Prof. Leandro M. Nascimento – UFRPE

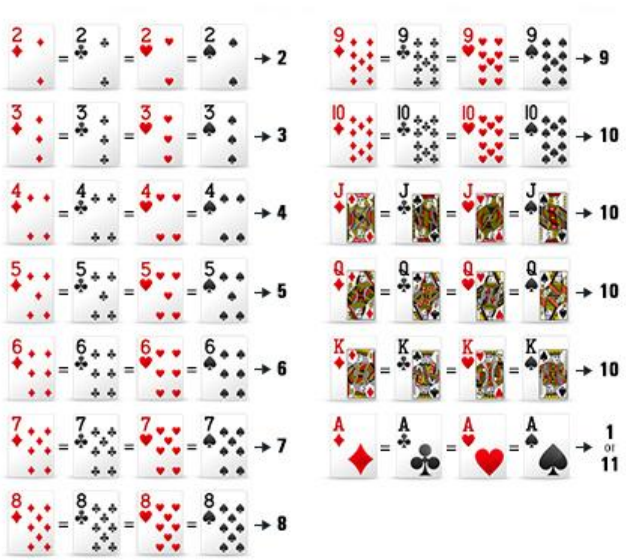
OBSERVAÇÃO 1: para cada questão a ser resolvida em sua lista de exercícios, crie um pacote com o nome "questaoXX", onde o XX deve ser substituído por 01, 02, 03, etc.
OBSERVAÇÃO 2: Espera-se que você use os seus conhecimentos de orientação a objetos e boas práticas de programação para resolver as questões. Não é preciso dizer que seu código NÃO DEVE ser um bloco monolítico de código com todas as instruções e variáveis dentro um único método *main* que executa todo o algoritmo.
OBSERVAÇÃO 3: Plágio não será tolerado, podendo resultar na anulação da pontuação da lista.

1) (4,0) Blackjack

Um dos jogos mais populares em diversos cassinos ao redor do mundo é o Blackjack, em português, o famoso "Vinte e um".

O jogo consiste de uma batalha da banca (*dealer* ou crupiê) contra 1 ou mais adversários (máximo de 6) envolvendo uma regra simples de comparação de cartas: quem tiver as cartas cuja soma dos seus valores seja o mais próximo de 21 ganha e, se a soma ultrapassar 21, você perde automaticamente.

No jogo, as cartas apresentam os seguintes valores para:



A mecânica do jogo é a seguinte:

1. Antes de receber as cartas o jogador decide quanto quer apostar a partir de uma aposta mínima. A aposta mínima é obrigatória caso o jogador não queira apostar naquela rodada.
2. A banca recebe uma carta visível publicamente e entrega duas cartas também visíveis para o jogador que decide então entre duas ações: 1) se quer receber mais cartas (chamado de

pedir ou *hit* em inglês) ou 2) se quer parar (chamado de *ficar* ou *stay* em inglês). O jogador pode receber quantas cartas quiser se a soma de seus valores não ultrapassar 21 (*estourar*). Se a soma ultrapassar, o jogador perde automaticamente aquela rodada.

3. Uma vez escolhida a ação de *ficar*, a banca então inicia suas jogadas, tentando ultrapassar o valor do jogador. Se ultrapassar sem estourar 21 pontos, a banca ganha. Se estourar, a banca perde.
4. Na situação em que o jogador perde, o valor apostado vai para a banca. Quando o jogador ganha, a banca paga para ao jogador o mesmo valor que ele apostou (1:1).

