MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO TRIÂNGULO MINEIRO
CAMPUS UBERLÂNDIA CENTRO

TECNOLOGIA EM SISTEMAS PARA INTERNET
Turma: **3º PERÍODO**
Unidade Curricular: **PROGRAMAÇÃO ORIENTADA A OBJETOS**
Professor: **WILL ROGER PEREIRA**

LISTA 2-8

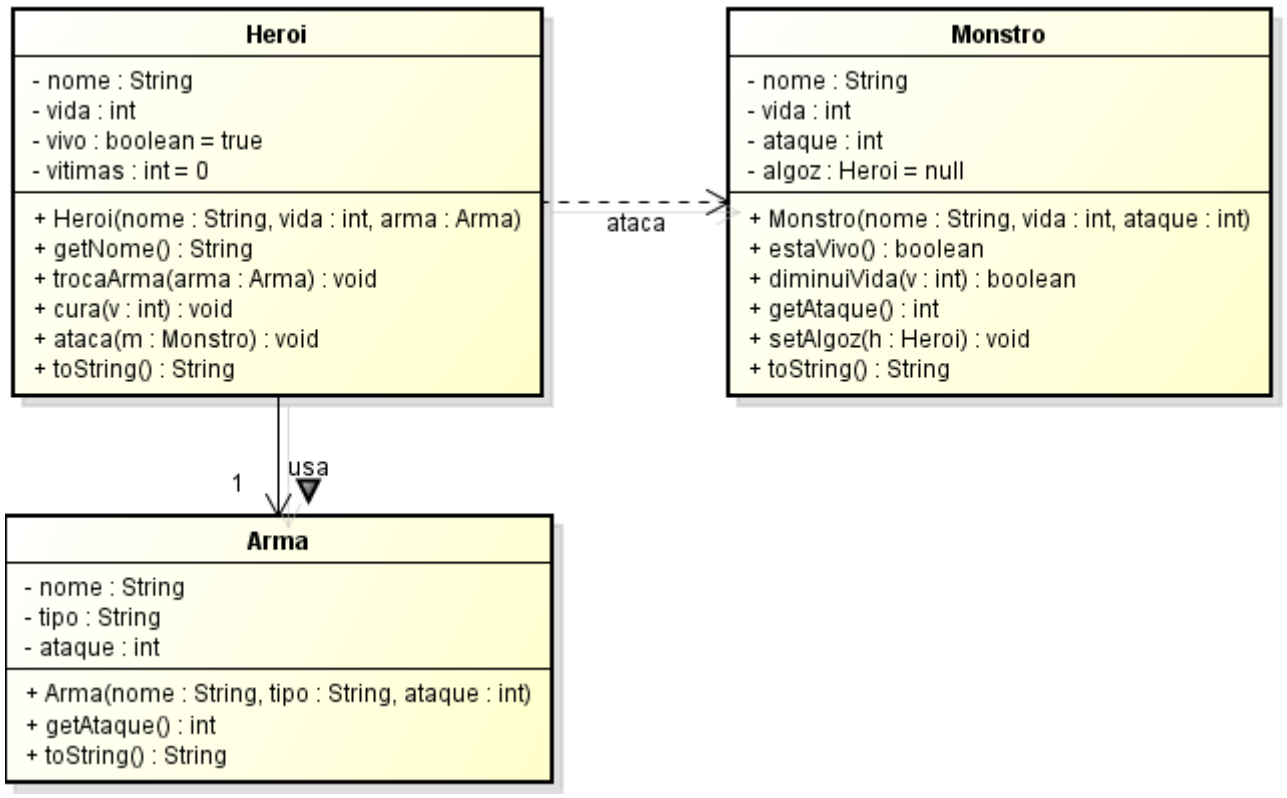
Obs: Para todos os exercícios, proceda conforme a aula. Construa objetos, contemple a multiplicidade e execute os métodos.

Obs2: As especificações e/ou restrições para os valores dos atributos sempre se encontrarão neles!!! Caso este valor esteja fora das especificações dentro de um método, sempre mostre uma mensagem de erro. No caso dos construtores, caso aconteça algum problema com os atributos, atribua valores padrões.

Obs3: O levantamento de restrições também é de sua responsabilidade. Portanto, sempre que encontrar alguma irregularidade na execução de um método, informe este erro.

Obs4: LEIA, NA ÍNTEGRA, A DESCRIÇÃO DE TODOS OS ATRIBUTOS E MÉTODOS.

1ª Questão



Classe Arma: É uma descrição da arma. Dois heróis podem usar a mesma arma, mas serão itens diferentes.
- nome : String → Nome da Arma. Não pode ser uma String vazia.
- tipo : String → Tipo da Arma, e.g. espada, machado, arco e flecha, adaga. Não pode ser uma String vazia.
- ataque : int → Ataque da Arma. Dano que ela causará a um Monstro. Deve ser um número positivo.

