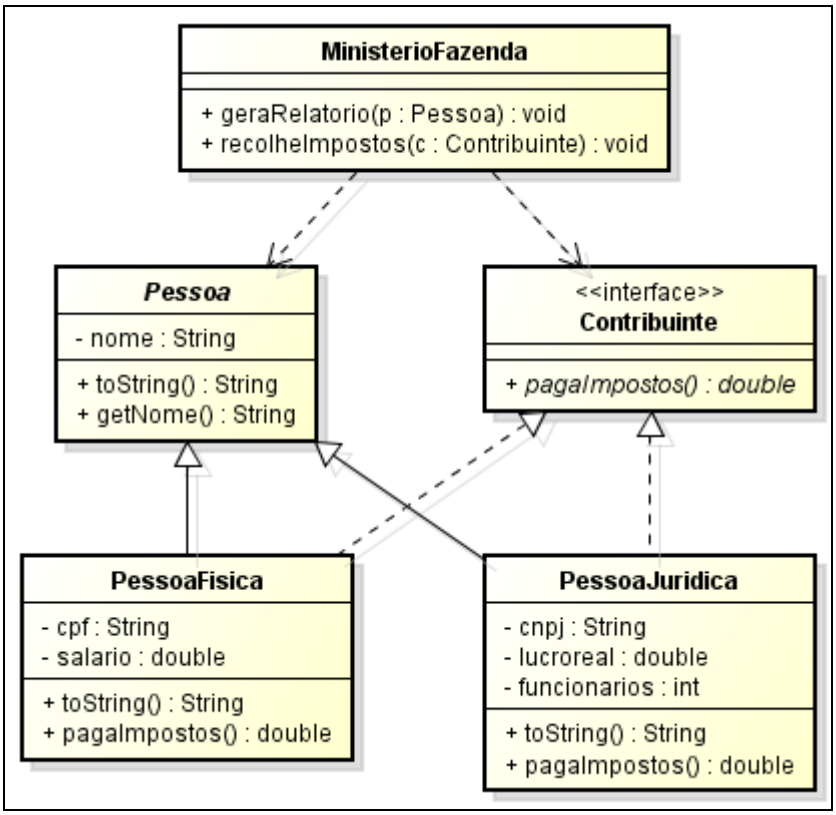
MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO TRIÂNGULO MINEIRO
CAMPUS UBERLÂNDIA CENTRO

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

LISTA 3-5

Obs: Caso haja alguma divergência em relação a visibilidade, abstração e/ou parâmetros, consulte o diagrama do exercício correspondente. Bom estudo.

1ª Questão



Classe Pessoa:
- nome : String → Nome da Pessoa.

+ toString() : void → Retorna as informações de uma Pessoa, ou seja, seu nome.
+ getNome() : String → Retorna o nome.

Classe PessoaFisica:
- cpf : String → CPF da PessoaFisica.
- salario : double → Salário anual da PessoaFisica.

+ toString() : void → Retorna as informações de uma PessoaFisica, ou seja, suas informações como Pessoa, além de CPF e salário.
+ pagaImpostos() : double → Retorna o valor devido de impostos pela PessoaFisica. Pesquise como realizar este pagamento no ano atual.

