Desafio - Kotlin

A folha de exercícios foi desenvolvida para ser resolvida por módulos. Os módulos estão ordenados para serem resolvidos progressivamente e são dependentes, ou seja, precisam ser resolvidos em ordem. O exercício deverá ser feito individualmente.

Arquivos a entregar:

- **Código;**
- **Um diagrama UML que contenha todos os relacionamentos apresentados no exercício.** Não é necessário declarar os construtores, getters e setters em um diagrama UML. Usar a seguinte ferramenta para entregar o UML:
    - **https://www.draw.io/**

Importante:

Todos os arquivos devem ser entregues em um repositório no GitHub e o link enviado aos professores

- Todos os enunciados devem ser realizados corretamente.

Data limite para entrega:

- **02/10/2020**

ESCLARECIMENTO:

Como estamos trabalhando com listas, eliminando e adicionando objetos, você deverá implementar o equals às classes que sejam necessárias. Além disso, se você quiser, poderá implementar o toString() desses objetos.

DigitalHouse >
Coding School

Parte A

1. Criar um diagrama de classes que modele a classe Aluno. A princípio, um aluno possui um nome **(String**), um sobrenome **(String)** e um código de aluno **(Integer)**.

2. Implementar a classe criando os atributos necessários.

3. Criar um construtor para o aluno que tome como parâmetro um nome, um sobrenome e um código de aluno.

4. Um aluno será igual a outro se seus códigos de aluno forem iguais.

DigitalHouse >
Coding School

Parte B

1. Criar um diagrama de classes que modele a classe Curso. A princípio, um curso possui um nome **(String)** e um código de curso **(Integer)**.

2. Implementar a classe criando os atributos necessários.

3. Um curso será igual a outro se seus códigos de curso forem iguais.

DigitalHouse >
Coding School

Parte C

1. Criar um diagrama de classes que modele a classe Professor. A princípio, um professor possui um nome **(String)**, um sobrenome **(String)**, um tempo de casa **(Integer)** e um código de professor **(Integer)**.

2. Implementar a classe criando os atributos necessários.

3. Um professor será igual a outro se seus códigos de professor forem iguais.

DigitalHouse >
Coding School

Parte D

Queremos adicionar duas categorias de professores ao modelo anterior. Os professores titulares e os professores adjuntos. Um professor titular tem uma especialidade **(String)** e um professor adjunto tem uma quantidade de horas de monitoria **(Integer)**.

1. Como você modificaria o diagrama de classe de Professor criado anteriormente?

2. Modificar a implementação considerando as novas alterações. Criar as classes que forem necessárias.

Parte E

Além de ter um nome e código de curso, um curso possui um professor titular **(ProfessorTitular)**, um professor adjunto **(ProfessorAdjunto)**, uma quantidade máxima de alunos **(Integer)** e uma lista de alunos matriculados.

1. Como você modificaria o diagrama de classe de Curso criado anteriormente?

2. Modificar a implementação considerando as novas alterações.

DigitalHouse >
Coding School

Parte F

1. Criar um diagrama de classes que modele a classe Matrícula. A princípio, uma matrícula tem um aluno **(Aluno)**, um curso **(Curso)** e uma data de matrícula **(Date)**.

2. Implementar a classe criando os atributos necessários.

3. Criar um construtor de Matrícula que tome um aluno e um curso e construa uma matrícula com a data do dia. A classe Date permite utilizar datas em Kotlin. Para criar a data do dia basta seguir o exemplo:

   Exemplo:

   ```
   var data = Date()
   ```

Parte G

1. Criar um diagrama de classes que modele a classe DigitalHouseManager. A princípio, DigitalHouseManager tem uma lista de alunos, uma lista de professores, uma lista de cursos e uma lista de matrículas

2. Implementar a classe criando os atributos necessários.

DigitalHouse >
Coding School

Parte H

1. Criar um método na classe **Curso** que permita adicionar um aluno à lista. O método retornará **true** se o aluno puder ser adicionado ou **false** caso não haja vagas disponíveis.