+ Arma(nome : String, tipo : String, ataque : int) → Construtor.

+ getAtaque() : int → Retorna o ataque da Arma.
+ toString() : String → Retorna as informações da Arma, retornando seus atributos.

Classe Monstro:
- nome : String → Nome do Monstro. Não pode ser uma String vazia.
- vida : int → Vida do Monstro. Deve ser um número positivo.
- ataque : int → Ataque do Monstro. Dano que ele causará a um Heroi. Deve ser um número positivo.
- algoz : Heroi = null → Algoz do Monstro. Caso seja nulo, o Monstro está vivo.

+ Monstro(nome : String, vida : int, ataque : int) → Construtor.

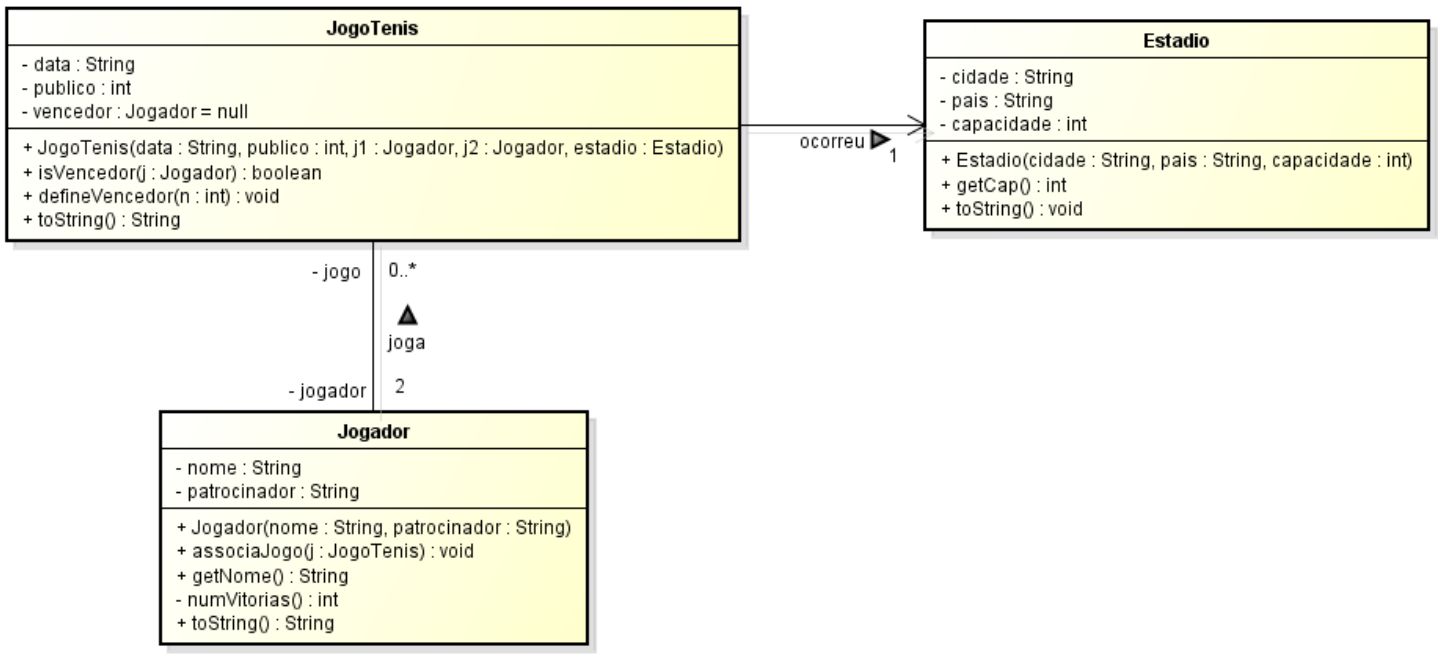
+ estaVivo() : boolean → Retorna **true** caso o Monstro não possua um assassino, e **false** caso esteja vivo.
+ diminuiVida(v : int) : boolean → Diminui a vida do Monstro de acordo com o argumento, e retorna **true** caso o Monstro seja morto e **false** caso continue vivo.
+ getAtaque() : int → Retorna o ataque do Monstro.
+ setAlgoz(h : Heroi) : void → O Heroi argumento torna-se o algoz do Monstro, mostrando que o Monstro está morto.
+ toString() : String → Retorna as informações do Monstro, retornando seu nome, vida, ataque e o nome de seu algoz, caso houver.

Classe Heroi:
- nome : String → Nome do Heroi. Não pode ser uma String vazia.
- vida : int → Vida do Heroi. Deve ser um número positivo.
- vivo : boolean = true → Indica se o Heroi está vivo ou não.
- vitimas : int = 0 → Indica quantos Monstros foram vítimas do Heroi.