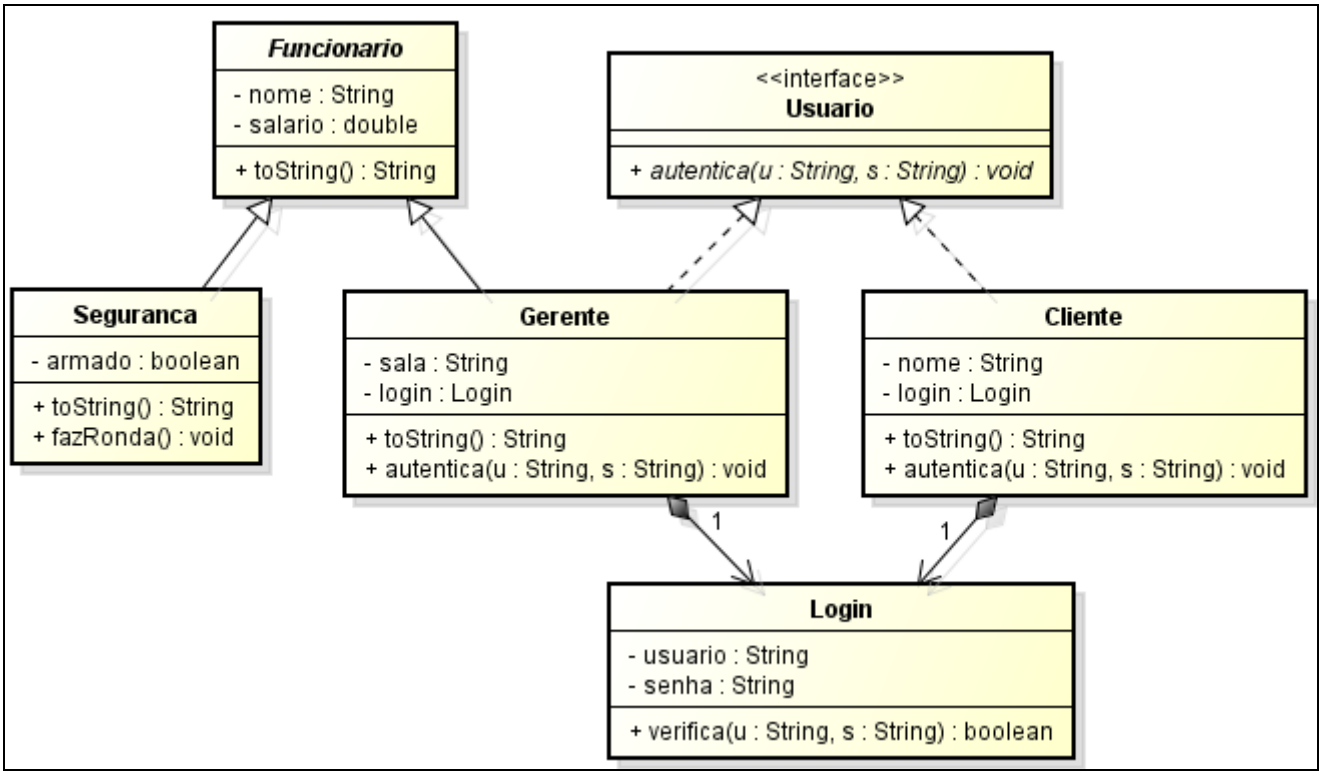
Classe PessoaJuridica:
- cnpj : String → CNPJ da PessoaJuridica.
- lucroreal : double → Lucro real anual da PessoaJuridica.
- funcionários : int → Número de funcionários da PessoaJuridica.

+ toString() : void → Retorna as informações de uma PessoaJuridica, ou seja, suas informações como Pessoa, além de CNPJ, lucro real e quantidade de funcionários.
+ pagaImpostos() : double → Retorna o valor devido de impostos pela PessoaJuridica. Pesquise como realizar este pagamento no ano corrente.

Classe MinisterioFazenda:
+ geraRelatorio(p : Pessoa) : void → Gera relatório de uma Pessoa. Esta Pessoa deve ter suas informações mostradas na tela. Mostre qual tipo de Pessoa está tendo seu relatório gerado.
+ recolheImpostos(c : Contribuinte) : void → Recolhe os impostos de alguém que se comporta como Contribuinte. Este Contribuinte deve pagar seus impostos. Mostre qual tipo de Contribuinte está pagando os impostos, juntamente com os impostos devidos.

Exercício:
Crie objetos das classes concretas. Faça um MinisterioFazenda. Gere relatórios e recolha impostos das Pessoas e Contribuintes, respectivamente.

2ª Questão



Classe Funcionario:
- nome : String → Nome do Funcionario.
- salario : String → Salário do Funcionario.

+ toString() : String → Retorne as informações do Funcionario: Seu nome e salário.

Classe Seguranca:
- armado : boolean → Especifica se o Seguranca possui ou não arma de fogo.

+ toString() : String → Retorne as informações do Segurança,
+ fazRonda() : void → Mostra na tela que o Segurança fez a ronda, juntamente com suas informações.

Classe Login:
- usuario : String → Usuário do Login.
- senha : String → Senha do Login.

+ verifica(u : String, s : String) : void → Caso as informações dos argumentos coincidam com o usuário e senha, respectivamente, retorne **true**. Caso contrário, retorne **false**.

Classe Gerente:
- sala : String → Sala em que o Gerente trabalha.
- login : Login → Login do Gerente, construído a partir de um usuário e senha.

