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 2-7

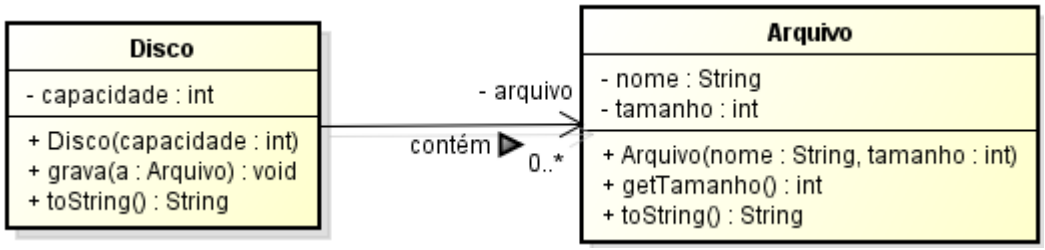
Obs: Para todos os exercícios, proceda conforme a aula. Construa objetos, contemple a multiplicidade e execute os métodos.

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 Disco:
- capacidade : int → Capacidade do Disco, em kilobytes(kB). Deve ser um número positivo.

+ Disco(capacidade : int) → Construtor.

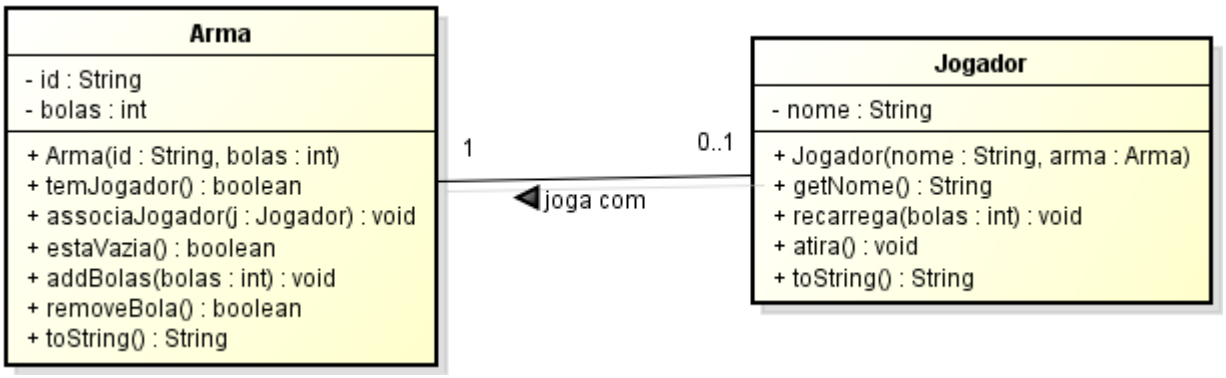
+ grava(a : Arquivo) : void → Grava o Arquivo argumento no Disco. Isto só pode acontecer se o Disco comportar o Arquivo.
+ toString() : String → Retorna as informações do Disco, mostrando seus atributos e dos Arquivos que ele contém.

Classe Arquivo:
- nome : String → Nome do Arquivo. Não pode ser uma String vazia.
- tamanho : int → Tamanho do Arquivo, em kilobytes(kB). Deve ser um número positivo.

+ Arquivo(nome : String, capacidade : int) → Construtor.

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

2ª Questão



Classe Arma: Arma de Paintball
- id : String → Identificação da Arma. Não pode ser uma String vazia.
- bolas : int → Quantidade de bolas de paintball presentes na arma.

+ Arma(id : String, bolas : int) → Construtor.