+ Heroi(nome : String, vida : int, a : Arma) → Construtor.

+ getNome() : String → Retorna o nome do Heroi.
+ trocaArma(arma : Arma) : void → Troca a Arma usada pelo argumento. Mostre uma mensagem de erro se as Armas forem as mesmas.
+ cura(v : int) : void → Aumenta a vida do Heroi de acordo com o argumento. O argumento deve ser um valor positivo.
+ ataca(m : Monstro) : void → Ataca o Monstro presente no argumento com sua Arma. Um Heroi morto não pode atacar. O Heroi não pode atacar um Monstro morto. Ao atacá-lo, a vida do Monstro é diminuída de acordo com o ataque da Arma. Se diminuiVida retornar true, o Heroi torna-se seu algoz e sua quantidade de vítimas é incrementada. Caso contrário, o Monstro contra-ataca com seu ataque, e a vida do Heroi deve ser diminuída deste valor. Se a vida do Heroi ficar menor ou igual a 0, o Heroi morre.
+ toString() : String → Retorna as informações do Heroi, retornando seus atributos e as informações da Arma que ele está usando.

2ª Questão



Classe Estadio:
- cidade : String → Cidade onde o Estadio se encontra. Não pode ser uma String vazia.
- pais : String → País onde o Estadio se encontra. Não pode ser uma String vazia.
- capacidade : int → Capacidade do Estadio, em número de pessoas.

+ Estadio(cidade : String, pais : String, capacidade : int) → Construtor.

+ getCap() : int → Retorna a capacidade do Estadio, em número de pessoas.
+ toString() : String → Retorna as informações do Estadio, mostrando seus atributos.

Classe JogoTenis:
- data : String → Data do Jogo de Tenis. Não pode ser uma String vazia.
- publico : int → Público presente no Jogo de Tenis. Deve ser um número positivo. Não pode ser maior que a capacidade do Estadio que sedia o jogo.
- vencedor : Jogador = null → Jogador que venceu a partida. Caso o jogo esteja sendo jogado, vencedor será nulo.

+ JogoTenis(data : String, publico : int, j1 : Jogador, j2 : Jogador, estádio : Estadio) → Construtor.

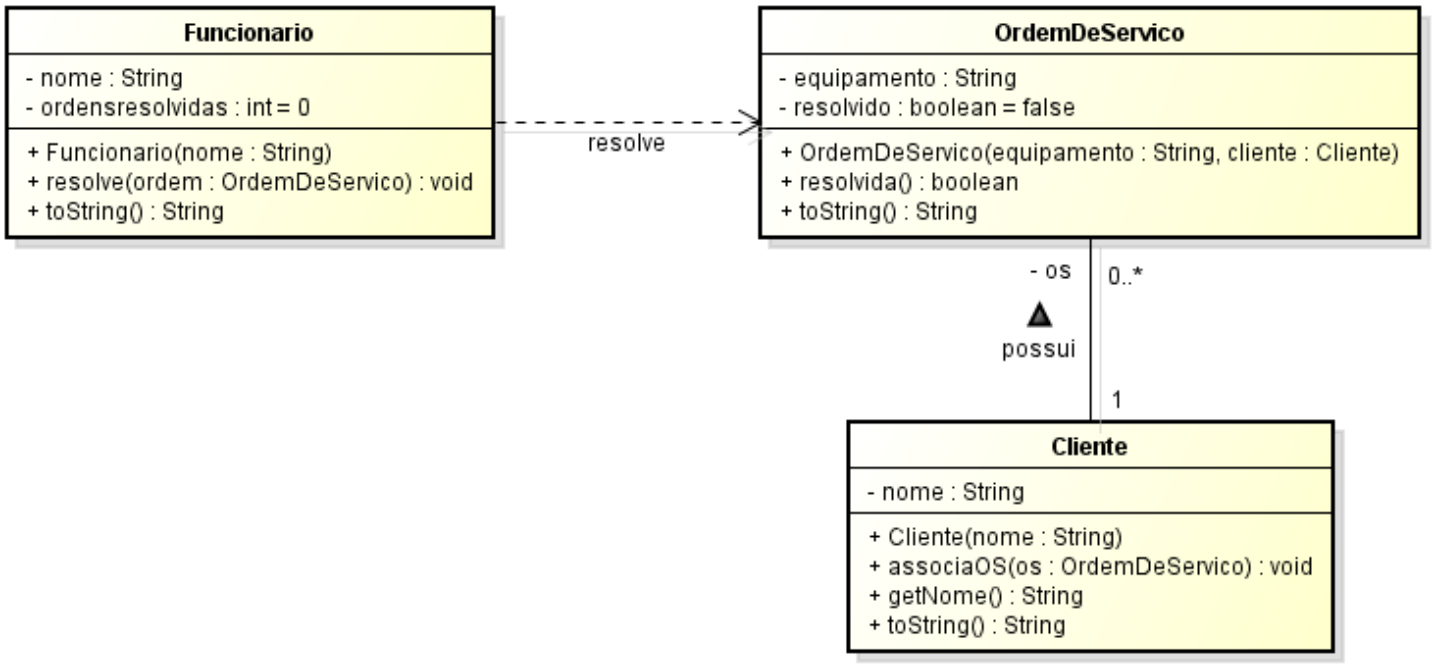
+ isVencedor(j : Jogador) : boolean → Retorna **true** se o Jogador argumento for o vencedor ou **false** caso contrário.
+ defineVencedor(n : int) : void → Define o vencedor do jogo baseado no número presente no argumento. Não pode-se definir vencedor em um jogo que este já foi definido. Se o argumento for 1, um jogador é declarado vencedor. Se for 2, o outro que é declarado vencedor. O argumento não pode ser outro número.
+ toString() : String → Retorna as informações do Jogo de Tênis, retornando seus atributos, nome dos jogadores, informações do estádio e nome do vencedor, caso haja um.