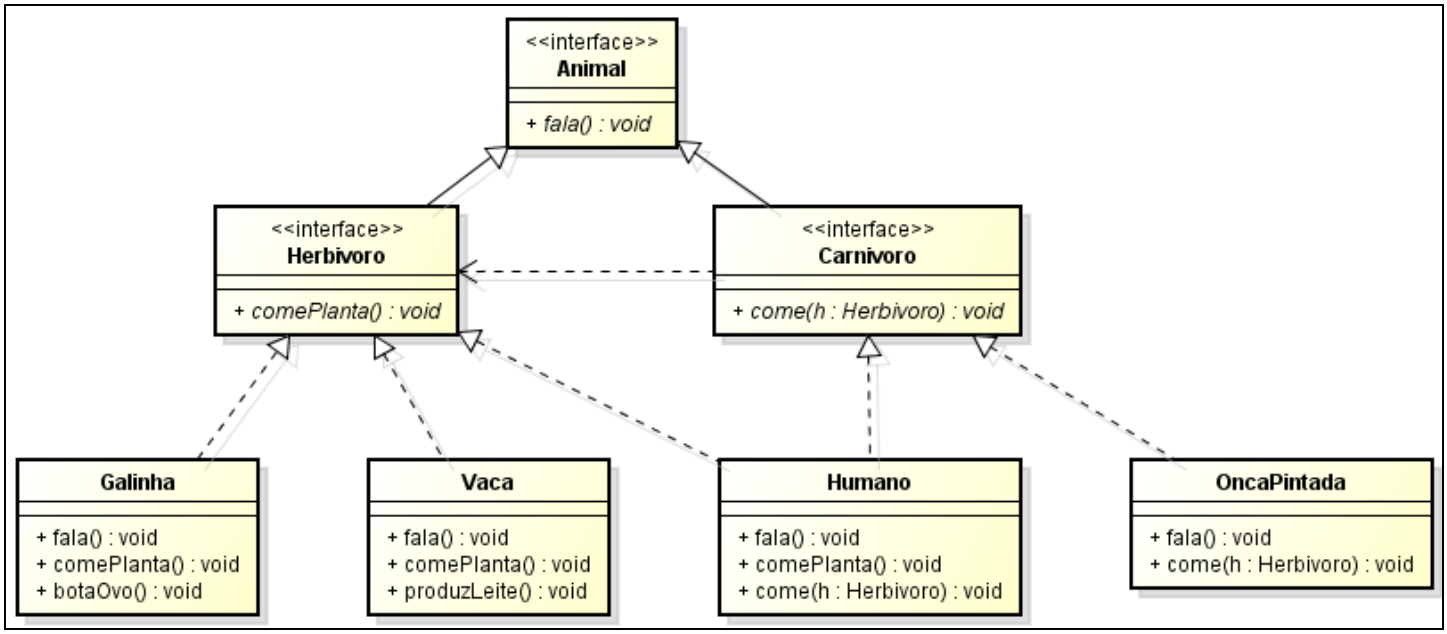
+ toString() : String → Retorna as informações do Gerente: As informações dele como Funcionário, além de sua sala.
+ autentica(u : String, s : String) : void → Verifica se os argumentos coincidem com o seu Login. Caso positivo, mostra na tela que a autenticação do Gerente foi realizada com sucesso. Caso contrário, mostre uma mensagem de falha.

Classe Cliente:
- nome : String → Nome do Cliente.
- login : Login → Login do Cliente, construído a partir de um usuário e senha.

+ toString() : String → Retorna as informações do Gerente: As informações dele como Funcionário, além de sua sala.
+ autentica(u : String, s : String) : void → Verifica se os argumentos coincidem com o seu Login. Caso positivo, mostra na tela que a autenticação do Cliente foi realizada com sucesso. Caso contrário, mostre uma mensagem de falha.

Exercício:
Crie objetos das classes concretas. Execute os métodos. Tente autenticar Gerentes e Clientes, com sucesso e falha.

3ª Questão



Interface Animal:
+ fala() : void → Toda classe que se comporta como Animal, fala.

Interface Herbivoro:
+ comePlanta() : void → Toda classe que se comporta como Herbivoro, come plantas, ou seja, alimentos de origem vegetal.

Interface Carnivoro:
+ come(h : Herbivoro) : void → Toda classe que se comporta como Carnívoro, come alguém que se comporta como Herbívoro.

Classe Galinha:
+ fala() : void → Uma Galinha fala "Cocorico". Mostre isso na tela.
+ comePlanta() : void → Uma Galinha come milho. Mostre isso na tela.