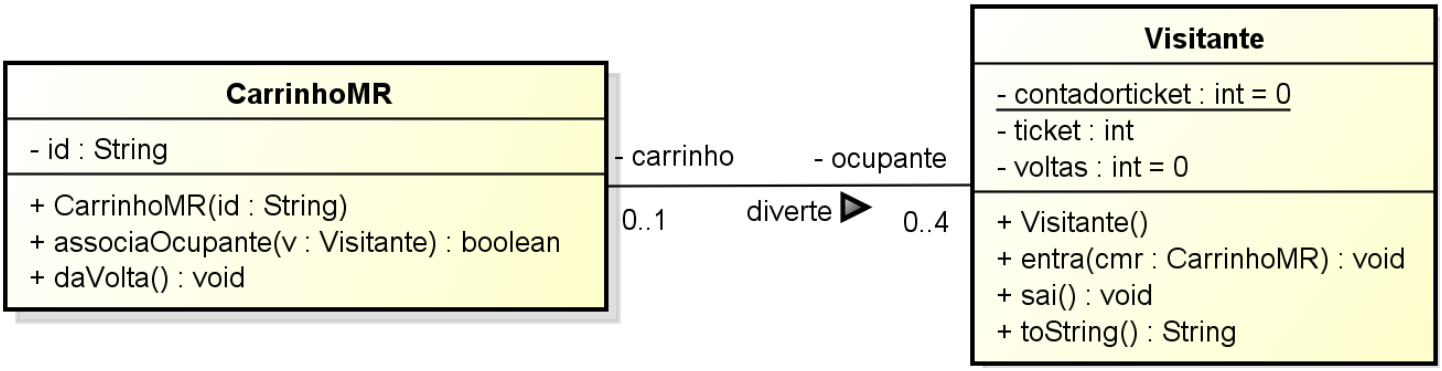
+ temJogador() : boolean → Retorna **true** se a Arma já possuir Jogador que joga com dela e **false** caso contrário.
+ associaJogador(j : Jogador) : void → O Jogador presente no argumento se tornará o jogador da Arma.
+ estaVazia() : boolean → Retorna **true** se a arma estiver vazia e **false** caso contrário.
+ addBolas(bolas : int) : void → Adiciona a quantidade de bolas presente no argumento à Arma. O argumento deve ser um número positivo.
+ removeBola() : boolean → Retorna **false** se a Arma estiver sem bolas, ou remova uma bola e retorne **true**.
+ toString() : String → Retorna as informações da Arma, mostrando seus atributos e o nome do Jogador que joga com esta arma.

Classe Jogador: Jogador de Paintball – Antes de criar um Jogador, verifique se a Arma com que ele jogará já tem jogador.
- nome : String → Nome do Jogador. Não pode ser uma String vazia.

+ Jogador(nome : String, arma : Arma) → Construtor.

+ getNome() : String → Retorna o nome do Jogador.
+ recarrega(bolas : int) : void → Recarrega a Arma do Jogador com a quantidade de bolas presente no argumento. A Arma só pode ter bolas adicionadas caso não haja nenhuma bola. Mostre as informações da Arma do Jogador quando recarregá-la.
+ atira() : void → Atira com a Arma do Jogador, removendo uma bola dela. A Arma só pode ser disparada caso haja balas nela. Mostre um aviso em caso de erro.
+ toString() : String → Retorna as informações do Jogador, mostrando seus atributos e as informações de sua Arma.

3ª Questão



Classe CarrinhoMR: Carrinho de Montanha-Russa
- id : String → Identificação do Carrinho. Não pode ser uma String vazia.

+ CarrinhoMR(id : String) → Construtor.