    - **fun adicionarUmAluno(umAluno: Aluno): Boolean**

2. Criar um método na classe **Curso** que permita excluir um aluno da lista de alunos do curso.
    - **fun excluirAluno(umAluno: Aluno)**

Parte I

1. Criar um método na classe **DigitalHouseManager** que permita registrar um curso. O método recebe como parâmetros o nome do curso, o código e a quantidade máxima de alunos admitidos. O método deve criar um curso com os dados correspondentes e adicioná-lo à lista de cursos.

    - **fun registrarCurso(nome: String, codigoCurso: Integer, quantidadeMaximaDeAlunos: Integer)**

2. Criar um método na classe **DigitalHouseManager** que permita excluir um curso. O método recebe como parâmetro o código do curso. O método deve utilizar o código do curso para encontrá-lo na lista de cursos e excluí-lo da lista.
    - **fun excluirCurso(codigoCurso: Integer)**

3. Criar um método na classe **DigitalHouseManager** que permita registrar um professor adjunto. O método recebe como parâmetros o nome do professor, o sobrenome, o código e a quantidade de horas disponíveis para monitoria. O tempo de casa inicial do professor será zero. O método deve criar um professor adjunto com os dados correspondentes e adicioná-lo à lista de professores.
    - **fun registrarProfessorAdjunto(nome: String , sobrenome: String , codigoProfessor: Integer, quantidadeDeHoras: Integer)**

4. Criar um método na classe **DigitalHouseManager** que permita registrar um professor titular. O método recebe como parâmetros o nome do professor, o sobrenome, o código e a especialidade. O tempo de casa inicial do professor

será zero. O método deve criar um professor titular com os dados correspondentes e adicioná-lo à lista de professores.

- **fun registrarProfessorTitular(nome: String , sobrenome: String, codigoProfessor: Integer, especialidade: String)**

5. Criar um método na classe **DigitalHouseManager** que permita excluir um professor. O método recebe como parâmetro o código do professor. O método deve utilizar o código do professor para encontrá-lo na lista de professores e eliminá-lo da lista.
    - **fun excluirProfessor(codigoProfessor: Integer)**

6. Criar um método na classe **DigitalHouseManager** que permita registrar um aluno. O método recebe como parâmetros o nome, o sobrenome e o código do aluno. O método deve criar um aluno com os dados correspondentes e adicioná-lo à lista de alunos.

    - **matricularAluno(nome: String, sobrenome: String, codigoAluno: Integer)**

7. Criar um método na classe **DigitalHouseManager** que permita matricular um aluno em um curso. O método recebe como parâmetros o código do aluno e o código do curso em que ele está se matriculando.
    - **matricularAluno(codigoAluno: Integer, codigoCurso: Integer)**

    O método deve:
    - Encontrar o curso em que o aluno está se matriculando.
    - Encontrar o aluno que queremos matricular.
    - Matricular o aluno, se for possível.
    - No caso de ser possível, criar uma matrícula e configurá-la com os dados correspondentes.
        - Adicionar a matrícula à lista de matrículas.

    - Informar na tela que a matrícula foi realizada.
- Se não houver vagas disponíveis:
    - Informar na tela que não foi possível realizar a matrícula porque não há vagas.

8. Criar um método na classe **DigitalHouseManager** que permita alocar professores a um curso. O método recebe como parâmetros o código do curso, o código do professor titular e o código do professor adjunto.

    - **fun alocarProfessores(codigoCurso: Integer, codigoProfessorTitular: Integer, codigoProfessorAdjunto: Integer)**

    O método deve:
    - Encontrar o professor titular na lista de professores.
    - Encontrar o professor adjunto na lista de professores.
    - Alocar ambos professores ao curso.

Parte J

1. Criar uma classe Principal e a função main.

2. Registrar dois professores titulares e dois professores adjuntos. (Inventar todos os seus valores)

3. Registrar dois cursos.

    - **Nome do curso: Full Stack**
      Código do curso: 20001
      Quantidade máxima: 3

    - **Nome do curso: Android**
      Código do curso: 20002
      Quantidade máxima: 2

4. Alocar um professor titular e um adjunto para cada curso.

5. Matricular dois alunos no curso de Full Stack.

6. Matricular três alunos no curso de Android.

Parte K

1. Como você modificaria o diagrama de classes para que um aluno possa consultar em que curso se matriculou?