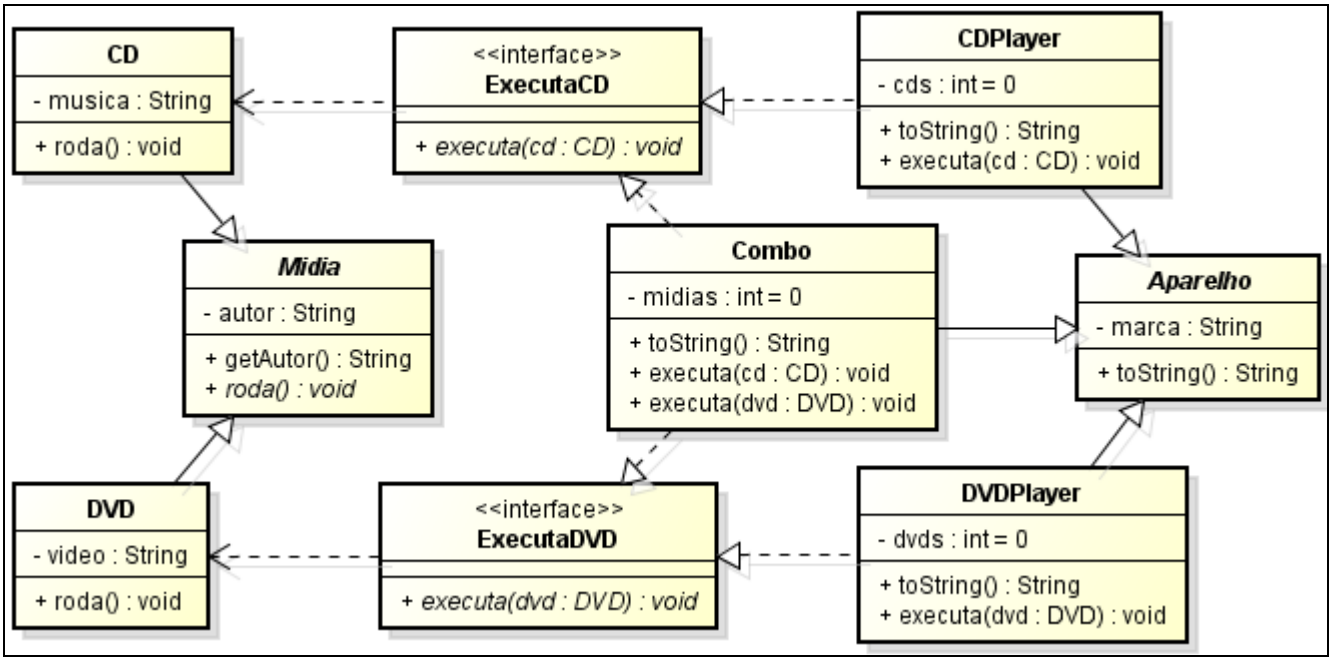
Classe Vaca:
+ fala() : void → Uma Vaca fala "Moo". Mostre isso na tela.
+ comePlanta() : void → Uma Vaca come pasto. Mostre isso na tela.

Classe Humano:
+ fala() : void → Um Humano fala "Olá Jovem". Mostre isso na tela.
+ comePlanta() : void → Um Homem come arroz. Mostre isso na tela.
+ come(h : Herbivoro) : void → Um Humano come um Herbívoro. Porém, não é permitido canibalismo. Por isto um Humano não pode comer outro Homem. Mostre qual tipo de Herbívoro foi consumido. Caso um Humano seja o argumento, mostre uma mensagem de erro.

Classe OncaPintada:
+ fala() : void → Uma OncaPintada fala "Roar!". Mostre isso na tela.
+ come(h : Herbivoro) : void → Uma OncaPintada come um Herbívoro. Mostre qual tipo de Herbívoro foi consumido.

Exercício:
Crie objetos das classes concretas e invoque os métodos delas. Tente fazer com que os que se comportam como Carnívoros comam todos os Herbívoros.

4ª Questão



Classe Midia:
- autor : String → Autor da Mídia.

+ getAutor() : String → Retorna o autor da Mídia.
+ *roda() : void* → Toda Mídia deve ser rodada.

Classe CD:
- musica : String → Música contida no CD.

+ roda() : void → Roda o CD, mostrando na tela que a ação foi feita, juntamente com seu autor e música.

Classe DVD:
- video : String → Vídeo contido no DVD.

+ roda() : void → Roda o DVD, mostrando na tela que a ação foi feita, juntamente com seu autor e vídeo.

Classe Aparelho:
- marca : String → Autor da Mídia.

+ toString() : String → Retorna as informações do Aparelho: Sua marca.

Classe CDPlayer:
- cds : int = 0 → CDs rodados pelo CDPlayer.

+ toString() : String → Retorna as informações do CDPlayer: Suas informações como Aparelho, além de seus CDs rodados.
+ executa(cd : CD) : void → Roda o CD presente no argumento deste método. Incremente os CDs rodados.

Classe DVDPlayer:
- dvds : int = 0 → DVDs rodados pelo DVDPlayer.

