Disciplinas Lotadas!

Não se sabe ao certo o motivo, mas nos últimos semestres um conjunto de disciplinas provocou uma retenção grande de alunos, e foi necessário criar várias turmas extras em um mesmo período letivo, para que as salas de aula não ficassem lotadas. Entretanto, com tantas disciplinas de mesmo nome ocorrendo paralelamente, algunas dados foram digitados incorretamente no Sistema Acadêmico.

Seu professor de Programação pediu para você criar um sistema em Java para organizar essas informações. Existem três arquivos textos à sua disposição:

- O arquivo **alunos.txt** contém dados de todos os alunos do curso, mesmo os que não fizeram tais disciplinas. A primeira linha do arquivo contém a quantidade n de alunos cadastrados. Em seguida, para cada n, existem três linhas consecutivas com **nome**, **CPF** e **matrícula** (nesta ordem) de cada aluno. Exemplo:

  ```
  4
  Leonor Almeida
  906.827.881-72
  621306114
  Matilde Almeida
  006.189.399-36
  685194292
  Francisca Braganca
  756.837.927-17
  283195103
  Mariana Sampaio
  972.999.637-35
  868717473
  ```

  No arquivo do exemplo, existem informações sobre 4 alunos. A segunda aluna, por exemplo, se chama "Matilde Almeida", seu CPF é "006.189.399-36" e sua matrícula é "685194292".

- O arquivo **professores.txt** contém os dados dos professores que se envolveram com essas disciplinas. A primeira linha do arquivo contém a quantidade n de professores salvos no arquivo. Em seguida, para cada n, existem duas linhas consecutivas com **nome** e **CPF** (nesta ordem) de cada professor. Exemplo:

  ```
  2
  Hilario Seibel Jr
  475.235.784-21
  Wagner Kirmse
  569.778.576-74
  ```

Neste caso, temos nome e CPF de apenas dois professores.

- O arquivo **turmas.txt** contém as informações sobre essas turmas. A primeira linha do arquivo contém a quantidade n de turmas cadastradas. Em seguida, para cada n, o arquivo apresenta os seguintes dados (um por linha, nesta ordem):
  - Nome da disciplina.
  - Ano em que ela ocorreu.
  - Semestre do ano em que ela ocorreu (1 ou 2).
  - CPF do professor responsável.
  - Número x de alunos matriculados.
  - Para cada x, duas linhas contendo matrícula e nota final de um dos alunos.

  Exemplo:

```
2
Programacao 2
2021
1
475.235.784-21
2
621306114
100
685194292
75
Programacao 2
2021
1
569.778.576-74
2
283195103
88
868717473
30
```

Neste exemplo, temos apenas duas turmas (uma destacada em azul e outra em vermelho). As duas foram turmas de "Programação 2" ofertadas em "2021/1", ambas com dois alunos matriculados. Na segunda, o professor "Wagner Kirmse" (CPF 569.778.576-74) deu aula para "Francisca Braganca" (matrícula 283195103, que passou com 88) e para "Mariana Sampaio" (matrícula 868717473, que teve nota 30).

Você deve criar, ao menos, as seguintes classes:

- **Pessoa**: Classe abstrata que contém os atributos **nome** e **cpf** (ambos Strings).

- **Aluno**: Subclasse de **Pessoa**, contém também o atributo matrícula (String) e implementa a interface **Comparable<Aluno>**. Quando dois alunos forem comparados, deverá ser usada ordem crescente das Strings com seus nomes. Caso seus nomes sejam iguais, serão ordenados crescentemente pelas Strings com suas matrículas.

- **Professor**: Outra subclasse de **Pessoa**, também deve implementar a interface **Comparable<Professor>**. Quando dois professores forem comparados, deverá ser usada ordem crescente das Strings com seus nomes. Caso seus nomes sejam iguais, serão ordenados crescentemente pelas Strings com seus CPFs.

- **AlunoNota**: Associa um aluno à nota que ele tirou em um determinada turma. Armazena apenas dois atributos: o objeto da classe Aluno e sua nota (double). Deve implementar a interface **Comparable<AlunoNota>**. Quando dois objetos dessa classe forem comparados, deverá vir primeiro o objeto com **maior** nota. Caso elas sejam iguais, será usada ordem já definida para objetos da classe Aluno.

