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 2-6

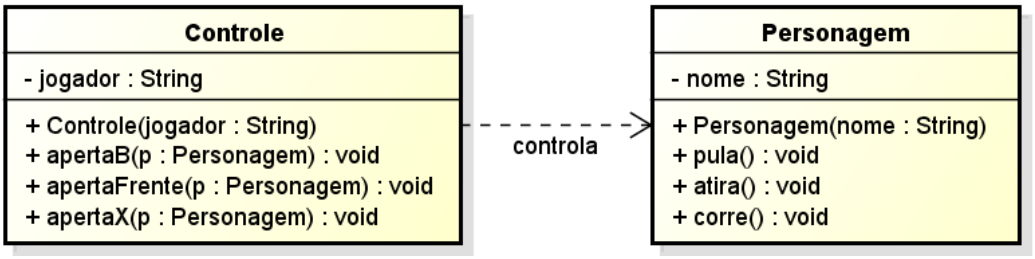
Obs: Para todos os exercícios, proceda conforme a aula. Construa objetos, contemple a multiplicidade e execute os métodos.

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 Personagem:
- nome : String → Nome da Personagem. Não pode ser uma String vazia.

+ Personagem(nome : String) → Construtor.

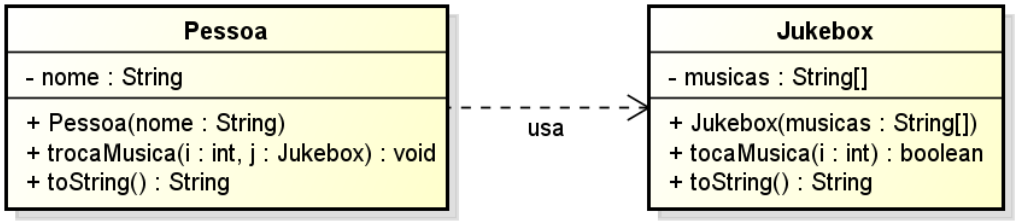
+ pula() : void → Mostra na tela que a Personagem pulou.
+ atira() : void → Mostra na tela que a Personagem atirou.
+ corre() : void → Mostra na tela que a Personagem correu.

Classe Controle:
- jogador : String → Nome do jogador do Controle. Não pode ser uma String vazia.

+ Controle(jogador : String) → Construtor.

+ apertaB(p : Personagem) : void → Indica que o botão B foi pressionado pelo jogador. Quando este botão é pressionado, a Personagem pula.
+ apertaFrente(p : Personagem) : void → Indica que o botão Frente foi pressionado pelo jogador. Quando este botão é pressionado, a Personagem corre.
+ apertaX(p : Personagem) : void → Indica que o botão X foi pressionado pelo jogador. Quando este botão é pressionado, a Personagem atira.

2ª Questão



Classe Jukebox:
- musicas : String[] → Vetor com o nome das músicas presentes no Jukebox. Não pode ser um vetor vazio.

+ Jukebox(musicas : String[]) → Construtor.

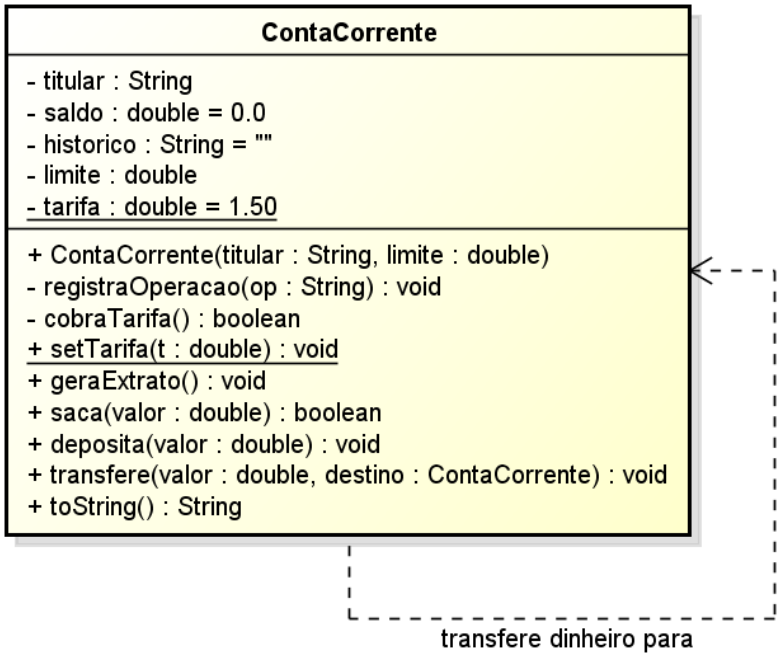
+ tocaMusica(i : int) : boolean → Toda a música presente na posição presente no argumento. Consulte o vetor de músicas. Caso a música seja tocada, mostre seu nome na tela e uma indicação da ação, além de retornar **true**. Caso contrário, retorne **false**.
+ toString() : String → Retorna as informações do Jukebox, ou seja, as músicas presentes nele.

Classe Pessoa:
- nome : String → Nome da Pessoa. Não pode ser uma String vazia.

+ Pessoa(nome : String) → Construtor.

+ trocaMusica(i : int, j : Jukebox) : void → Troca a música de determinado Jukebox, recebendo uma confirmação se a alteração foi feita. Caso seja, feita, mostre uma mensagem de sucesso. Caso contrário, mostre uma mensagem de erro.
+ toString() : String → Retorna as informações da Pessoa, mostrando seus atributos.

3ª Questão



Classe ContaCorrente:
- titular : String → Nome do titular da ContaCorrente. Não pode ser uma String vazia.
- saldo : double = 0.0 → Saldo da ContaCorrente. Deve ser sempre maior que o negativo de limite.
- histórico : String = “” → String que armazenará o histórico de operações da ContaCorrente.
- limite : double → Limite da ContaCorrente. Não pode ser um valor negativo. Nunca poderá ser menor que o negativo de saldo.
- tarifa : double = 1.50 → Tarifa cobrada nas operações de saque e geração de extrato.

+ ContaCorrente (titular : String, limite : double) → Construtor.

- registraOperacao(op : String) : void → Registra determinada operação(op) realizada no histórico, dependendo da operação realizada.
- cobraTarifa() : boolean → Cobra tarifa por operação, dependendo se há saldo disponível na ContaCorrente. Retorna **true** se a tarifa for cobrada com sucesso, e **false** caso contrário.
- setTarifa(t : double) : void → Modifica o valor da tarifa baseado no argumento. O argumento não pode ser negativo..
+ geraExtrato() : void → Mostra na tela, o histórico da conta, somente se a tarifa puder ser cobrada com sucesso. Operação tarifada. Necessita de registro no histórico.
+ saca(valor : double) : boolean → Retira determinado valor da ContaCorrente. O argumento não pode ser negativo. Operação tarifada. Necessita de registro no histórico em caso de sucesso. Além de considerar o valor do saque, deve-se considerar se a tarifa também pode ser cobrada sem que o saldo fique inválido. Caso o saque seja feito com sucesso, retorne **true**. Caso contrário, retorne **false**.
+ deposita(valor : double) : void → Deposita determinado valor na ContaCorrente. O argumento não pode ser negativo. Necessita de registro no histórico em caso de sucesso.
+ transfere(valor : double, destino : ContaCorrente) : void → Transfere determinado valor da ContaCorrente que executar o método para a ContaCorrente destino. O argumento não pode ser negativo. Necessita de registro no histórico em caso de sucesso. Aproveite o retorno do método saca para ajudar.
+ toString() : String → Retorna as informações da ContaCorrente, exceto o histórico.
