INSTITUTO FEDERAL
Sul de Minas Gerais
Campus
Poços de Caldas

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO SUL DE MINAS GERAIS

Campus Poços de Caldas

Avaliação Final

1) Crie um projeto Java com quatro pacotes distintos:

- `ifsuldeminas.academico`
- `ifsuldeminas.alunos`
- `ifsuldeminas.funcionarios`
- `ifsuldeminas.main`

No pacote `ifsuldeminas.alunos` deve-se implementar a classe `Aluno` com os seguintes atributos, construtores e métodos:

- `nome:String` (um atributo para armazenar o nome de um aluno).
- `ra:int` (Um inteiro no intervalo [1000,9999] que representa o registro acadêmico do aluno).
- `periodo:int` (um inteiro para armazenar o período do aluno).
- `curso:String` (Um atributo para armazenar qual é o curso do aluno).
- `Aluno(String nome, int ra, String curso)` (Um construtor que permite construir um objeto aluno com nome, registro acadêmico e curso. Assuma que um objeto criado com esse contrutor permite criar um aluno no primeiro período, isto é, `periodo=1`).
- `Aluno(String nome, int ra, int periodo, String curso)` (Um construtor que permite construir um objeto aluno com nome, registro acadêmico, período e curso).
- *setters* e *getters*.
- `exibirAluno():void` (Um método para exibir todas as informações do aluno).

No pacote `ifsuldeminas.funcionarios` deve-se implementar a classe `Professor` com os seguintes atributos, construtor e métodos:

- `nome:String` (um atributo *String* cujo valor é o nome do professor).
- `suap:int` (um atributo inteiro no intervalo [1000, 9999]).
- `Professor(String nome, int suap)` (Um construtor que permita criar um objeto que represente um professor com nome e número suap informados).

- *setters* e *getters*.
- `exibirProfessor():void` (Um método que exibe as informações do professor).

No pacote `ifsuldeminas.academico` deve-se implementar a classe `Disciplina` com os seguintes atributos, construtor e métodos:

- `nome:String` (Um atributo que armazena o nome da disciplina).
- `int:periodo` (Um atributo para armazenar o período em que uma disciplina é ofertada).
- `int:numAulasSemana` (Um atributo para armazenar o número de aulas da disciplina por semana).
- `int:numTotalAulas` (Um atributo para armazenar o número total de aulas da disciplina).
- `Professor:professor` (Um atributo para armazenar qual é o professor da disciplina).
- `ArrayList<Aluno> alunos` (Uma lista para armazenar todos os alunos que fazem determinada disciplina).
- `ArrayList<Double> notas` (Uma lista para armazenar as notas dos alunos na disciplina. Essa lista deverá ter sempre o mesmo tamanho da lista `alunos` e o valor de `notas.get(i)` corresponde ao valor da nota do aluno na posição `alunos.get(i)`).
- `ArrayList<Integer> frequencias` (Uma lista para armazenar as frequências dos alunos na disciplina. Essa lista deverá ter sempre o mesmo tamanho da lista `alunos` e, portanto, o valor de `frequencia.get(i)` corresponde à frequencia do aluno na posição `alunos.get(i)`).
- `Disciplina(String nome, int periodo, int numAulasSemana, int numSemanas)` (Um construtor que permita criar uma disciplina com nome, período e número de aulas especificados. O valor do inteiro `numSemanas` refere-se ao número de semanas que acontecerão as aulas e, portanto,

  ```
  numTotalAulas = numAulasSemana*numSemanas
  ```

  Nesse construtor deve-se construir o objeto `alunos`, o objeto `notas`, o objeto `frequencias` e armazenar no atributo `professor` o valor `null`, indicando que a disciplina ainda não tem professor).
- `Disciplina(String nome, int periodo, int numAulasSemana, int numSemanas, Professor professor)` (Um construtor que permita criar uma disciplina com nome, período, número de aulas e o professor responsável. Nesse construtor deve-se construir o objeto `alunos`, `notas` e `frequencia`. O valor do atributo `numTotalAulas` também deve ser configurado adequadamente, conforme descrito na descrição do construtor anterior).
- `matricularAluno(Aluno aluno):boolean` (Um método que permita matricular um aluno na disciplina. Ao matricular aluno deve-se registrar uma posição em `notas` e em `frequencias` iguais a zero, isto é, adicionar em `notas` e em `frequencias` o valor zero. O método retorna `true` se o aluno foi matriculado e `false`, caso contrário. Não poderão ser matriculados alunos com o mesmo valor de `ra`).

