

PRÁTICA LABORATORIAL 14

Objetivos:

- Herança
- Classes Abstratas

EXERCÍCIOS

Parte 1

1. Crie uma classe abstrata chamada "FiguraGeometrica" com o seguinte atributo: Cor (String). A classe deve ter um construtor que recebe a cor como parâmetro.
 - a. Crie o método mostrarCor() que imprime na consola: "A forma geométrica é (cor)"
 - b. Crie os métodos abstratos: double calcularArea(), double calcularPerimetro().
 - c. Crie duas subclasses concretas de FiguraGeometrica: "Retangulo", "Circulo" e "Triangulo". Essas classes devem ter um construtor que chama o construtor da superclasse (FiguraGeometrica), assim como atributos específicos para cada um deles (comprimento, base, altura, raio, etc).
 - d. Para Retangulo: "calcularArea()" que calcula a área do retângulo (base x altura) e retorna o valor como um double. Também deve ter o método calcularPerimetro().
 - e. Para Circulo: "calcularArea()" que calcula a área do círculo ($\pi \times \text{raio}^2$) e retorna o valor como um double. Também deve ter o método calcularPerimetro().
 - f. Para Triangulo: "calcularArea()" que calcula a área do triângulo $((\text{base} \times \text{altura}) / 2)$ e retorna o valor como um double. Também deve ter o método calcularPerimetro().
 - g. Crie uma classe chamada "FigurasGeometricasDemo" com um método main que cria um objeto de cada uma das classes (**FiguraGeometrica**, Retangulo, Circulo e Triangulo) e chama o método "mostrarCor()" e "calcularArea()" de cada um deles.

2. Implemente um jogo sobre Personagens que enfrentam inimigos perigosos, o objetivo será o jogador definir qual a sua personagem e, seguidamente, jogar contra os inimigos, para isso:
 - a. Crie uma classe abstrata “Entidade” que tem como atributos nome (String), vida (int), força (int). Seguidamente, crie duas subclasses Personagem e NPC (non-playable character).
 - b. A classe Personagem terá o atributo nível (int), categoria (ENUM contendo, por exemplo, cavaleiro, mago, arqueiro, etc...) e arma (String).
 - c. Deve ser implementado o método atacar que recebe um inimigo (NPC) como parâmetro. Este método “atacar” compara as estatísticas da personagem vs NPC. Tenha em consideração o seguinte:
 - i. O jogo será por turnos, pelo que no primeiro turno de ataque começa sempre o personagem. Será deduzida a força da personagem à vida do inimigo. Ao fim do ataque do jogador, o inimigo retribui, sendo deduzida à vida do jogador a força do inimigo.
 - ii. O método deve executar até que uma das entidades fique com a vida igual ou menor que zero. Se o jogador ficar vivo, imprime uma mensagem na consola com o seguinte aspeto: “Parabéns (“categoria da personagem”) (“nome da personagem”), ganhou um combate contra um (“inimigo”) com o uso de (“arma”).” e passa de nível e incrementa a vida e a força em 10%, restaurando a vida ao valor total mais incremento. Se o jogador morrer, o programa deve encerrar com a mensagem “Perdeu”.
 - d. Numa classe Jogo, defina um método main, instancie uma personagem e três inimigos, de seguida, faça a personagem enfrentar todos os inimigos.
3. Após conclusão do curso Software Developer, foi contactado por uma entidade de formação para implementar um programa que permita gerir os seus formadores, formandos e respetivas nota, assim como cursos e disciplinas. Para isso:
 - Crie a classe abstrata Pessoa que tem atributos: String nome, int anoNascimento, String email, String telemovel.
 - Crie o método abstrato obterFuncao().
 - Crie o método exibirDetalhes() que imprime na consola os detalhes de uma Pessoa.
 - Crie o método abstrato imprimirHorario().

Seguidamente, deve criar também as classes Professor, Funcionário e Aluno.

- Professor terá os atributos: ArrayList<AreaFormacao> areaFormação [Informática, Economia, Contabilidade, Medicina, etc], int nivelAcademico.
- Funcionário terá os atributos: String função [Administração, Recursos Humanos, Limpeza, etc]
- Aluno terá os atributos: Curso curso, double mediaNotas, String[][] pauta.