Classe Jogador:
- nome : String → Nome do Jogador. Não pode ser uma String vazia.
- patrocinador : String → Nome do patrocinador do Jogador, e.g. a marca que patrocina seu uniforme. Não pode ser uma String vazia.

+ Jogador(nome : String, patrocinador : String) → Construtor.

+ associaJogo(j : JogoTenis) : void → Associa o Jogador a um Jogo de Tenis argumento, colocando-o nos jogos jogados pelo jogador.
+ getNome() : String → Retorna o nome do Jogador.
- numVitorias() : int → Retorna o número de vitórias que este Jogador possui. Pesquise em seus jogos para saber este valor.
+ toString() : String → Retorna as informações do Jogador, retornando seus atributos, número de jogos e número de vitórias.

3ª Questão



Classe OrdemDeServico:
- equipamento : String → Nome do equipamento da OrdemDeServico. Não pode ser uma String vazia.
- reolvido : boolean = false → Determina se a OrdemDeServico foi resolvida(**true**) ou não(**false**).

+ OrdemDeServico(equipamento : String, cliente : Cliente) → Construtor.

+ resolvida() : boolean → Retorna **true**, caso a OrdemDeServico já estiver resolvida ou **false** caso esteja a resolver. Caso ela esteja a resolver, ela é resolvida.
+ toString() : String → Retorna as informações da OrdemDeServico, retornando seus atributos e o nome do Cliente que possui a ordem de serviço.

Classe Cliente:
- nome : String → Nome do Cliente. Não pode ser uma String vazia.

+ Cliente(nome : String) → Construtor.

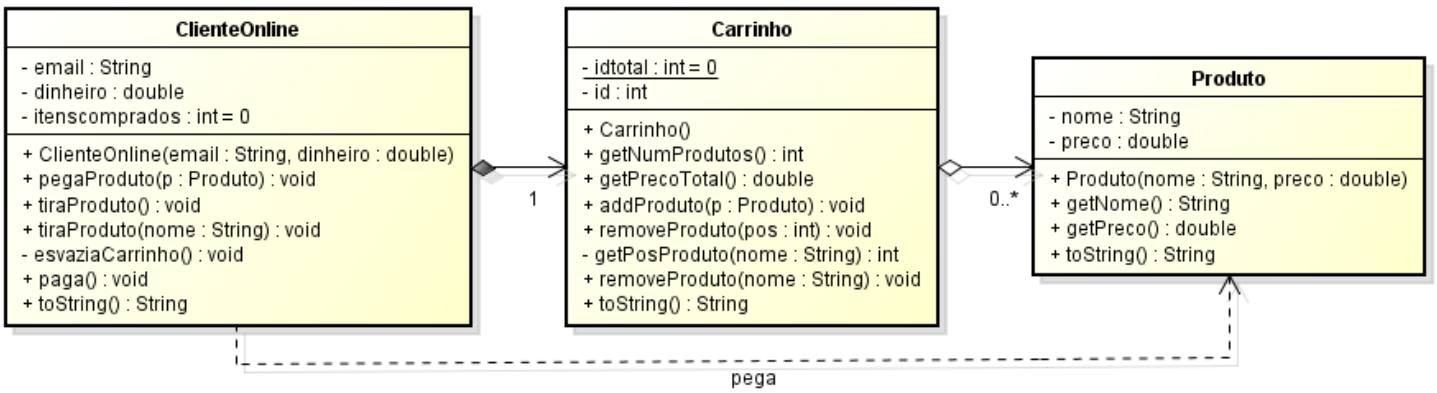
+ getNome() : String → Retorna o nome do Cliente.
+ associaOS(ordem : OrdemDeServico) : void → Associa a ordem de serviço argumento ao Cliente, colocando-a nas ordens que o Cliente abriu na loja.
+ toString() : String → Retorna as informações do Cliente, retornando seus atributos e as informações de suas ordens de serviço.

