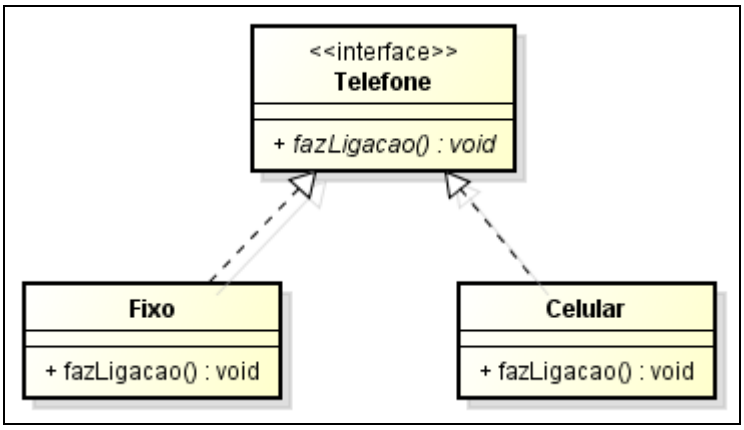
MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO TRIÂNGULO MINEIRO
CAMPUS UBERLÂNDIA CENTRO

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

LISTA 3-4

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

1ª Questão



Experimente compilar o programa, sem ter reescrito os métodos nas classes concretas, para fazer observações acerca dos erros. Verifique a obrigatoriedade de se reescrever os métodos que a Interface confere ao programa.

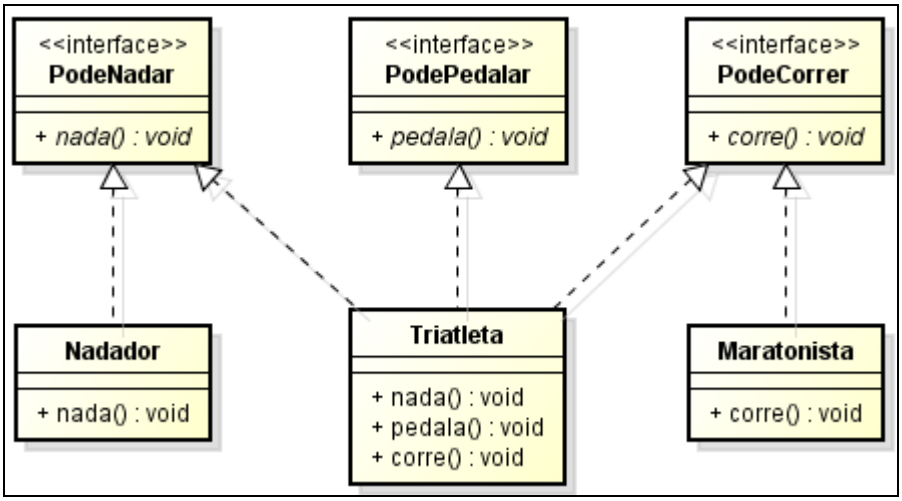
Interface Telefone:
+ fazLigacao() : void → Toda classe que se comporta como Telefone deve fazer ligações.

Classe Fixo:
+ fazLigacao() : void → Um Fixo faz ligações como telefone fixo. Mostre isso na tela.

Classe Celular:
+ fazLigacao() : void → Um Celular faz ligações como telefone celular. Mostre isso na tela.

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

2ª Questão



Experimente compilar o programa, sem ter reescrito os métodos nas classes concretas, para fazer observações acerca dos erros. Verifique a obrigatoriedade de se reescrever os métodos que a Interface confere ao programa.

Interface PodeNadar:
+ nada() : void → Toda classe que PodeNadar, nada.

Interface PodeNadar:
+ pedala() : void → Toda classe que PodePedalar, pedala.

Interface PodeCorrer:
+ corre() : void → Toda classe que PodeCorrer, corre.