- `desmatricularAluno(int posAluno):boolean` (Um método que permite desmatricular um aluno. Este método deve receber como parâmetro a posição do aluno na lista (`posAluno`) a ser removido. Consulte na documentação da classe `ArrayList` um método que permita remover objetos do `ArrayList`. Você também deverá remover a nota do aluno na lista de notas e sua frequência na lista de frequência, caso contrário, ambas ficarão inconsistente. O método retorna `true` se o aluno foi desmatriculado e `false`, caso contrário).
- `setProfessor(Professor professor):void` (Um método que aloca à disciplina a um professor.).
- `removerProfessor():void` (Um método que remove um professor da disciplina. Uma disciplina não possui professor se o atributo `professor` é `null`).
- `getNumeroAlunos():int` (Um método que permite retornar o número de alunos matriculados na disciplina).
- `addNota(int posAluno, double nota):boolean` (Um método que permite configurar para o aluno na posição `posAluno` a sua nota (`nota`) na disciplina. Retorne verdadeiro ou falso se isso for possível. O valor para `nota` deve estar no intervalo [0,10]).
- `addFrequencia(int posAluno, int frequenciaTotal):boolean` (Um método que permite configurar para o aluno na posição `posAluno` da lista, a sua frequência total na disciplina. Retorne verdadeiro ou falso se isso for possível. O valor de `frequenciaTotal` deve estar no intervalo [0,`numTotalAulas`]).
- `estaAprovado(int posAluno):boolean` (Um método que permite verificar se o aluno na posição `posAluno` da lista está ou não aprovado na disciplina. O método retorna `true` se aprovado e `false`, caso contrário. Um aluno aprovado possui nota maior OU igual a 6 e frequência maior ou igual a 75%).
- `exibirReprovados():void` (Um método que permite exibir todos os alunos aprovados na disciplina. Se não houverem alunos reprovados, então deve-se imprimir `"Não há alunos reprovados."`. Se não houver alunos matriculados na disciplina imprimir `"Disciplina não possui alunos matriculados"`).
- `exibirAprovados():void` (Um método que permite exibir todos os alunos aprovados. Se não houverem alunos aprovados, então deve-se imprimir `"Não há alunos aprovados."`. Se não houver alunos matriculados na disciplina imprimir `"Disciplina não possui alunos matriculados"`).
- `calcularMedia():double` (Um método que retorna a média das notas de todos os alunos matriculados na disciplina. Esse valor é a média total da turma).
- `exibirMaiorNota():void` (Mostra as informações do aluno com a maior nota na disciplina).
- `exibirMenorNota():void` (Mostra as informações do aluno com a menor nota na disciplina).
- `getQuantidadeReprovados():int` (Método que retorna um inteiro que corresponde à quantidade de alunos reprovados.).
- `getQuantidadeAprovados():int` (Método que retorna um inteiro que corresponde à quantidade de alunos aprovados.).