Classe Funcionario:
- nome : String → Nome do Funcionario. Não pode ser uma String vazia.
- ordensresolvidas : int = 0 → Número de Ordens de Serviço resolvidas pelo Funcionario.

+ Funcionario(nome : String) → Construtor.

+ resolve(ordem : OrdemDeServico) : void → O Funcionario resolve uma ordem de serviço. Não é possível resolver uma ordem de serviço já resolvida. Em caso de sucesso, incremente a quantidade de ordens de serviço resolvidas.
+ toString() : String → Retorna as informações do Funcionario, mostrando seus atributos.

4ª Questão



Classe Produto:
- nome : String → Nome do produto. Não pode ser uma string vazia.
- preco : double → Preço do produto, em reais. Deve ser um valor positivo.

+ Produto(nome : String, preco : double) → Construtor.

+ getNome() : double → Retorna o nome do Produto.
+ getPreco() : double → Retorna o preço do Produto.
+ toString() : String → Retorna as informações do Produto.

Classe Carrinho:
- idtotal : int = 0 → Id contador de Carrinhos. Incremente-o a cada Carrinho criado.
- id : int → Id do Carrinho. Deve ser o valor corrente de idtotal.

+ Carrinho() → Construtor.

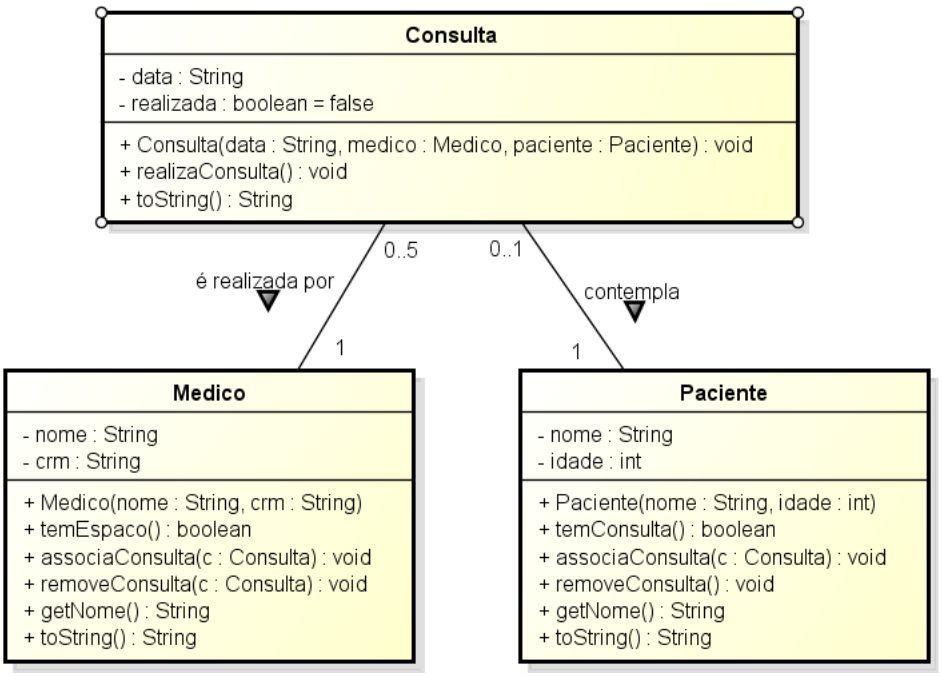
+ getNumProdutos() : int → Retorna o número de produtos dentro do Carrinho.
+ getPrecoTotal() : double → Retorna a soma dos preços dos produtos dentro do Carrinho.
+ addProduto (p : Produto) : void → Adiciona um Produto, argumento do método, ao carrinho.
+ removeProduto(pos : int) : void → Remove o Produto no índice do argumento. Um produto deve existir naquela posição para ser removido.
- getPosProduto (nome : String) : int → Retorna o índice do primeiro produto encontrado com o nome igual ao argumento do método. Caso o nome não tenha sido encontrado, retorne o valor -1.
+ removeProduto (nome : String) : void → Remove o primeiro Produto encontrado que tenha nome igual ao argumento. Obrigatório o uso dos métodos removeProduto e getPosProduto.
+ toString() : String → Retorna as informações do Carrinho. Seus atributos e as informações dos Produtos que o agregam, além da quantidade de produtos.