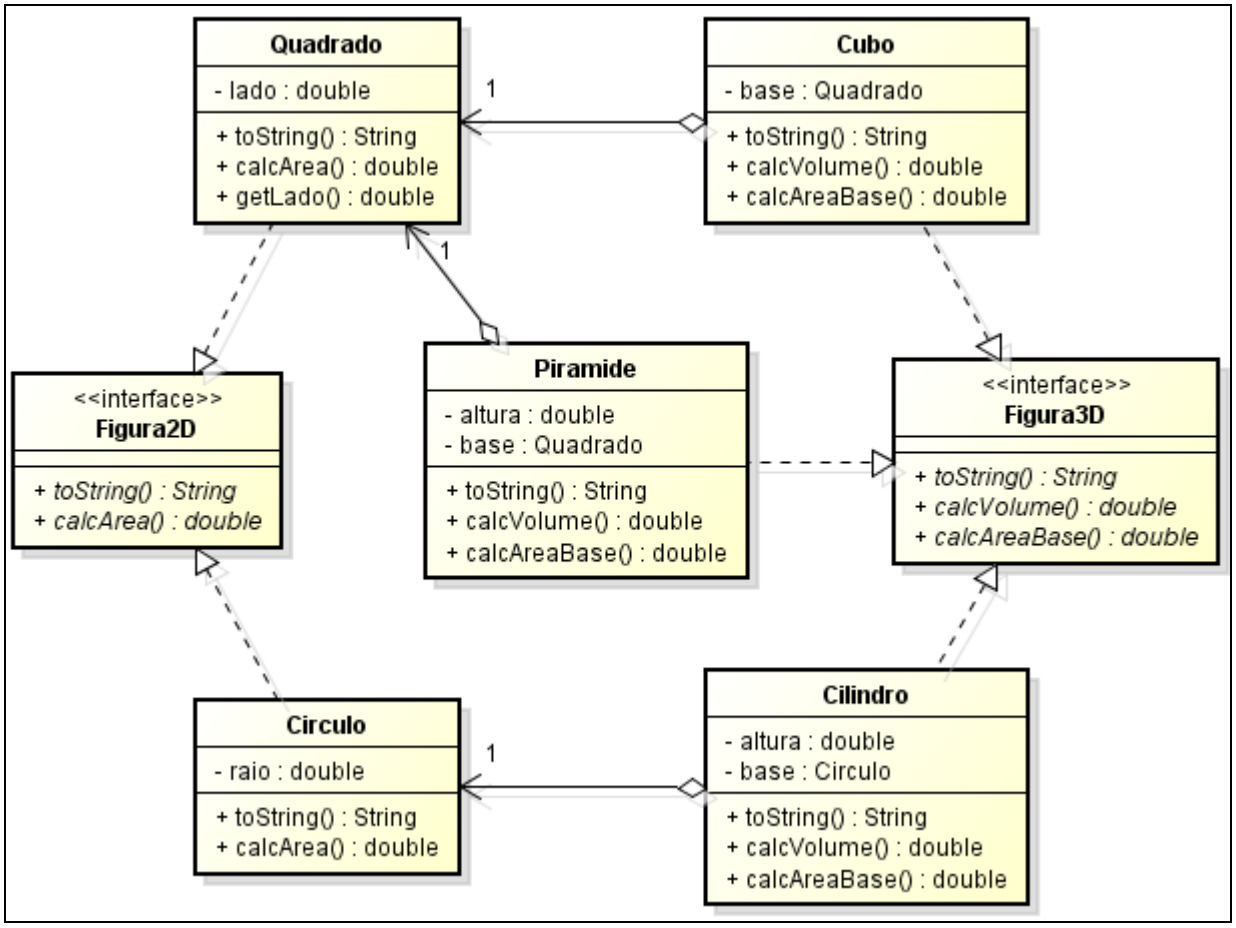
+ toString() : String → Retorna as informações do DVDPlayer: Suas informações como Aparelho, além de seus DVDs rodados.
+ executa(dvd : DVD) : void → Roda o DVD presente no argumento deste método. Incremente os DVDs rodados.

Classe Combo:
- midias : int = 0 → Mídias rodadas pelo Combo.

+ toString() : String → Retorna as informações do Combo: Suas informações como Aparelho, além de suas Mídias rodadas.
+ executa(cd : CD) : void → Roda o CD presente no argumento deste método. Incremente as Mídias rodadas.
+ executa(dvd : DVD) : void → Roda o DVD presente no argumento deste método. Incremente as Mídias rodadas.

Exercício:
Crie objetos das classes concretas e invoque os métodos delas. Rode todas as Mídias que podem ser executadas pelos Aparelhos. Mostre informações sobre objetos das classes concretas.

5ª Questão



Faça uma pesquisa sobre as fórmulas que resultam nas áreas e volumes das figuras apresentadas no diagrama.

Classe Quadrado:
- lado : double → Lado do quadrado.

+ toString() : String → Retorna as informações do Quadrado: Seu lado e sua área.
+ calcArea() : double → Retorna o valor da área do Quadrado.
+ getLado() : double → Retorna o lado do Quadrado.

Classe Cubo:
- base : Quadrado → Base do Cubo, que fornecerá o lado deste Cubo.

+ toString() : String → Retorna as informações do Cubo: Seu lado, área da base e volume.
+ calcVolume() : double → Retorna o valor do volume do Cubo.
+ calcArea() : double → Retorna o valor da área da base do Cubo.

Classe Pirâmide:
- altura : double → Altura da Pirâmide.
- base : Quadrado → Base da Pirâmide.