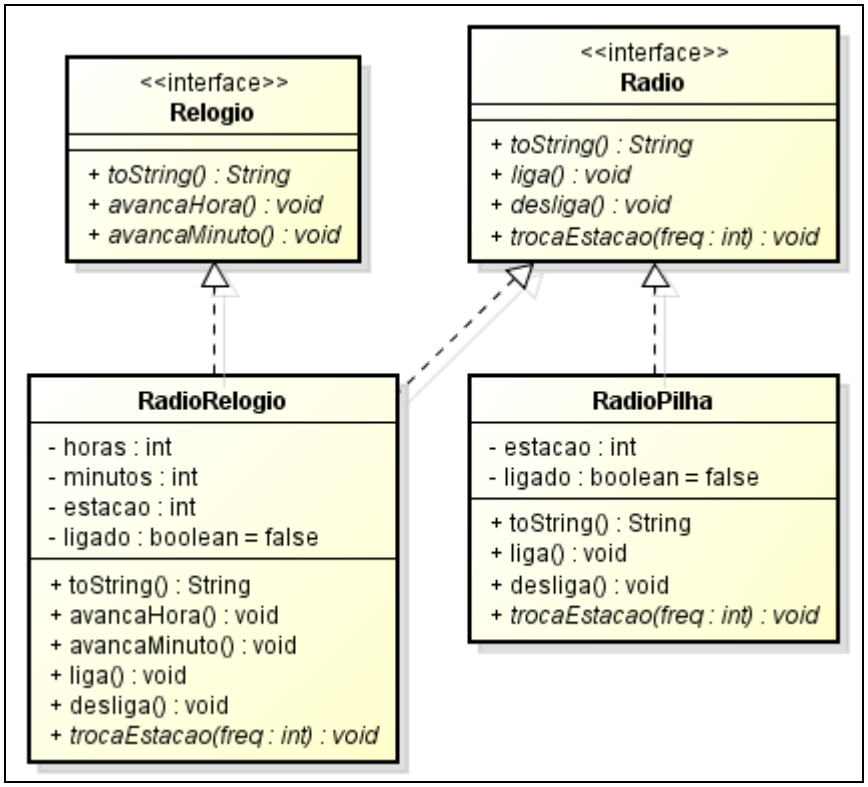
Classe Nadador:
+ nada() : void → Um nadador pula na piscina e nada. Mostre isso na tela.

Classe Triatleta:
+ nada() : void → Um triatleta pula no mar e nada. Mostre isso na tela.
+ pedala() : void → Um triatleta sobe na bicicleta e pedala. Mostre isso na tela.
+ corre() : void → Um triatleta coloca o tênis e corre. Mostre isso na tela.

Classe Maratonista:
+ corre() : void → Um maratonista parte e corre 42km. Mostre isso na tela.

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

3ª Questão



Classe RadioRelogio:
- horas : int → Horas atuais. Deve ser um valor entre 0 e 24.
- minutos : int → Minutos atuais. Deve ser um valor entre 0 e 59.
- estacao : int → Estação atual. Deve ser um valor entre 87 e 108.
- ligado : boolean = false → Define se está ligado ou desligado.

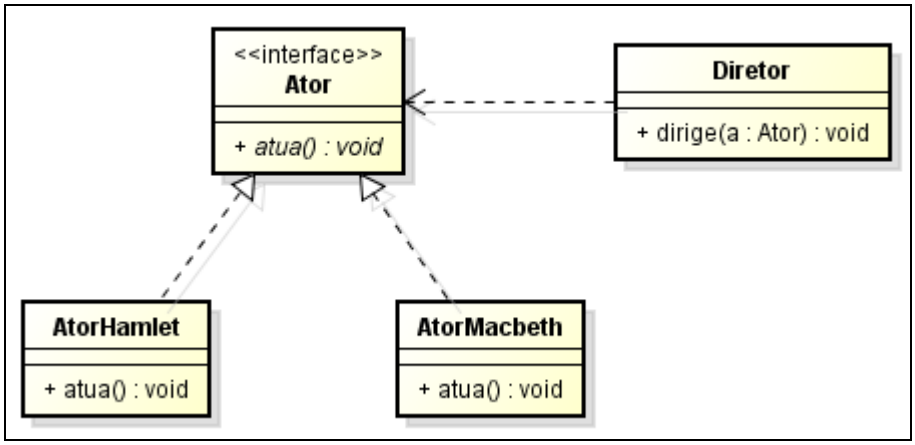
+ toString() : String → Retorne as informações do RadioRelogio. Suas horas no formato hora:minuto, com preenchimento de zeros a esquerda (opcional), além de sua estação atual e se ele está ligado.
+ avancaHora() : void → Incrementa em 1 a hora do relógio. Faça uma reflexão sobre o comportamento de um Relógio.
+ avancaMinuto() : void → Incrementa em 1 o minuto do relógio. Faça uma reflexão sobre o comportamento de um Relógio.
+ liga() : void → Liga o RadioRelogio.
+ desliga() : void → Desliga o RadioRelogio.
+ trocaEstacao(freq : int) : void → Sintoniza o RadioRelogio em uma estação conforme o argumento. Respeite os limites da estação.

Classe RadioPilha:
- estacao : int → Estação atual. Deve ser um valor entre 87 e 108.
- ligado : boolean = false → Define se está ligado ou desligado.

+ liga() : void → Liga o RadioPilha.
+ desliga() : void → Desliga o RadioPilha.
+ trocaEstacao(freq : int) : void → Sintoniza o RadioPilha em uma estação conforme o argumento. Respeite os limites da estação.