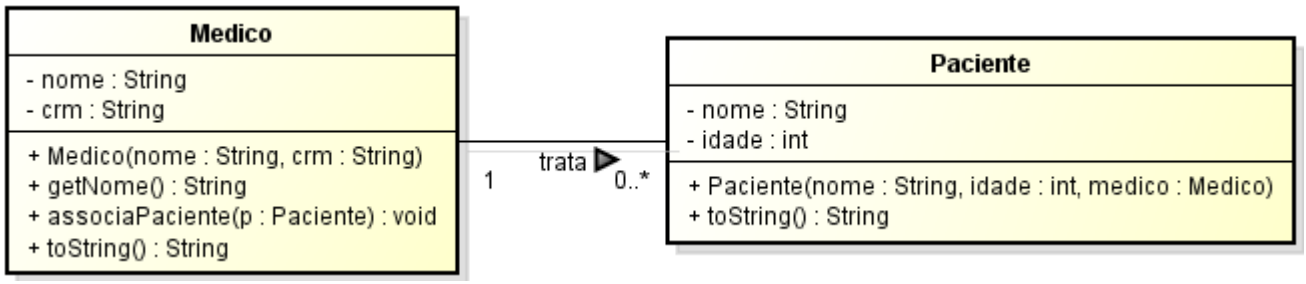
+ associaOcupante(v : Visitante) : boolean → Se o CarrinhoMR não estiver cheio o Visitante deve entrar no CarrinhoMR. Assim, coloque-o na lista de ocupantes e retorne true. Caso contrário retorne false.
+ daVolta() : void → Realiza uma volta com os ocupantes. Uma volta só pode ser realizada com o Carrinho cheio. Após a volta, os Visitantes devem sair (retire o visitante), e o Carrinho deve ser esvaziado.

Classe Visitante:
- contadorticket : int = 0 → Contador de tickets do parque. **Incremente toda vez que criar um visitante.**
- ticket : int → Ticket do visitante. Deve ser igual ao contador de tickets.
- voltas: int = 0 → Número de voltas que o Visitante realizou na Montanha-Russa.

+ Visitante() → Construtor.

+ entra(cmr : CarrinhoMR) : void → O Visitante entra no CarrinhoMR presente no argumento. Isto só pode ser feito se o Visitante já não estiver em um CarrinhoMR, e se na hora da associação o CarrinhoMR não estiver cheio. Em caso de erro mostre uma mensagem. Em caso de sucesso, associe o Carrinho ao Visitante, que será seu ocupante.
+ sai() : void → O Visitante sai do seu Carrinho. Isto só pode ser feito se o Visitante já estiver em um CarrinhoMR. Em caso de erro mostre uma mensagem. Em caso de sucesso, quer dizer que o Visitante realizou uma volta. Assim sendo, incremente o atributo voltas.
+ toString() : String → Retorna as informações da Pessoa, mostrando seus atributos.

4ª 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.

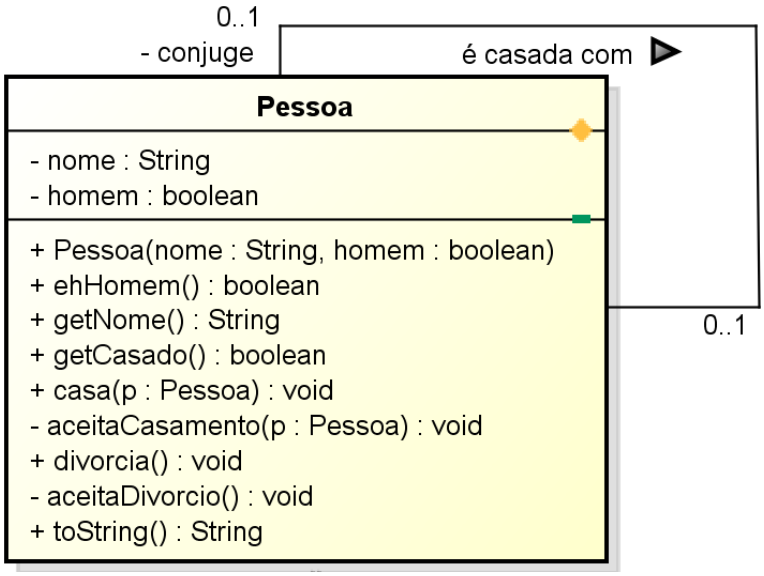
+ getNome() : String → Retorna o nome do Medico.
+ associaPaciente(p : Paciente) : void → Associa o Paciente, colocando-o na lista de Pacientes que consultam com o Medico.
+ toString() : String → Retorna as informações do Medico, além da informação de seus Pacientes, se houver.

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.

+ toString() : String → Retorna as informações do Paciente, além do nome de seu Medico.

