

|  |
| --- |
| CURSO |
| **MOBILE ANDROID** |

|  |
| --- |
| DESAFIO |
| DESAFIO - KOTLIN  **DATA DE ENTREGA: 02/10/2020**  **PROFESSORES:**  **O diagrama e o link do repositório com o código deverão ser entregues por email para os professores** |

|  |
| --- |
| EXPECTATIVA |
| Espera-se que o aluno consiga executar as seguintes atividades:  **Documentação**   * Diagrama UML que contenha todos os relacionamentos apresentados no exercício. Não é necessário declarar os construtores diagrama UML. * Usar a seguinte ferramenta para desenvolvimento do UML [Draw.io](https://www.draw.io/)  Lembre-se que assim que terminar o diagrama você deve exportá-lo como PDF ou imagem(.jpg) para ser entregue   **Desenvolvimento**   * Diagrama de classe com os relacionamentos do sistema * Cadastro e alocação de professores titulares e adjuntos, cadastro de cursos e matrícula de alunos * Tratamento de exceção * Organização e estrutura de código   **Organização**   * Trabalhar com GitHub * Trabalhar com GIT |

|  |
| --- |
| ENUNCIADO |
| * Criar a classe Aluno que deverá conter:   + Propriedades: nome, sobrenome e um código de aluno * Criar a classe Professor que deverá conter:   + Propriedades: nome, sobrenome, tempo de casa e um código do professor * Criar a classe Professor Titular:   + Propriedades: especialidade * Criar a classe Professor Adjunto:   + Propriedades: quantidade de horas de monitoria * Criar a classe Curso que deverá conter:   + Propriedades: nome do curso, código do curso, professor titular, professor adjunto, quantidade máxima de alunos, lista de alunos matriculados   + Métodos:   Criar uma função 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  Criar um método na classe Curso que permita excluir um aluno da lista de alunos do curso.  fun excluirAluno(umAluno: Aluno)   * Criar a classe Matricula:   + Propriedades: aluno, curso e data de matrícula   + Observação: Ao criar a instância de um objeto já deve ser inicializado uma matrícula contendo aluno, curso e a data da matrícula. Para recuperar a data atual você pode estar usando o LocalDateTime.now() * Criar a classe DigitalHouse Manager:   + Propriedades: lista de alunos, lista de professores, lista de cursos, lista de matrículas.   + Métodos:  Criar uma função na classe DigitalHouse Managerque permite 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: Int, quantidadeMaximaDeAlunos: Int)  Criar uma função na classe DigitalHouse Managerque permite excluir um curso. A função recebe como parâmetro o código do curso. A função deve utilizar o código do curso para encontrá-lo na lista de cursos e excluí-lo da lista.  fun excluirCurso(codigoCurso: Int)  Criar uma função na classe DigitalHouse Managerque permite registrar um professor adjunto. A função 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. A função deve criar um professor adjunto com os dados correspondentes e adicioná-lo à lista de professores.  registrarProfessorAdjunto(nome: String, sobrenome: String, codigoProfessor: Int, quantidadeDeHoras: Int)  Criar um método na classe DigitalHouse Managerque 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: Int, especialidade: String )   * + Criar um método na classe DigitalHouse Managerque permite 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: Int)   * + Criar um método na classe DigitalHouse Managerque 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.   fun matricularAluno(nome: String , sobrenome: String , codigoAluno: Int)  Criar um método na classe DigitalHouse Managerque permite 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.  fun matricularAluno(codigoAluno: Int, codigoCurso: Int)  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.   Criar um método na classe DigitalHouse Managerque 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: Int, codigoProfessorTitular: Int, codigoProfessorAdjunto: Int)  O método deve:   * Encontrar o professor titular na lista de professores. * Encontrar o professor adjunto na lista de professores. * Alocar ambos professores ao curso. |