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

4ª Questão



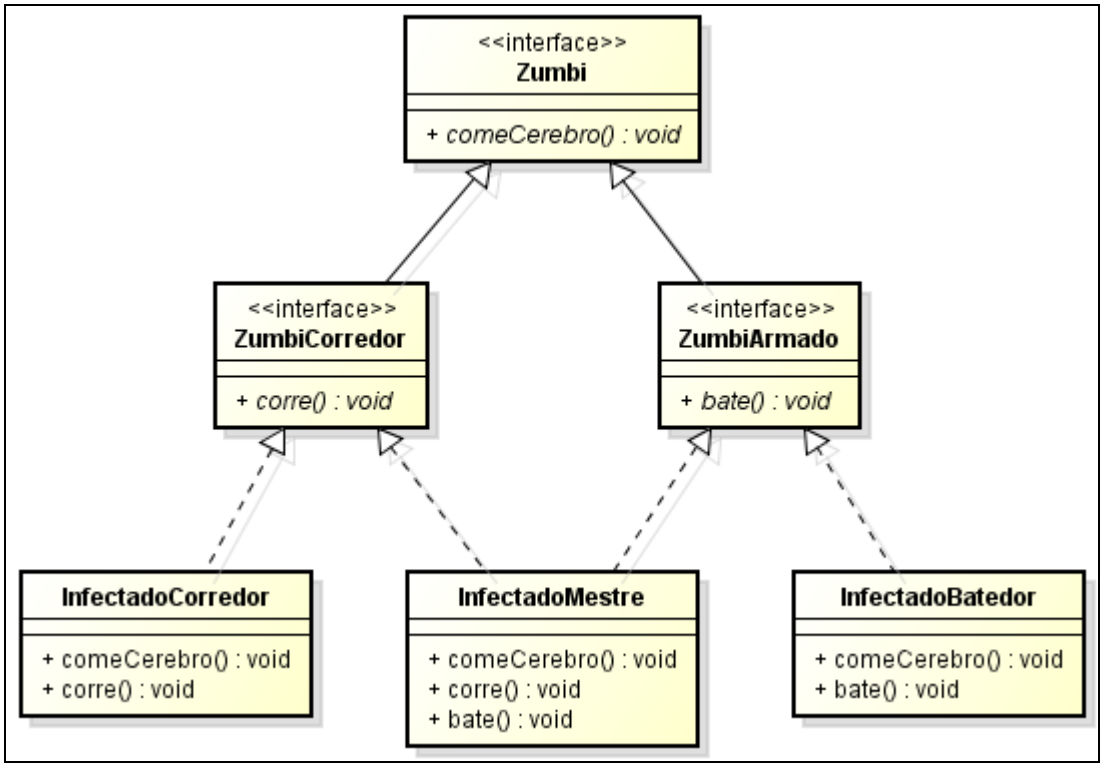
Classe AtorHamlet:
+ atua() : void → Um AtorHamlet atua proferindo a famosa frase: "Ser ou não ser, eis a questão". Mostre isso na tela.

Classe AtorMacbeth:
+ atua() : void → Um AtorMacbeth atua proferindo a famosa frase: "Tão feio e belo dia eu jamais vi". Mostre isso na tela.

Classe Diretor:
+ dirige(a : Ator) : void → Um Diretor dirige alguém que se comporta como Ator. O Ator, presente no argumento, deve atuar. Mostre na tela qual classe que se comporta como Ator está atuando.

Exercício:
Crie objetos das classes concretas e invoque os métodos delas. Faça um Diretor e dirija ambas as classes que se comportem como Ator. Crie uma nova classe que se comporte como Ator, e veja que o polimorfismo ajuda a manter o código sem mudanças.

5ª Questão



Experimente compilar o programa, sem ter reescrito os métodos nas classes concretas, para fazer observações acerca dos erros. Verifique a obrigatoriedade de se reescrever os métodos que a Interface, mesmo com herança entre elas, confere ao programa.

Interface Zumbi:
+ comeCerebro() : void → Toda classe que se comporta como Zumbi, come cerebro.

Interface ZumbiCorredor:
+ corre() : void → Toda classe que se comporta como ZumbiCorredor, corre.

Interface ZumbiArmado:
+ bate() : void → Toda classe que se comporta como ZumbiArmado, bate.

Classe InfectadoCorredor:
+ comeCerebro() : void → Um InfectadoCorredor come cérebro. Mostre isso na tela.
+ corre() : void → Um InfectadoCorredor corre. Mostre isso na tela.

Classe InfectadoMestre:
+ comeCerebro() : void → Um InfectadoCorredor come cérebro. Mostre isso na tela.
+ corre() : void → Um InfectadoCorredor corre. Mostre isso na tela.
+ bate() : void → Um InfectadoBatedor bate. Mostre isso na tela.

Classe InfectadoBatedor:
+ comeCerebro() : void → Um InfectadoBatedor come cérebro. Mostre isso na tela.
+ bate() : void → Um InfectadoBatedor bate. Mostre isso na tela.

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