Classe ClienteOnline:
- email : String → E-mail do cliente.
- dinheiro : double → Importância carregada pelo cliente.
- itenscomprados : int = 0 → Quantidade de itens comprados e pagos pelo Cliente.

+ ClienteOnline(email : String, dinheiro : double) → Construtor.

+ pegaProduto(p : Produto) : void → Adiciona um Produto, argumento do método, em seu Carrinho.
+ tiraProduto() : void → Remove o último Produto de seu Carrinho, caso haja algum.
+ tiraProduto (nome : String) : void → Remove um Produto, cujo nome é o argumento do método, de seu Carrinho. Um Produto só pode ser removido caso exista.
- esvaziaCarrinho () : void → Esvazia o Carrinho, atribuindo null a ele. Construa um novo objeto da classe Carrinho, substituindo o antigo. Emita uma mensagem com o email do cliente que teve o carrinho esvaziado.
+ paga() : void → Realiza uma compra, dos produtos que estão no Carrinho. Mostre mensagens, independente da situação, mostrando o valor da compra e o dinheiro do cliente, bem como se a compra foi realizada com sucesso ou não. Uma compra só deve ser realizada caso o Cliente tenha dinheiro suficiente. Atualize dinheiro a cada compra realizada com sucesso, subtraindo o valor comprado, e a quantidade de itens comprados. Esvazie o carrinho após a compra.
+ toString() : String → Retorna as informações do ClienteOnline. Seus atributos, quantidade de produtos em seu Carrinho e o preço total de produtos em seu Carrinho.

5ª Questão



Classe Medico:
- nome : String → Nome do Medico. Não pode ser uma String vazia.
- crm : String → Registro do Medico. Não pode ser uma String vazia.

+ Medico(nome : String, crm : String) → Construtor.

+ temEspaco() : boolean → Retorna **true** se o Medico tem espaço em sua agenda para consultas ou **false** caso contrário.
+ associaConsulta(c : Consulta) : void → Associa a Consulta, colocando-a na agenda das consultas do Medico. Isto pode acontecer somente se o Medico possuir espaço para mais consultas.
+ removeConsulta(c : Consulta) : void → Remove a Consulta da agenda das consultas do Medico. Isto acontecerá quando ela for realizada e existir na agenda do Médico.
+ getNome() : String → Retorna o nome do Medico.
+ toString() : String → Retorna as informações do Medico, além da quantidade de consultas que ele irá realizar e as informações delas.

Classe Paciente:
- nome : String → Nome do Paciente. Não pode ser uma String vazia.
- idade : int → Idade do Paciente. Deve ser um número natural.

+ Paciente(nome : String, idade : int) → Construtor.

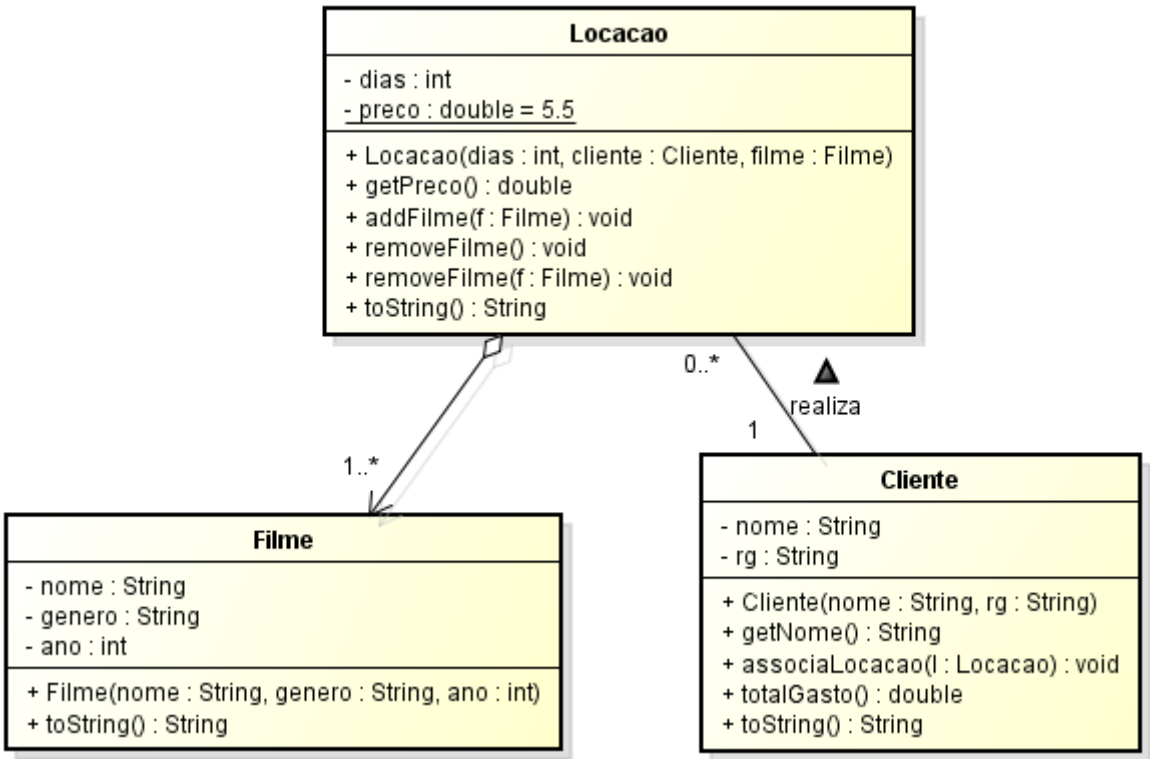
+ temConsulta() : boolean → Retorna **true** se o Paciente tem consulta marcada ou **false** caso contrário.
+ associaConsulta(c : Consulta) : void → Associa a Consulta, colocando-a na agenda do Paciente. Isto só pode acontecer se não houver nenhuma consulta marcada para este paciente.
+ removeConsulta(c : Consulta) : void → Remove a Consulta da agenda das consultas do Paciente. Isto acontecerá quando ela for realizada e o Paciente tiver esta Consulta marcada.
+ getNome() : String → Retorna o nome do Paciente.
+ toString() : String → Retorna as informações do Paciente, além das informações de sua Consulta, se houver.