+ toString() : String → Retorna as informações da Pirâmide: Sua altura, além de área da base e volume.
+ calcVolume() : double → Retorna o valor do volume da Pirâmide.
+ calcArea() : double → Retorna o valor da área da base da Pirâmide.

Classe Circulo:
- raio : double → Raio do Circulo.

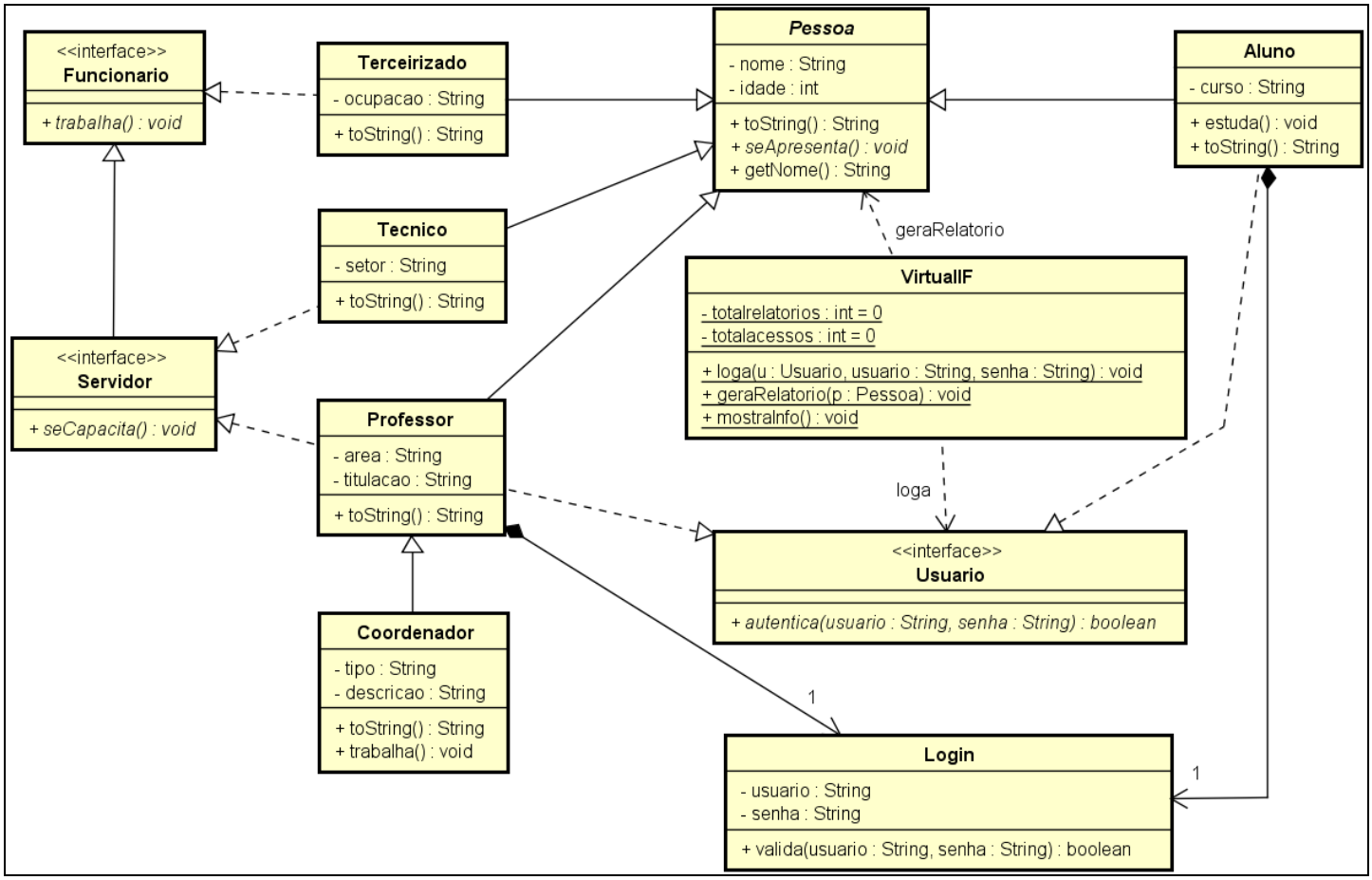
+ toString() : String → Retorna as informações do Circulo: Seu raio e sua área.
+ calcArea() : double → Retorna o valor da área do Circulo.

Classe Cilindro:
- altura : double → Altura do Cilindro.
- base : Circulo → Base do Cilindro.

+ toString() : String → Retorna as informações do Cilindro: Sua altura, além de área da base e volume.
+ calcVolume() : double → Retorna o valor do volume do Cilindro.
+ calcArea() : double → Retorna o valor da área da base do Cilindro.