Para entender melhor a mecânica do jogo, acesse http://www.blackjack.org/free-online-blackjack-game/ e veja como funciona. As regras podem ser encontradas aqui: http://pt.blackjack.org/regras-do-blackjack/

Você como experiente programador, tem a missão de implementar um jogo de *blackjack* em **Java** seguindo as regras oficiais citadas anteriormente e considerando as seguintes premissas:

1. O jogador irá começar com o valor de 100 fichas para apostar.
2. A aposta mínima para todas as disputas será de 2 fichas, podendo o usuário aumentar essa aposta até o valor máximo de todas as fichas que possui (*all-in*)
3. O jogo deve começar perguntando quantos decks de baralho serão usados para a partida que irá se seguir e deve restringir essa quantidade para um mínimo de 2 e máximo de 10 decks.
4. Durante a execução do jogo, a carta jogada não deve voltar mais ao conjunto de cartas que ainda podem ser usadas. Isto é, você começa o jogo e escolhe 2 decks de baralho para as disputas que se seguem, que equivalem a 104 cartas. Diversas disputas depois, pode não haver mais cartas a serem usadas e somente aí as 104 cartas voltam para o montante novamente e são embaralhadas.
5. A cada nova disputa, a quantidade de cartas restantes no deck deve ser apresentada.
6. O jogo termina quando o jogador perde todas as suas fichas ou quando decide parar de jogar.
7. Ao término de uma partida, que inclui várias disputas, a quantidade final de fichas deve ser apresentada informando se o jogador ganhou ou perdeu em relação à quantidade inicial de fichas, seguida de uma mensagem de incentivo, por exemplo: “PARABÉNS!!! Você ganhou 25 fichas” ou “Que pena, você perdeu 35 fichas. Tente outra vez!”

Observações: você **NÃO precisa** implementar nenhuma regra adicional de blackjack, como por exemplos, regras relacionadas à divisão de apostas ou ao conhecido “duplo”.

A seguir você visualiza um exemplo de execução do programa e como você deveria seguir na solução:

```
Bem-vindo ao Blackjack Rural:
Legenda dos naipes: P – PAUS / C – COPAS / E – ESPADAS / O - OUROS

Quantos decks de baralho, você deseja utilizar nessa partida:
2 (input de usuário)
Iniciando partida...
=============================================
Disputa n. 1
Quantidade de cartas no montante: 104
Quantidade de fichas: 100
Digite sua aposta (mínimo de 2) ou ‘X’ para sair do jogo:
```

```
8 (input de usuário)

Essas são as cartas da banca
K-E
Total de pontos: 10
------x------
Essas são suas cartas:
A-C | 8-O
Total de pontos: 19

O que você deseja fazer?
(H) HIT - para pedir mais cartas ou
(S) STAY – para parar e deixar a banca jogar
S (input de usuário)

Essas são as cartas da banca
K-E | 2-O | 10-P
Total de pontos: 22
------x------
Essas são suas cartas:
A-C | 8-O
Total de pontos: 19

RESULTADO: A banca estourou 21 pontos
           Você ganhou e recebeu 8 fichas!

=============================================
Disputa n. 2
Quantidade de cartas no montante: 99
Quantidade de fichas: 108
Digite sua aposta (mínimo de 2) ou 'X' para sair do jogo:
10 (input de usuário)

Essas são as cartas da banca
J-C
Total de pontos: 10
------x------
Essas são suas cartas:
7-C | 7-E
Total de pontos: 14

O que você deseja fazer?
(H) HIT - para pedir mais cartas ou
(S) STAY – para parar e deixar a banca jogar
H (input de usuário)
Carta sorteada: 3-C


Essas são as cartas da banca
J-C
Total de pontos: 10
------x------
Essas são suas cartas:
7-C | 7-E | 3-C
Total de pontos: 17
O que você deseja fazer?
(H) HIT - para pedir mais cartas ou
(S) STAY – para parar e deixar a banca jogar
S (input de usuário)

Essas são as cartas da banca
J-C | A-O
```

```
Total de pontos: 21
------x------
Essas são suas cartas:
7-C | 7-E | 3-C
Total de pontos: 17

RESULTADO: A banca ganhou e você perdeu 10 fichas!

==============================================
Disputa n. 3
Quantidade de cartas no montante: 94
Quantidade de fichas: 98
Digite sua aposta (mínimo de 2) ou 'X' para sair do jogo:
6 (input de usuário)

Essas são as cartas da banca
4-E
Total de pontos: 4
------x------
Essas são suas cartas:
9-C | 6-P
Total de pontos: 15
O que você deseja fazer?
(H) HIT - para pedir mais cartas ou
(S) STAY - para parar e deixar a banca jogar
H (input de usuário)
Carta sorteada: 8-O

Essas são as cartas da banca
4-E
Total de pontos: 4
------x------
Essas são suas cartas:
9-C | 6-P | 8-O
Total de pontos: 23

RESULTADO: Você estourou os 21 pontos.
           A banca ganhou e você perdeu 6 fichas!

==============================================
Disputa n. 4
Quantidade de cartas no montante: 90
Quantidade de fichas: 92
Digite sua aposta (mínimo de 2) ou 'X' para sair do jogo:
2 (input de usuário)


# O jogo continua até o usuário perder todas as fichas ou decidir parar
```