Classe Consulta:
- data : String → Data da Consulta. Não pode ser uma String vazia.
- realizada : boolean = false → Determina se a Consulta foi realizada(**true**) ou não(**false**).

+ Consulta(data : String, medico : Medico, paciente : Paciente) → Construtor.

+ realizaConsulta() : void → Realiza a Consulta. Isto pode acontecer se ela não já estiver sido realizada. Modifica os status de realizada, e remove a Consulta da agenda do Médico e do Paciente. Deixe o registro deles na Consulta, pois servirá de log.
+ toString() : String → Retorna as informações da Consulta, além dos nomes do Paciente e do Médico.

6ª Questão



Classe Filme:
- nome : String → Nome do Filme. Não pode ser uma String vazia.
- genero : String → Gênero do Filme, e.g. terror, comédia, aventura. Não pode ser uma String vazia.
- ano : int → Ano de lançamento do Filme. Deve ser posterior à invenção dos filmes (pesquise).

+ Filme(nome : String, genero : String, ano : int) → Construtor.

+ toString() : String → Retorna as informações do Filme, ou seja, seus atributos.

Classe Cliente:
- nome : String → Nome do Humano. Não pode ser uma String vazia.
- rg : String → Registro geral do Cliente. Não pode ser uma String vazia.

+ Cliente(nome : String, rg : String) → Construtor.

+ getNome() : String → Retorna o nome do Cliente.
+ associaLocacao(l : Locacao) : void → Associa a Locacao ao Cliente, colocando-a em sua coleção de Locações.
+ totalGasto() : Double → Retorna o total de dinheiro gasto pelo Cliente em Locações.
+ toString() : String → Retorna as informações do Cliente, mostrando seus atributos e a quantidade e informações de suas locações, caso haja alguma.