Exercício:
Crie objetos das classes concretas e invoque os métodos delas. Mostre informações sobre objetos das classes concretas.

6ª Questão



Os construtores em todas as classes devem ser construídos inicializando todos os atributos não inicializados.

Interface Funcionario:
+ trabalha() : void

Interface Servidor:
+ seCapacita() : void

Classe *Pessoa*:
- nome : String → Nome da Pessoa.
- idade : int → Idade da Pessoa.

+ toString() : String → Retorna as informações da Pessoa em uma String formatada.
+ seApresenta() : void
+ getNome() : String → Retorna o nome da Pessoa.

Classe Terceirizado:
- ocupacao : String → Ocupação do Terceirizado.

+ toString() : String → Adiciona ao comportamento anterior as informações específicas em uma String formatada.
+ trabalha() : void →Mostre na tela uma mensagem, contendo o nome e ocupação, dizendo que o Terceirizado trabalhou em sua ocupação.
+ seApresenta() : void →Mostre uma mensagem na tela, contendo o nome, onde o Terceirizado se apresenta.

Classe Tecnico:
- setor : String → Setor do Tecnico.

+ toString() : String → Adiciona ao comportamento anterior as informações específicas em uma String formatada.
+ trabalha() : void →Mostre uma mensagem na tela, contendo o nome e setor, dizendo que o Tecnico trabalhou em seu setor.

+ seCapacita() : void →Mostre uma mensagem na tela, contendo o nome, onde o Tecnico realiza uma capacitação técnica.
+ seApresenta() : void →Mostre uma mensagem na tela, contendo o nome, onde o Tecnico se apresenta.

Interface Usuario:
+ autentica(usuário : String, senha : String) : boolean

Classe Login:
- usuario : String → Nome de usuário do Login.
- senha : String → Senha do Login.

+ valida(usuario : String, senha : String) : boolean → Valida as informações inseridas. Retorna **true** caso os argumentos sejam respectivamente iguais aos atributos, ou **false** caso contrário.

Classe Aluno:
- curso : String → Curso do Aluno.

+ toString() : String → Adiciona ao comportamento anterior as informações específicas em uma String formatada.
+ seApresenta() : void →Mostre uma mensagem na tela, contendo o nome e curso, onde o Aluno se apresenta.
+ estuda() : void → Mostre uma mensagem na tela, contendo seu nome, que o Aluno estudou.
+ autentica(usuário : String, senha : String) : boolean → Retorna a validação dos argumentos (informações) para logar no sistema.

Classe Professor:
- area : String → Área de atuação do Professor.
- titulacao : String → Titulação do Professor. Valores possíveis em ordem. String vazia (“”) para formação superior, “Esp” para especialista, “Ms” para mestre ou “Dr” para doutor.

+ toString() : String → Adiciona ao comportamento anterior as informações específicas em uma String formatada.
+ trabalha() : void →Mostre na tela uma mensagem, contendo o nome e titulação, dizendo que o Professor ministrou aula.
+ seApresenta() : void →Mostre uma mensagem na tela, contendo a titulação, além do nome e área de atuação, onde o Professor se apresenta.
+ seCapacita() : void → O Professor se capacita. O título obtido será o próximo, dependendo do título já possuído. Mostre uma mensagem e mude o respectivo atributo. Caso o Professor já seja doutor, ele deve cursar um pós-doutorado, sem mudança de titulação (pós-doutorado é apenas um curso, e não um título).
+ autentica(usuário : String, senha : String) : boolean → Retorna a validação dos argumentos (informações) para logar no sistema.

Classe Coordenador:
- tipo : String → Tipo da coordenação.
- descricao : String → Descrição da coordenação.

