a mensagem de erro: **`5:Nenhum jogo encontrado`**.
   Se existir, mostra os dados do(s) jogo(s) no formato: **`5:atributo1,atributo2,atributo3,...,preço final`**
6. **Mostrar o somatório de preço final de todos os jogos**: calcula o somatório do preço final de todos os jogos do sistema. Se não existir jogo cadastrado no sistema, mostra a mensagem de erro: **`6:Nenhum jogo encontrado`**.
   Se existir, mostra a mensagem no formato: **`6:valor do somatório`**
7. **Mostrar os dados do jogo de tabuleiro com maior preço final**: localiza o jogo de tabuleiro cadastrado com maior preço final. Se não existir nenhum jogo de tabuleiro cadastrado, mostra a mensagem de erro: **`7:Nenhum jogo encontrado.`**
   Se existir, mostra os dados do jogo no formato: **`7:nome,preço final`**

2. Definição do exercício:

O objetivo do exercício é implementar um sistema que capaz de atender as necessidades da empresa descrita no enunciado geral, e que atenda as restrições a seguir:

- A entrada de dados ocorrerá por leitura de arquivo de texto.
  - Pode-se utilizar de redirecionamento de E/S: ajuste a classe ACMEGames para ler e escrever em arquivos: veja na área Moodle da disciplina > módulo: Materiais de apoio > CÓDIGOS AUXILIARES > Redirecionamento de entrada/saída de dados para arquivos.
  - Outra alternativa é a leitura e escrita em arquivos-texto.
- Os dados de entrada estarão no arquivo 'dadosin.txt':
  - No passo **1. Cadastrar jogos eletrônicos**: cada linha corresponde ao nome, ano, preço base, plataforma e categoria de um jogo eletrônico. Quando a linha lida for -1, não há mais jogos eletrônicos a serem cadastrados.
  - No passo **2. Cadastrar jogos de tabuleiro**: cada linha corresponde ao nome, ano, preço base e número de peças de um jogo de tabuleiro. Quando a linha lida for -1, não há mais jogos de tabuleiro a serem cadastrados.
  - As últimas linhas do arquivo 'dadosin.txt' correspondem a:
    - Nome do jogo para o passo 3.
    - Ano para o passo 4.
    - Categoria para o passo 5.
- A saída de dados deve ser gravada no arquivo 'dadosout.txt'
- Toda entrada e saída de dados com o usuário deve ocorrer apenas na classe ACMEGames.
- Para o armazenamento dos jogos no sistema deve haver apenas uma lista de jogos (List ou similar).
- Todos os atributos das classes devem ser privados.
- É permitida a criação de novos métodos, atributos e relacionamentos, mas as informações definidas no diagrama de classes original não podem ser alteradas.
- O diagrama de classes deve ser atualizado conforme as alterações realizadas e deve ser entregue em arquivo Astah ou PDF.

3. Critérios de avaliação

O exercício será avaliado conforme os seguintes critérios:

- Diagrama de classes atualizado: 1 ponto.
- Implementação de pacotes e enumeração: 1 ponto.
- Uso e implementação de generalização e interface: 2 pontos.
- Uso de polimorfismo: 1 ponto.
- Implementação correta conforme a descrição do exercício e o diagrama de classes: 2 pontos.
- Execução correta das opções previstas: 3 pontos.

- *Ponto extra (opcional) de 1 ponto (máximo de 10 pontos): implementar passos adicionais:*
  - **Mostrar os dados do jogo com preço base mais próximo da média dos preços base**: calcula a média dos preços base dos jogos cadastrados e localiza o jogo com preço base mais próximo da média calculada. Se não existir nenhum jogo cadastrado, mostra a mensagem de erro: **`8:Nenhum jogo encontrado.`**
    Se existir, mostra os dados do jogo no formato: **`8:média dos preços base,atributo1,atributo2,atributo3,...,preço final`**
  - **Mostrar os dados do jogo de tabuleiro mais antigo**: mostra os dados do jogo de tabuleiro mais antigo. Se não existir nenhum jogo de tabuleiro cadastrado, mostra a mensagem de erro: **`9:Nenhum jogo encontrado.`**
    Se existir, mostra os dados do jogo no formato: **`9:nome do jogo,ano`**

4. **Entrega:**
   - A entrega do exercício envolverá:
     - arquivos dos códigos-fonte do sistema (e demais arquivos necessários para a compilação do sistema).
     - diagrama de classes atualizado.
   - Deverá ser gerado um arquivo compactado (.zip ou .rar), com os itens acima, e entregue na tarefa da área Moodle da disciplina.
   - Data de entrega: **1º / 11 / 2023**

5. **Considerações finais:**
   - O exercício deve ser desenvolvido **individualmente**.
   - A implementação deve seguir o Java Code Conventions para nomes de identificadores e estruturas das classes.
   - Não será aceito exercício com erros de compilação. Programas que não compilarem corretamente terão nota zerada.
   - A cópia parcial ou completa do exercício terá como consequência a atribuição de nota 0 (zero) aos exercícios dos alunos envolvidos. Para análise de similaridade será utilizado o MOSS (https://theory.stanford.edu/~aiken/moss/).