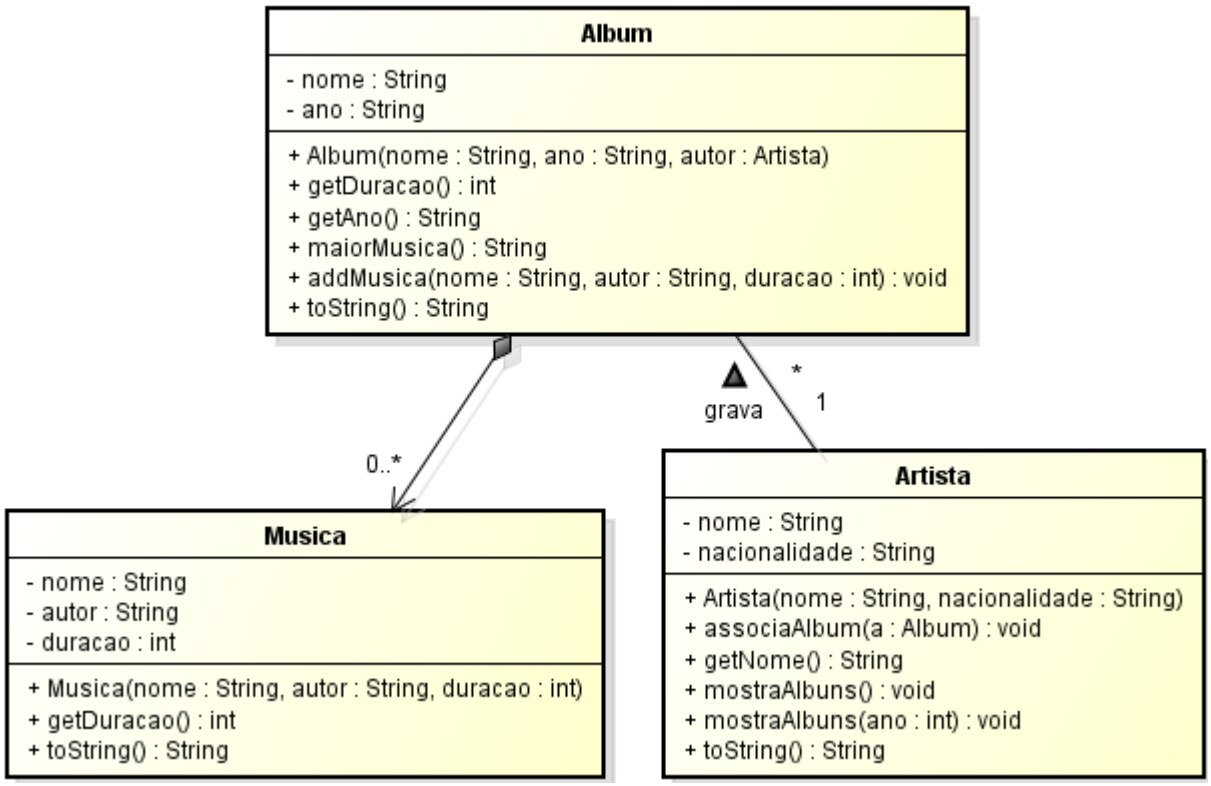
Classe Locacao:
- dias : int → Quantidade de dias que a Locacao ficará com o Cliente. Deve ser um número positivo.
- preco : double = 5.5 → Preço de cada filme locado por dia.

+ Locacao(dias : int, cliente : Cliente, filme : Filme) → Construtor.

+ getPreco() : double → Calcula e retorna o preço dessa Locação, baseando na quantidade de filmes, dias e no preço do filme locado por dia.
+ addFilme(f : Filme) : void → Adiciona um Filme na Locação. Isto pode acontecer se o filme já não existir na Locação.
+ removeFilme() : void → Remove o último Filme da Locação. Respeite a multiplicidade.

+ removeFilme(f : Filme) : void → Remove um Filme da Locação. Isto pode acontecer se o filme já existir na Locação. Respeite a multiplicidade.
+ toString() : String → Retorna as informações da Locacao: Seus atributos, as informações dos Filmes que formam-na, além do nome do Cliente que realiza a Locação.

7ª Questão



Classe Musica:
- nome : String → Nome da Musica. Não pode ser uma String vazia.
- autor : String → Nome do autor da Musica. Não pode ser uma String vazia.
- duracao : int → Duração da Musica, em segundos. Não pode ser uma String vazia.

+ Musica(nome : String, autor : String, duracao : int) → Construtor.

+ getDuracao() : int → Retorna a duração da Musica.
+ toString() : String → Retorna as informações da Musica, contendo seus atributos.

Classe Artista:
- nome : String → Nome do Artista. Não pode ser uma String vazia.
- nacionalidade : String → País de nascimento do Artista. Não pode ser uma String vazia.

+ Artista(nome : String, nacionalidade : String) → Construtor.

+ associaAlbum(a : Album) : void → Coloca o Album argumento nos Albuns gravados pelo Artista.
+ getNome() : String → Retorna o nome do Artista.
+ mostraAlbuns() : void → Mostra na tela os atributos do Artista e a informação de todos os Albuns que ele já gravou.
+ mostraAlbuns(ano : int) : void → Mostra na tela os atributos do Artista e a informação de todos os Albuns que ele gravou no ano presente no argumento.
+ toString() : String → Retorna as informações do Artista, contendo seus atributos, a quantidade de Albuns gravados e a duração de todas as suas gravações, dispostas no formato (minutos:segundos).

Classe Album:
- nome : String → Nome do Album. Não pode ser uma String vazia.
- ano : String → Ano de lançamento do Album. Deve ser posterior ao invento da gravação de músicas(pesquise).

+ Album(nome : String, ano : String) → Construtor.

+ getDuracao() : int → Retorna a duração do Album.
+ getAno() : String → Retorna o ano de lançamento do Album.
+ maiorMusica() : String → Retorna as informações da Musica com maior duração presente no Album. Caso haja um empate, retorne as informações das músicas empatadas.
+ addMusica(nome : String, autor : String, duração : int) : void → Adiciona a Musica ao Album.

+ toString() : String → Retorna as informações do Album, contendo seus atributos, as informações das Musicas contidas nele (caso haja alguma) e o nome do artista.