2) (3,5) Você é o novo gerenciador de contas do YouTube, e agora?

Você como exímio *full-stack developer* da UFRPE foi contratado para implementar um sistema de controle das mídias providas pelo YouTube e precificação dos valores a serem pagos aos produtores de conteúdo, que aumentam suas receitas quanto mais visualizações alguma de suas mídias produzidas tiver.

O sistema deve gerenciar todos os produtores de conteúdo, as mídias adicionadas por eles e contabilizar cada reprodução dessas mídias, feitas por qualquer outro usuário que não seja o produtor daquele conteúdo.

Um usuário apresenta as informações de nome completo, data de nascimento e e-mail, que o identifica unicamente no sistema. Um usuário deve ser definido como uma entidade abstrata, pois não deverá ser instanciado. Não deverão existir no sistema dois usuários que tenham o mesmo e-mail. Duas entidades concretas herdam diretamente da entidade usuário (produtores e consumidores de conteúdo). Uma mídia representa um objeto que mantém o nome do seu arquivo, tem uma faixa etária mínima para poder ser reproduzido e a data/hora em que foi produzida (data do upload para o sistema). A entidade que representa a reprodução de uma mídia deve manter o usuário que a reproduziu, a mídia reproduzida e data/hora em que foi reproduzida. As entidades apresentam os seguintes atributos (você decide quais construtores definir e métodos get/set estão omitidos, mas devem ser implementados):

- **Usuario**: representa um usuário abstrato
  - email: String representa unicamente um usuário no sistema
  - nomeCompleto: String nome completo do usuário no sistema
  - dataNascimento: java.time.LocalDate data de nascimento do usuário (utilizada para calcular sua idade e mídias que são apropriadas para sua faixa etária)
- **Produtor**: herda de "Usuario" e representa um usuário que produz e faz upload de mídias em um determinado canal
  - nomeCanal: String nome do canal que o produtor gerencia
  - categorias: List<String> lista de categorias que aquele canal do produtor se enquadra
- **Consumidor**: herda de "Usuario" e representa um usuário comum que consome (reproduz) mídias
  - categoriasInteresse: List<String> lista de categorias de interesse do usuário consumidor
- **Midia**: representa uma mídia (normalmente vídeo) que pode ser reproduzida
  - dataHoraUpload: java.time.LocalDateTime a data em que a mídia foi subida (*uploaded*) para o sistema.
  - arquivo: String nome do arquivo que aquela mídia representa.
  - faixaEtariaMinima: int faixa etária mínima para a reprodução da mídia. Se a classificação etária for livre, este deve valor deverá assumir um valor negativo.
  - categoria: String categoria na qual a mídia se encaixa.
  - Produtor: Produtor usuário que produziu a mídia.
- **ReproducaoMidia**: representa o momento que um usuário consumidor realiza a reprodução de alguma mídia num dado instante
  - midia: Midia objeto do tipo "Midia" reproduzido.
  - consumidor: Consumidor objeto do tipo "Consumidor" que reproduziu a mídia.
  - dataHoraReproducao: java.time.LocalDateTime data/hora de reprodução da mídia

A decisão de implementar métodos 'equals' e 'toString' em cada uma das classes básicas descritas anteriormente é exclusivamente sua de acordo com as necessidades do sistema.

O sistema deve ser estruturado em camadas, seguindo o estilo arquitetural MVC, com três repositórios, um controlador único e uma classe main que testa as funcionalidades do sistema.

