UNIVERSIDADE FEDERAL DE LAVRAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

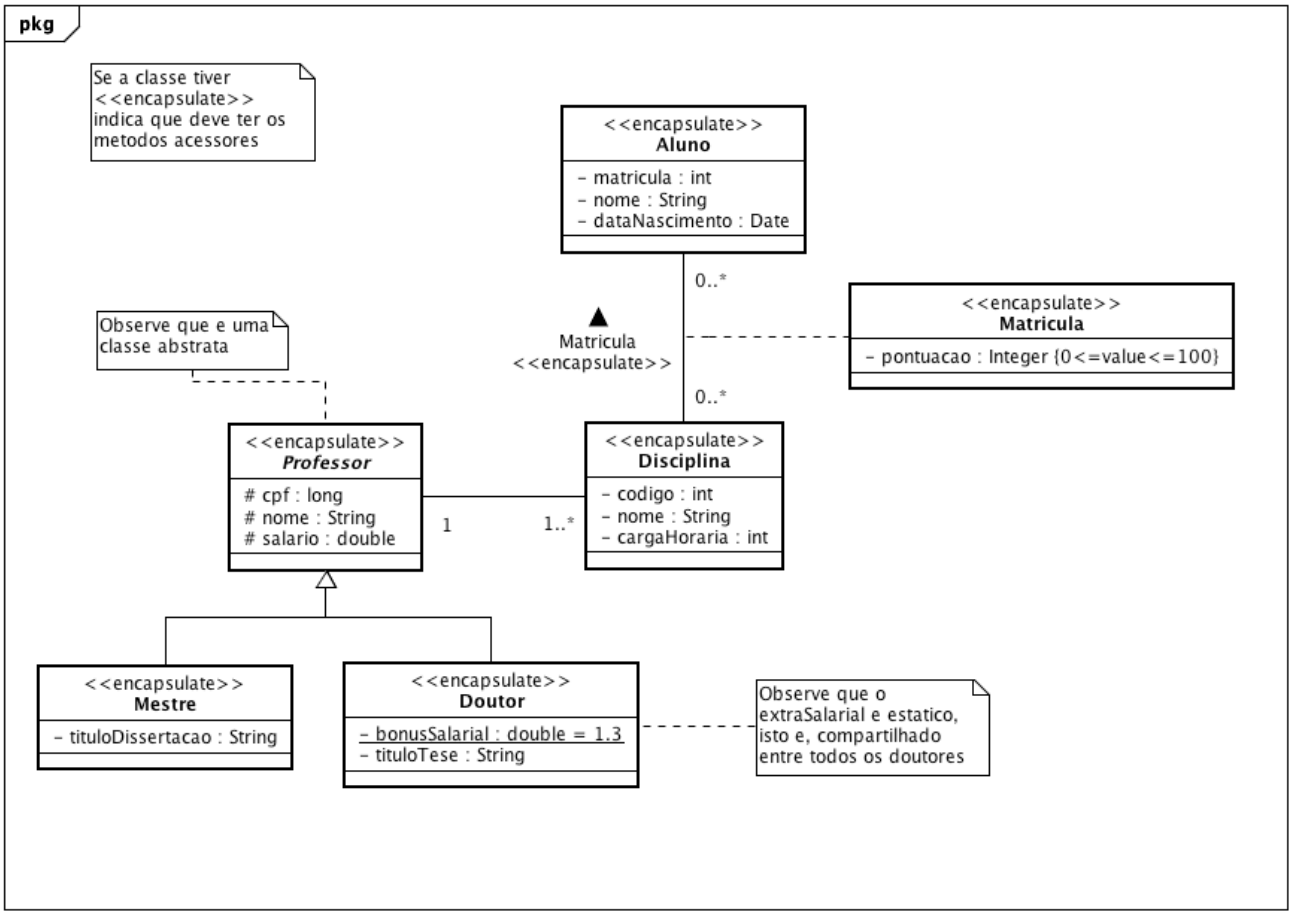
Curso: *Ciência da Computação* | *Grupo de Estudos Java* | Professor: *Ricardo Terra*

Pontuação: ϕ *pontos* (1 *questões)* | **UML** | Data: ϕ

1. Estude as seguintes seções da Apostila:

   - Herança
   - Modificadores
   - Classes Wrapper (e.g. Integer)
   - Classe Date
   - Classe StringBuilder e StringBuffer
   - Coleções

2. Dado o seguinte Diagrama de Classe, o implemente em um pacote chamado **`entidades`**.



3. Para todas as classes, implemente o método **`toString`**, de forma que exiba as informações mais relevantes de um objeto daquela classe, e o método **`equals`**, de forma que compare a igualdade entre dois objetos de uma classe pelos seus atributos.

