Lista 7 - Herança e polimorfismo

Prazo de entrega: 07/06/2024 até 23:59 (Sexta)

Instruções de entrega: Você deve ter um repositório em seu github para a disciplina de POO chamado **Programacao-Orientado-Objetos**.
Crie uma pasta dentro dele chamada Listas e dentro dela crie uma pasta chamada Lista7.
Para cada exercício da lista deve ser criado um projeto java.
Ao terminar a lista, suba seus exercícios no github e envie o link no formulário abaixo.

Link para o formulário de envio: https://forms.gle/8h8hUa3mBQ2UFdpd8

MONITORIAS: Serão realizados quatro dias de monitorias para auxiliar com a resolução dos exercícios. As datas e horários das monitorias são:

Dia	Horários		
Segunda-Feira	19h até 21h		
Terça-Feira	19h até 21h	-	-
Quarta-Feira	09h até 11h	19h até 21h	-
Quinta-Feira	19h até 21h	-	-
Sexta-Feira	09h até 11h	19h até 21h	22h até 23:59

OBS: Se você não conseguir participar das monitorias nos dias e horários propostos, você pode solicitar uma monitoria em uma data e horário diferente, desde que seja comunicado com pelo menos um dia de antecedência 🙂

EXERCÍCIO

Você foi contratado por uma escola de Hogwarts para desenvolver um novo sistema de gestão escolar.

Como parte da equipe de desenvolvimento, você foi designado para criar a estrutura de classes para representar os diferentes tipos de usuários no sistema: alunos e professores.

1 - Crie um pacote chamado `escola`.

2 - Dentro do pacote `escola`, crie a classe base `Usuario`.

- **Atributos**:
  - `nome`: `String` - Representa o nome do usuário.
  - `email`: `String` - Representa o email do usuário.
  - Os atributos devem ser **privados**.
- **Construtor**:
  - Crie um construtor que inicialize os atributos `nome` e `email` com os valores fornecidos.
- **Métodos**:
  - `public void exibirInfo()`: Este método deve imprimir o nome e o email do usuário.
- **Getters e Setters:**
  - Desenvolva getters e setters para cada um dos atributos.

3 - Crie a subclasse `Aluno` que herda de `Usuario`**.**

- **Atributo adicional**:
  - `matricula`: `String` - Representa a matrícula do aluno.
  - O atributo deve ser **privado**.
- **Construtor**:
  - Crie um construtor que inicialize os atributos `nome`, `email` e `matricula` com os valores fornecidos.
- **Métodos**:
  - Sobrescreva o método `exibirInfo()` da super classe, para que, além de exibir o nome e o email do usuário, exiba também a matrícula do aluno.
  - Sobrecarregue o método `exibirInfo(boolean exibirNome, boolean exibirEmail, boolean exibirMatricula)`, cada parâmetro é um booleano que representa um atributo, e imprima cada atributo onde seu parâmentro booleano seja true.
- **Getter e Setter**:
  - Desenvolva getter e setter para o atributo adicional.

4 - Crie a subclasse `Professor` que herda de `Usuario`.

- **Atributo adicional**:
  - `disciplina`: `String` - Representa o departamento do professor.
  - O atributo deve ser **privado**.
- **Construtor**:
  - Crie um construtor que inicialize os atributos `nome`, `email` e `disciplina` com os valores fornecidos.
- **Métodos**:
  - Sobrescreva o método `exibirInfo()` para que, além de exibir o nome e o email do usuário, exiba também o departamento do professor.
  - Sobrecarregue o método `exibirInfo(boolean exibirNome, boolean exibirEmail, boolean exibirDisciplina)`, cada parâmetro é um booleano que representa um atributo, e imprima cada atributo onde seu parâmentro booleano seja true.
- **Getter e Setter:**
  - Desenvola getter e setter para o atributo adicional.

Na classe `App`:

- **Método `main`**:
  - Crie uma instância de `Aluno` utilizando o construtor com parâmetros.
  - Crie uma instância de `Professor` utilizando o construtor com parâmetros.
  - Chame os métodos `exibirInfo (O sobrescrito e o sobrecarregado)` para cada objeto, utilizando diferentes combinações de parâmetros booleanos.