Os três repositórios devem ser: RepositorioUsuario, RepositorioMidia, RepositorioReproducaoMidia. Cada repositório deve ter as seguintes funcionalidades e podem ser implementados com ArrayList:

- **RepositorioUsuario**:
  - void cadastrarUsuario(Usuario u): salva um novo "Usuario" que pode ser Consumidor ou Produtor. Não permite que dois ou mais usuários tenham o mesmo e-mail
  - void removerUsuario(Usuario u): remove instância do usuário passado como parâmetro. Você pode testar pelo login para remover.
  - List<Usuario> listarUsuariosComIdadeAcimaDe(int idade): lista todos os usuários com idade acima da informada como parâmetro
  - List<Usuario> listarUsuariosPorTipo(Class tipo): lista todos os usuários que são daquele tipo passado como parâmetro.
- **RepositorioMidia**:
  - void cadastrarMidia(Midia m): salva uma nova mídia no sistema. Não pode haver duas mídias que tenham o mesmo nome de arquivo.
  - void removerMidia(Midia m): remove instância da mídia passada como parâmetro. Você pode testar pelo nome do arquivo para remover.
  - List<Midia> listarMidiasPorFaixaEtaria(int faixaEtaria): retorna todas as mídias que tem faixa igual ou inferior à faixa etária passada como parâmetro.
  - List<Midia> listarMidiasPorCategoria(String categoria): retorna todas as mídias que tem a categoria igual à passada como parâmetro.
- **RepositorioReproducaoMidia**:
  - void cadastrarReproducaoMidia(ReproducaoMidia rm): salva uma nova reprodução de uma mídia realizada por um usuário consumidor
  - List<ReproducaoMidia> listarReproducoesNoPeriodo(LocalDateTime inicio, LocalDateTime fim): listar todas as reproduções que ocorreram dentro do intervalo de data/hora informado como parâmetro (inclusivo)
  - List<ReproducaoMidia> listarReproducoesPorUsuario(Consumidor usuario): listar todas as reproduções que foram feitas por um determinado usuário.
  - List<ReproducaoMidia> listarReproducoesPorCategorias(List<String> categorias): listar todas as reproduções que estão categorizadas dentro de alguma das categorias da lista passada como parâmetro.

Uma vez criados os repositórios, você deve criar um, e somente um, controlador responsável por gerenciar todas as funcionalidades do sistema, acessando todos os repositórios, que deve se chamar ControladorMidia, com as premissas:

- A classe deve apresentar 3 atributos do tipo de cada um dos repositórios
- A classe deve implementar o padrão de projeto *Singleton*
- A classe deve apresentar todos os métodos do tipo *delegate* para todos métodos dos repositórios que retornem algum tipo de lista (List<?>)

Além disso, a classe deve apresentar os seguintes métodos:

- **ControladorMidia**:
  - void cadastrarUsuario(Usuario u): valida se usuário não é nulo e salva no repositório correspondente. Se o usuário for do tipo Produtor, não permitir que o mesmo tenha sua lista de categorias nula ou vazia.

- void removerUsuario(Usuario u): valida se usuário não é nulo e remove do repositório correspondente.
- void cadastrarMidia(Midia m): valida se mídia não é nula e salva no repositório correspondente. Validar se a categoria da mídia está contida na lista de categorias do Produtor e, se não estiver, não adicionar no repositório.
- void removerMidia(Midia m): valida se mídia não é nula e remove através do repositório correspondente.
- void reproduzirMidia(Consumidor consumidor, Midia midia): este método checa algumas regras de negócio e cria o objeto ReproducaoMidia, adicionando posteriormente ao repositório e representando a real reprodução da mídia. As regras são:
  - Parâmetros 'consumidor' e 'midia' não podem ser nulos
  - Validar se a idade do consumidor é maior ou igual à faixa etária da mídia.
  - Criar o objeto do tipo ReproducaoMidia associando o consumidor, a mídia e a data atual e então salvar no repositório.