- `exibirProfessor():void` (Método para mostrar as informações do professor responsável pela disciplina. Se a disciplina não houver um professor então deve-se exibir `"Disciplina não possui professor"`).
- `getAproveitamentoAluno(int posAluno):double` (Método que retorna o aproveitamento de um aluno na posição `posAluno` da lista. O aproveitamento do aluno é determinado a partir da sua frequência total. Um aluno que nunca faltou tem aproveitamento de 100% e um aluno que faltou metade do número total de aulas tem aproveitamento de 50%. Nesses casos o método retorna os valores 100 e 50, respectivamente).
- `getNotaAluno(int posAluno):double` (Método que retorna a nota de um aluno na posição `posAluno` da lista.).
- `exibirAlunosAcimaMedia():void` (Método para exibir todos os alunos acima da média total da turma. Por exemplo, se a média for 7.2, então todos os alunos com nota maior OU igual a 7.2 devem ser exibidos. Se não houver alunos matriculados na disciplina imprimir `"Disciplina não possui alunos matriculados"`).
- `exibirAlunosAbaixoMedia():void` (Método para exibir todos os alunos abaixo da média total da turma. Se não houver alunos matriculados na disciplina imprimir `"Disciplina não possui alunos matriculados"`).
- `exibirAlunoNotaAproveitamento():void` (Método para exibir apenas o nome dos alunos, suas respectivas notas e aproveitamento na disciplina. Se não houver alunos matriculados na disciplina imprimir `"Disciplina não possui alunos matriculados"`).
- `desmatricularAlunos():void` (Um método que permite desmatricular todos os alunos matriculados na disciplina).
- `exibirOrdenadosPorNota():void` (Um método para exibir apenas o nome dos alunos e suas respectivas notas. A exibição deverá ser feita de forma decrescente as nota, isto é, o aluno com a maior nota deverá ser o primeiro a ser exibido, seguido do aluno com a segunda maior nota e assim até o aluno com a menor nota. Se não houver alunos matriculados na disciplina imprimir `"Disciplina não possui alunos matriculados"`).
- `exibirDisciplina():void` (Um método para exibir todas as informações da disciplina. Deverá ser exibido as informações do professor e dos alunos, a nota e a frequência de cada aluno na disciplina. Imprimir a média total da turma, a quantidade de alunos a quantidade de alunos reprovados e aprovados. Quando não houver alunos matriculados imprimir apenas `"Disciplina não possui alunos matriculados"`).

No pacote `ifsuldeminas.main` deve-se implementar a classe `Main` e fazer as seguintes simulações:

1 Crie um objeto que represente uma disciplina do primeiro período. Configure o nome, o número de aulas na semana e o período da disciplina;

2 Crie 10 objetos `Aluno` do primeiro período;

3 Matricule todos os 10 objetos `Aluno` no objeto disciplina do passo 1;

4 Configure para cada objeto aluno uma nota e uma frequencia total;

5 Crie um objeto para representar um professor;

6 Aloque o professor ao objeto disciplina criado no passo 1.

7 Crie um objeto da classe `Scanner` que permita fazer leitura de dados do teclado;

8 Armazene em uma variável `opcao` um valor inteiro fornecido pelo teclado;

9 Enquanto `opcao` for diferente de zero. Permita que:

- `opcao == 1`: Imprimir informações da disciplina;
- `opcao == 2`: Apresentar número de alunos na disciplina, número de reprovados e número de aprovados;
- `opcao == 3`: Imprimir aprovados;
- `opcao == 4`: Imprimir reprovados;
- `opcao == 5`: Imprimir aluno com a maior e o aluno com menor nota;
- `opcao == 6`: Imprimir o professor responsável;
- `opcao == 7`: Imprimir média total da turma e listar todos os alunos acima da média;
- `opcao == 8`: Imprimir média total da turma e listar todos os alunos abaixo da média;
- `opcao == 9`: Imprimir nome dos alunos e suas respectivas notas e aproveitamento;
- `opcao == 10`: Imprimir alunos ordenados por nota.
- `opcao == 11`: Matricular aluno. Deve-se ler o nome, registro acadêmico e curso do aluno usando o objeto `Scanner`. Como a disciplina é uma disciplina do primeiro período, assume-se que o aluno é do primeiro período e, portanto, não há necessidades de ler seu período. Em seguida deve-se criar o objeto que represente o aluno e então matriculá-lo na disciplina; depois deve-se ler a nota e a frequência total do aluno na disciplina, esses valores deverão ser corretamente registrados para o aluno. Solicite ao usuário um valor válido a ser informado. Por exemplo: o registro acadêmico deve estar no intervalo [0, 9999], a nota deve estar no intervalo [0,10] e frequencia total não pode ser maior que o número total de aulas da disciplina. Apresente para o usuário se o aluno foi ou não matriculado com sucesso. Perceba que se o registro acadêmico fornecido pelo usuário for igual ao de um aluno já matriculado, então deverá ser exibido para o usuário que o aluno não foi matriculado, pois seu RA é semelhante ao de um aluno matriculado na disciplina.

10 Quando `opcao` for igual a zero, desmatricular todos os alunos, remover o professor da disciplina e finalizar a execução do programa.