5ª Questão

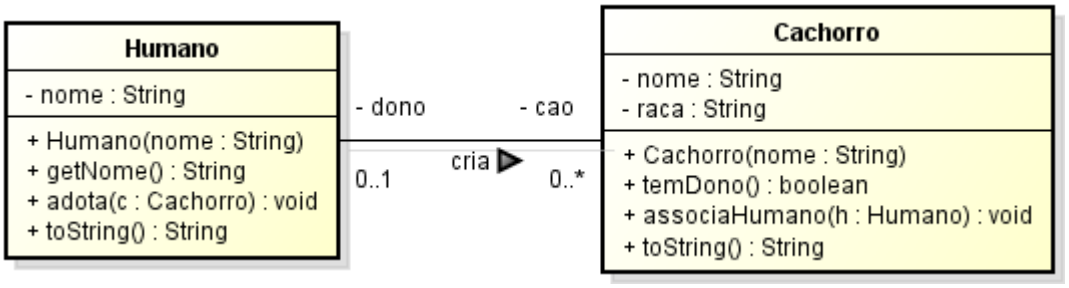


Classe Pessoa:
- nome : String → Nome da Pessoa. Não pode ser uma String vazia.
- homem : boolean → Indica se a Pessoa é homem ou mulher. Se for **true**, a Pessoa é homem. Caso seja **false**, a Pessoa é mulher.

+ Pessoa(nome : String, homem : boolean) → Construtor.

+ ehHomem() : boolean → Retorna **true** se a Pessoa for homem e **false** se for mulher.
+ getNome() : String → Retorna o nome da Pessoa.
+ getCasado() : boolean → Retorna **true** se a Pessoa for casada com outra Pessoa e **false** se ela não possuir nenhum cônjuge, assim sendo solteira.
+ casa(p : Pessoa) : void → Casa com outra Pessoa. Isto só pode acontecer se a outra Pessoa for do sexo oposto e solteira (de acordo com a constituição federal). Mostre um aviso em caso de erro. Caso não haja nenhum impedimento a Pessoa do argumento deve aceitar o casamento, e a Pessoa que invocou o método deve alterar seu cônjuge.
- aceitaCasamento(p : Pessoa) : void → A Pessoa foi pedida em casamento, e assim deve alterar seu cônjuge de nulo para o argumento do método.
+ divorcia() : boolean → Divorcia de seu cônjuge. Isto só pode acontecer a Pessoa já for casada. Mostre um aviso em caso de erro. Caso a Pessoa seja casada, seu cônjuge deve aceitar o divórcio, e a referência de cônjuge deve ser eliminada, atribuindo-se **null** a ela.
- aceitaDivorcio() : void → À Pessoa foi solicitado o divórcio, e assim a referência de cônjuge deve ser eliminada, atribuindo-se **null** a ela.
+ toString() : String → Retorna as informações da Pessoa, mostrando seus atributos e se a Pessoa é casada ou solteira. Caso seja casada, mostre o nome de seu cônjuge.

6ª Questão



Classe Cachorro:
- nome : String → Nome do Cachorro. Não pode ser uma String vazia.
- raca : String → Raça do Cachorro. Não pode ser uma String vazia.

+ Cachorro(nome : String, raca : String) → Construtor.

+ temDono() : boolean → Retorna **true** se o for Cachorro possuir um dono e **false** caso contrário.
+ associaHumano(h : Humano) : void → O Humano presente no argumento se tornará o dono do Cachorro e irá criá-lo.
+ toString() : String → Retorna as informações do Cachorro, mostrando seus atributos e o nome de seu Dono, caso possua um.

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

+ Humano(nome : String) → Construtor.

+ getNome() : String → Retorna o nome do Humano.
+ adota(c : Cachorro) : void → Adota o Cachorro presente no argumento. Porém, isto pode acontecer somente se ele não possuir um dono. Mostre um aviso em caso de erro. Em caso de sucesso, associe o Cachorro ao Humano utilizando o método adequado.
+ toString() : String → Retorna as informações do Humano, mostrando seus atributos e as informações de seus cães, caso haja algum.