Após a criação do controlador descrito anteriormente, você deve criar uma classe de testes que execute as seguintes instruções:

- Crie 3 objetos do tipo Produtor e cadastre-os usando o controlador
- Crie 7 objetos do tipo Consumidor e cadastre-os usando o controlador
- Remova o último objeto Consumidor cadastrado usando o controlador.
- Recupere a lista de todos os usuários acima de 16 anos usando o controlador e imprima-os na tela
- Recupera a lista de todos os usuários do tipo Consumidor usando o controlador e imprima-os na tela
- Crie 8 objetos do tipo Midia, associando-os aos objetos do tipo Produtor criados anteriormente e cadastre-os usando o controlador.
- Remova o último objeto do tipo Midia adicionado usando o controlador
- Recupere a lista de todas as mídias com faixa etária acima de 14 anos usando o controlador e imprima-as na tela
- Recupere a lista de todas as mídias de uma categoria qualquer usando o controlador e imprima-as na tela
- Execute pelo menos 20 chamadas diferentes ao método reproduzirMidia do ControladorMidia, passando como argumentos diversas combinações dos objetos do tipo Consumidor e Midia criados. Dentre essas chamadas, tente criar alguma chamada inválida com idade do consumidor incompatível com a faixa etária mínima da mídia.
- Recupere a lista de todos os objetos do tipo ReproducaoMidia executados dentro de um determinado intervalo de datas inicial e final e imprima-os na tela
- Recupere a lista de todos os objetos do tipo ReproducaoMidia executados por um usuário qualquer e imprima-os na tela
- Recupere a lista de todos os objetos do tipo ReproducaoMidia executados dentro de um determinado intervalo de datas inicial e final e imprima-os na tela
- Recupere a lista de todos os objetos do tipo ReproducaoMidia que se encaixam em uma determinada lista de categorias definidas por você e imprima-os na tela.

3) (3,5) REC'n'Play

O REC'n'Play é onde a galera se encontra para reprogramar o mundo. Um festival no Bairro do Recife com mais de 300 atividades. São 4 dias de shows, experiências, palestras e oficinas, nas áreas de Tecnologia, Economia Criativa e Cidades Inteligentes. Você, que é uma pessoa que gosta de reprogramar o mundo, foi convidado pelo Departamento de Computação da UFRPE para implementar o sistema de inscrições e avaliação desse grande evento. Para isso, os analistas de sistema enviaram um diagrama de classes em UML, juntamente com uma descrição dos requisitos desejados, conforme especificado abaixo.