Crie a classe `Disciplina` que terá como atributos: `String nome`, `AreaFormacao area`, `int duracaoHoras`.

- Crie o método construtor que permita criar uma disciplina pela consola.

Crie a classe `DisciplinaCurso` que terá como atributos: `Disciplina disciplina`, `Professor professor`.

Crie a classe `Curso` que terá como atributos: `String nome`, `ArrayList<DisciplinaCurso>`, `boolean longaDuracao` (`false` para curta duração (máximo de 100 horas, inclusive) ou `true` para longa duração (mínimo de 101 horas)). Para esta classe crie também os métodos:

- `criarCurso` que funcione como construtor através de feedback da consola. Deve perguntar os atributos `nome`, e quantas disciplinas terá.
Seguidamente, deve perguntar quais as disciplinas que quer adicionar ao curso (deve imprimir na consola todas as disciplinas do sistema). Quando o utilizador escolher uma disciplina, deve perguntar qual o professor que vai lecionar (deve imprimir na consola todos os professores aptos para lecionar aquela disciplina, tenha como fator de decisão a Área de Formação do Professor ser a mesma da Disciplina. Como por exemplo: Um professor que tenha formação em Informática apenas pode lecionar disciplinas de informática, um professor que tenha formação em Economia e Contabilidade pode lecionar disciplinas dessa natureza).
Finalmente, deve verificar o número de horas e atribuir automaticamente `true` ou `false` ao atributo `"longaDuracao"`.

Para a classe `Professor`:

- Crie o método `lançarNota(Aluno aluno, Disciplina disciplina, double nota)` que recebe como parâmetro o aluno, a disciplina e a nota. Deve verificar se o professor pode lançar a nota desta disciplina, se o aluno tem a disciplina no curso e se a nota se encontra entre 0 e 20.
- Implemente o método `imprimirHorario()` que imprime todos Cursos e Disciplinas que o professor leciona.

```
> *** Software Developer ***  
> Algoritmia e Programação  
> Programação Estruturada  
> Bases de Dados  
> Design Patterns  
  
> *** Front-End Developer ***  
> Algoritmia e Programação  
> Programação Estruturada
```

- Implemente o método `obterFuncao()` que devolve `"Professor"`.

Para a classe Aluno:

- Crie o método `imprimirPauta()` que imprime na consola a matriz com o seguinte aspeto:

```
> Algoritmia e Programação - 18.5
> Programação Estruturada - 19
> Bases de Dados - 15.8
> Design Patterns - 20
```

- Crie o método `estadoAprovacao()` que imprime na consola se o aluno está aprovado ou não (média igual ou superior a 9.5 é considerado aprovado).
- Crie o método `calcularMedia()`.
- Implemente o método `imprimirHorario()` que imprime as Disciplinas que o aluno tem de assistir.
- Implemente o método `obterFuncao()` que devolve "Aluno".

Para a classe Funcionario:

- Implemente o método `imprimirHorario()` que imprime na consola "8h a (administrar)(recrutar)(limpar)(etc)"
- Implemente o método `obterFuncao()` que devolve a respetiva função.

Crie a classe Escola que tenha os atributos: String nome, String localização, int lotação máxima de alunos, ArrayList<Professores>, ArrayList<Disciplina>, ArrayList<Cursos>, ArrayList<Alunos>. Deve ter os métodos adicionar novos professores, novos cursos, novas disciplinas e novos alunos.

Crie um menu que inicialmente pergunte o tipo de login: Professor, Aluno ou Funcionário.

De acordo com o login efetuado deve apresentar o menu adequado:

- Sendo que só o administrador pode criar disciplinas e cursos. Assim como ver a lista de todos os alunos aprovados.
- Só os recursos humanos podem criar professores e alunos.
- Professor pode lançar notas e imprimir horário.
- Aluno pode ver o estado de aprovação, ver a media e imprimir o horário.

Bom trabalho! 😊