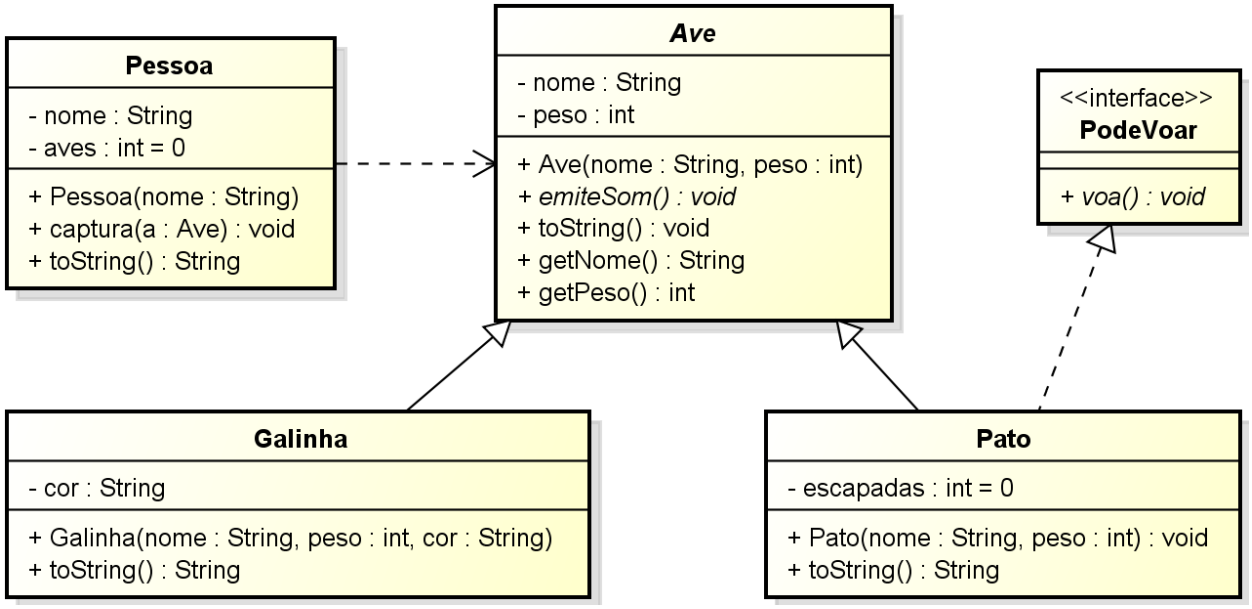
+ toString() : String → Adiciona ao comportamento anterior as informações específicas em uma String formatada.
+ trabalha() : void → Além da execução do comportamento anterior, mostre na tela uma mensagem, contendo o nome, dizendo que o Coordenador coordenou.

Classe VirtualIF:
<u>- totalrelatorios : int = 0</u> → Total de relatórios gerados no sistema VirtualIF.
<u>- totalacessos : int = 0</u> → Total de acessos com sucesso no sistema VirtualIF.

<u>+ loga(u : Usuario, usuario : String, senha : String) : void</u> → Realiza a autenticação do usuário argumento, utilizando o usuário e senha informados nos argumentos. Mostre mensagem em caso de sucesso ou falha. Em caso de sucesso incremente o respectivo atributo e mostre uma mensagem dizendo qual módulo foi aberto, de Professor ou Aluno.
<u>+ geraRelatorio(p : Pessoa) : void</u> → Mostra na tela as informações da Pessoa argumento. Incremente o respectivo atributo de contagem de relatórios.
<u>+ mostraInfo() : void</u> → Mostre na tela as informações do sistema, total de relatórios gerados e acessos com sucesso.

Exercício:
Crie objetos das classes concretas e execute todos os métodos.

7ª Questão
Faça um programa para contemplar o diagrama de classe e descrição abaixo.



Classe Ave:
- nome : String → Nome da Ave.
- peso : int → Peso da Ave, em gramas.

+ Ave(nome : String, peso : int) → Construtor.

+ emiteSom() : void
+ toString() : String → Retorna as informações da Ave.
+ getNome() : String → Retorne o nome da Ave.
+ getPeso() : int → Retorne o peso da Ave.

Classe Galinha:
- cor : String → Cor da Galinha.

+ Galinha(nome : String, peso : int, cor : String) → Construtor.

+ toString() : String → Retorna todas as informações da Galinha, com uso do método sobrescrito.
+ emiteSom() : void → Deve mostrar na tela, juntamente com seu nome, que a Galinha emite o som "Cocorico".

Interface PodeVoar:
+ voa() : void

Classe Pato:
- escapadas : int = 0 → Quantidade de escapadas que o Pato realizou.

+ Pato(nome : String, peso : int) → Construtor.

+ toString() : String → Retorna todas as informações do Pato, com uso do método sobrescrito.
+ emiteSom() : void → Deve mostrar na tela, juntamente com seu nome, que o Pato emite o som "Quack".
+ voa() : void → Deve mostrar na tela, juntamente com seu nome, que o Pato voou, escapando de uma Pessoa. Incremente o atributo correspondente.

Classe Pessoa:
- nome : String → Nome da Pessoa.
- aves : int = 0 → Quantidade de aves capturadas pela Pessoa.

+ Pessoa(nome : String) → Construtor.

+ captura(a : Ave) : void → Mostre uma mensagem, com o nome da Pessoa e da Ave, que a Pessoa tentou capturar a Ave. A Ave deve emitir seu som. Caso a Ave possa voar, ela deve voar, e não será capturada. Caso contrário, a Ave pode ser

capturada somente caso seu peso seja, no mínimo, 1kg. Mostre mensagens em todas as situações. Incremente o respectivo atributo caso a Ave seja capturada.
+ toString() : String → Retorna as informações da Pessoa.

Exercício:
Crie objetos das classes concretas e execute todos os métodos.