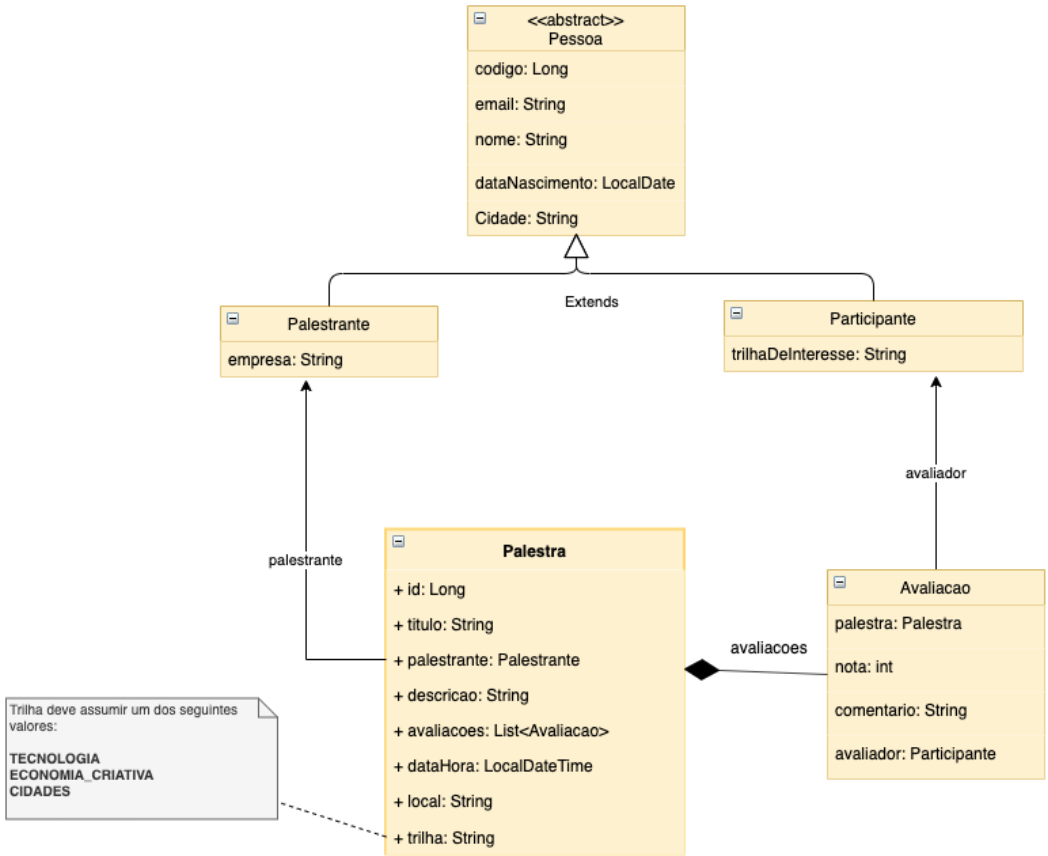


Figura 1 - Diagrama de classes do Rec'n'play

Requisitos

O sistema deve ser estruturado em camadas, seguindo o estilo arquitetural MVC, com repositórios, um controlador único e uma classe main que testa as funcionalidades do sistema.

Os repositórios devem ser: RepositorioPessoas e RepositorioPalestras. Cada repositório deve ter as seguintes funcionalidades e podem ser implementados com ArrayList:

- **RepositorioPessoas**:
  - void inserirPessoa(Pessoa pessoa): salva uma nova "Pessoa" que pode ser Palestrante ou Participante. Não permite que duas ou mais pessoas tenham o mesmo e-mail.
  - void removerPessoa(String email): remove instância da pessoa que possui o email passado como parâmetro.

  - Pessoa buscarPessoaPorEmail (String email): retorna a pessoa que possui o email passado como parâmetro
  - List<Palestrante> listarPalestrantesPorEmpresa (String empresa): lista todos os palestrantes da empresa passada como parâmetro
  - List<Participante> listarParticipantesComInteresseEm(String trilha): lista todos os participantes cuja trilha de interesse é igual à passada como parâmetro.
  - List<Participante> listarParticipantesComIdadeMairQue(int idade): lista todos os participantes com idade igual ou superior à passada como parâmetro.
- **RepositorioPalestras**:
  - void inserirPalestra(Palestra p): salva uma nova palestra no sistema. Não pode haver duas palestras com o mesmo id.
  - void buscarPalestraPorId(int id):retorna a palestra cujo id é o informado como parâmetro.
  - List<Palestra> listarPalestras (): retorna todas as palestras cadastradas
  - Palestra buscarPalestrasPorLocalEHorario(String local, LocalDateTime horário): retorna a palestra com o horário e local passados como parâmetro.
  - List<Palestra> listarPalestrasComMediaAvaliacaIgualOuMaiorQue(int nota): retorna todas as palestras que possuem a média de notas de suas avaliações igual ou maior que a nota passada como parâmetro.
  - List<Avaliacao> listarTodasAvaliacoes (): retorna todas as avaliações de todas as palestras cadastradas.

Uma vez criados os repositórios, você deve criar um, e somente um, controlador responsável por gerenciar todas as funcionalidades do sistema, acessando todos os repositórios. Tal controlador deve se chamar **ControladorRECnPlay**, com as seguintes premissas:

- A classe deve possuir 2 atributos – um do tipo RepositorioInscricoes e outro do tipo RepositorioPalestras
- A classe deve implementar o padrão de projeto *Singleton*
- A classe deve apresentar todos os métodos do tipo *delegate* para todos métodos dos repositórios que retornem algum tipo de lista (List<?>)

Além disso, a classe deve apresentar os seguintes métodos:

- **ControladorRECnPlay**:
  - void realizarInscricao(Participante p): valida se o participante não é nulo e salva no repositório de pessoas. Não permitir que o participante tenha sua trilha de preferência vazia ou nula. E verificar se a mesma possui um dos seguintes valores: `TECNOLOGIA`, `ECONOMIA_CRIATIVA` ou `CIDADES`

  - void cadastrarPalestra(Palestra p): valida se a palestra não é nula e se possui um palestrante informado, bem como todos os outros atributos. Não será permitido cadastrar palestras com o mesmo título. Antes de salvar a palestra no repositório de palestras, salve no repositório de pessoas o palestrante contido na palestra recebida como parâmetro, validando se a empresa do mesmo não é nula. Validar se existe alguma palestra com o mesmo horário/local já cadastrada e não realizar o cadastro em caso positivo.

- void avaliarPalestra(Avaliação aval): Verificar se o avaliador e a palestra existem no sistema e recusar a avaliação em caso negativo. A nota da avaliação deve ser entre 0 e 5.
- List<Palestra> listarPalestrasMaisBemAvaliadas(): retorna lista de Palestras que possui a maior média de notas de avaliação.
- int listarTotalPalestrasComMediaAvaliacaoMaiorQue(int nota): retorna o total de palestras com média de avaliação maior que a nota informada como parâmetro.

Após a criação do controlador descrito anteriormente, você deve criar uma classe de testes que execute as seguintes instruções:

- Cadastre 5 objetos do tipo Palestra usando os dados abaixo:

Data: 02/10/2023 - 10:00
Título: A inovação financeira e o sistema financeiro do futuro
Palestrante: GUSTAVO FRANCO
Empresa: Empresa X
Local: Teatro Apolo
Trilha: Tecnologia
Descrição: Palestra sobre como as inovações tecnológicas financeiras podem influenciar o sistema financeiro

Data: 02/10/2023 - 11:15
Título: Apresentação do Humanóide NAO
Palestrante: Simone Zelaquett
Empresa: Accenture
Local: Accenture Innovation Center
Trilha: Tecnologia
Nessa atividade os participantes assistirão a performance do NAO- humanóide programado por estudantes da rede municipal do Recife, am apresentações de Yoga e Dança que serão contagiantes e motivarão a interação dos presentes.

Data: 02/10/2023 - 10:00
Título: Imprensa Mirim
Palestrante: Andrea Pinho
Empresa: Prefeitura do Recife
Local: Teatro Apolo
Trilha: Economia Criativa
Descrição: Durante o evento os palestrantes serão entrevistados como pauta da Cobertura jornalística das ações do Rec'n'Play , realizadas por estudantes da Prefeitura de Recife. Muitos participantes poderão compor com os palestrantes as entrevistas prestando depoimentos sobre o evento e suas impressões e novas aprendizagens.

Data: 04/10/2023 - 10:00
Título: Educação experimental para famílias empresárias
Palestrante: Mariana Moura
Empresa: Grupo Moura
Local: CESAR Bom Jesus - Sala Garagem (5º andar)
Trilha: Cidades
Uma conversa com os sócios e membros da Escola F para debater um novo modelo educacional voltado para famílias empresárias. Um modelo de negócio colaborativo que valoriza a aprendizagem a partir da troca de experiências, investe na facilitação de grupos para gerar conhecimento e desenvolvimento, por meio da construção de entendimentos e não apenas a partir de conteúdos externos.

Data: 05/10/2023 - 10:00
Título: Como você e a sua empresa irão sobreviver ao apocalipse digital?
Palestrante: SILVIO MEIRA
Empresa: UFPE
Local: Cais do Sertão - Auditório
Trilha: Cidades
Talk Show com o renomado líder do setor de inovação que conversá sobre o tema de transformação digital.

- Caso dê algum erro no cadastro de alguma palestra, altere os dados e recadastre-a

- Realize a inscrição de 8 participantes

- Liste os participantes com idade maior que 39 anos
- Cadastre 3 avaliações diferentes para cada uma das palestras
- Liste o total de palestras com média de avaliação maior que 3
- Lista as palestras mais bem avaliadas

Let the coding begin!