- **Turma**: Possui cinco atributos:

  1. Nome da disciplina (String).
  2. Ano em que ocorreu (int).
  3. Semestre do ano em que ocorreu (int).
  4. Professor responsável pela turma (objeto da classe **Professor**).
  5. Lista de objetos da classe **AlunoNota**.

  Também deve implementar a interface **Comparable<Turma>**. Quando duas turmas forem comparadas, deverá vir primeiro a que ocorreu por último (para isso, analise o ano e o semestre). Se duas turmas ocorreram no mesmo período letivo, deve ser usada ordem alfabética de seus nomes (crescente). Se ainda assim houver empate, o último critério é a ordenação de seus professores (lembre-se que a classe Professor também é comparável).

- **Sistema**: Armazena uma lista com todos os alunos cadastrados, uma lista com todos os professores, e outra lista com todas as turmas. Quando um objeto desta classe for criado, as listas já devem ser inicializadas com os dados lidos automaticamente dos três arquivos de entrada. Para cada um dos arquivos de entrada, crie um método separado para leitura de seus dados, para melhorar a modularização do código. Por fim, impelemente o método **exibirNotas()**, que vai ordenar todos os dados apropriadamente e salvá-los no arquivo "saida.txt". O arquivo de saída deve conter, para cada turma:

  1. Uma linha contendo o nome da disciplina, o período letivo em que ocorreu, e o nome do professor responsável.

2. Para cada aluno da turma, uma linha contendo o nome do aluno, sua matrícula e sua nota final.

A saída deve respeitar **exatamente** o formato a seguir (cada caracter da sua saída para este exemplo deve ser **exatamente** igual a esta saída, cada espaço, hífen ou parênteses é importante):

```
Programacao 2 (2021/1) - Hilario Seibel Jr
- Leonor Almeida (621306114): 100.0
- Matilde Almeida (685194292): 75.0
Programacao 2 (2021/1) - Wagner Kirmse
- Francisca Braganca (283195103): 88.0
- Mariana Sampaio (868717473): 30.0
```

Além disso, a saída deve seguir os critérios de ordenação especificados anteriormente para turma, notas, etc. Repare que as turmas ocorreram no mesmo período letivo (2021/1) e possuem o mesmo nome, então foram ordenadas pelo professor. E em cada uma delas, os alunos foram ordenados decrescentemente pela nota final.

Você deve encapsular apropriadamente os atributos de todas as classes.

Tratamento de erros e exceções

Os métodos que manipulam o arquivo devem conter tratamento adequado de erros e exceções. O método que localiza um aluno dentro de uma lista de alunos, dada sua matrícula, deve lançar a exceção "AlunoNotFoundExcepection", que você deverá criar, imprimir a mensagem "Aviso: Aluno nao encontrado." e continuar executando o programa normalmente. Se você estiver lendo o arquivo "turmas.txt" quando isto ocorrer, o aluno não encontrado será ignorado e você continuará lendo o restante do arquivo. Por isso, lembre-se que matrícula e nota sempre vêm aos pares (um por linha), neste arquivo. Portanto, tente localizar o aluno apenas após ler estas duas informações (sua matrícula **e** sua nota).

O método que localiza um professor dentro de uma lista de professores, dado seu CPF, deve lançar a exceção "ProfessorNotFoundExcepection", que você deverá criar, imprimir a mensagem "Aviso: Professor nao encontrado." e abortar o programa (pois não seria possível continuar com o sistema se uma disciplina não tiver um professor).

Observações

- O trabalho vale 40 pontos e o código deve ser feito em Java.
- O trabalho pode ser feito em até 2 integrantes.

- Trabalhos entregues após o prazo serão automaticamente rejeitados.
- Trabalhos considerados plágio terão nota zero e serão enviados ao conselho de ética.
- Trabalhos com erro de execução, com formato de saída incorreto, ou que não compilarem terão nota 0.
- O trabalho deve ser enviado na sala da disciplina do AVA.
- Em caso de dúvidas na especificação do trabalho ou no próprio trabalho, contate-me em hsjunior@gmail.com