4. O método **`getSalario`** de um professor doutor deverá ser sobrescrito para retornar *salário x bônus salarial*. Isto é, se o salário é 1000.00 e o bônus salarial é 1.3, o **`getSalario`** deverá retornar 1300.00.

5. As classes **`Aluno`**, **`Disciplina`**, **`Matricula`**, **`Mestre`** e **`Doutor`** não devem possuir subclasses.

6. O número do cpf do professor, da matrícula do aluno e do código da disciplina não podem ser alterados, isto é, uma vez atribuídos não estarão susceptíveis a serem alterados.

7. Criar construtores para **todas** as classes de forma que tenha sempre que passar **todos** os atributos para que a mesma seja criada.
   Dicas: No caso da classe **`Professor`**, mesmo ela sendo abstrata, deverá ser criada um construtor recebendo o cpf, o nome e o salário e os construtores das subclasses (**`Mestre`** e **`Doutor`**) deverão obrigatoriamente chamar esse construtor com o uso da referência **`super`**.

8. Crie uma classe chamada **`BD`** que armazene uma lista (**`ArrayList`**, por exemplo) de **`Aluno`**, uma de **`Disciplina`**, uma de **`Matricula`**, uma de **`Professor`** (que armazenará tanto mestres como doutores).

9. Nessa classe **`BD`**, deverá existir um método **`gerarBackup`** que irá retornar um **`StringBuilder`** com um cabeçalho contendo a data atual no formato abaixo e o conteúdo textual (gerado pelo método **`toString`**) de todos os objetos de todas as listas. Observe exemplo:

```
Backup realizado em 5 de outubro de 2009 às 12:04:34

Alunos
1;Carlos Alberto;07/08/89
2;José Marcos;02/01/86

Professores
11111111111;Daniel de Paula;1000.00;Mestre
22222222222;César Couto;1400.00;Doutor

Disciplinas
100;LTP-III;60;22222222222
101;ED-I;80;11111111111

Matriculas
100;2;99
101;1;NULL
```

10. Crie uma classe chamada **`Aplicacao`** que faça as seguintes operações:
   a. Incluir, Excluir e Listar Aluno (conteúdo da lista de Aluno em **`BD`**)
   b. Incluir, Excluir e Listar Professor (conteúdo da lista de Professor em **`BD`**)
   c. Incluir, Excluir e Listar Disciplina (conteúdo da lista de Disciplina em **`BD`**)
   d. Matricular um aluno em uma disciplina (criar objeto **`Matricula`** e inserir na lista de Matricula em **`BD`**)
   e. Inserir nota para um aluno de uma disciplina (Se nota < 60, escrever REPROVADO, se nota >= 60, escrever APROVADO, se a nota ainda não tiver sido inserida, escrever EM CURSO)
   f. Listar as disciplinas que um dado aluno esteja matriculado
   g. Listar as disciplinas que um dado professor leciona
   h. Exibir conteúdo textual gerado pela cópia de segurança (*backup*)

11. Atualize o método listar de aluno, professor e disciplina para listar ordenado pelo respectivo nome. Caso haja colisão de nomes (i.e., nomes iguais), compare pela matrícula, cpf e código, respectivamente.

12. Fazer o método de atualização de Aluno, Professor e Disciplina.

13. Emitir um relatório geral. Nesse relatório deverá aparecer a listagem de professor e suas disciplinas, alunos e suas notas, etc. Seria uma visão geral de todas as informações do sistema.

Observações gerais:

- Siga a nomenclatura de nomes de classes, métodos e atributos;
- Atributos geralmente são privados;
- É recomendável a criação de construtores;
- Métodos acessores são indispensáveis.