SENAI (BA)
Curso Técnico em Desenvolvimento de Sistemas
Desenvolvimento de Sistemas I

Avaliação 1: POO Java - Parte 1 – Valor 10,0

Resolver as questões abaixo usando linguagem Java.

Questão 01 (Valor 3,0):

- Implemente uma classe que represente uma **pessoa**, com os atributos **nome**, **idade** e **altura**. Crie o **construtor**, os métodos **sets** e **gets** e também um método para retornar todos dados de uma pessoa (método **toString()**).
- Implemente uma classe que simule uma **Agenda.** Essa agenda deve ser capaz de armazenar no máximo **10 pessoas** e deve ser capaz de realizar as seguintes operações:
    - **void armazenaPessoa(String nome, int idade, float altura)**
    - **void removePessoa(String nome)**
    - **int buscaPessoa(String nome)** // informa em que posição da agenda está a pessoa.
    - **void imprimeAgenda()** // imprime os dados de todas as pessoas da agenda.
    - **void imprimePessoa(int index)** // imprime os dados da pessoa que está na posição “index” da agenda.
- Escreva um programa que:
    - Adicione 10 pessoas na agenda e depois tente adicionar mais uma
    - Busque uma pessoa na agenda pelo nome
    - Remova uma pessoa da agenda pelo nome
    - Imprima os dados de uma pessoa da agenda com base no index
    - Imprima os dados de todos os contatos da agenda
- ***OBS: USAR VETOR E NÃO LISTA NA CLASSE AGENDA!!!***

Questão 02 (Valor 1,5):

- Implemente uma classe que simule o comportamento de um **Elevador** dentro de um prédio. A classe deve armazenar o **andar atual** (térreo = 0), **total de andares** no prédio (desconsiderando o térreo), **capacidade do elevador** e **quantas pessoas estão presentes** nele. A classe deve também disponibilizar os seguintes métodos:
    - **Construtor**: que deve receber como parâmetros a capacidade do elevador e o total de andares no prédio (os elevadores sempre começam no térreo e vazios);
    - **Entrar**: para acrescentar uma pessoa no elevador (só deve acrescentar se ainda houver espaço);
    - **Sair**: para remover uma pessoa do elevador (só deve remover se houver alguém dentro dele);
    - **Subir**: para subir um andar (não deve subir se já estiver no último andar)
    - **Descer**: para descer um andar (não deve descer se já estiver no térreo);
    - Métodos **gets** e **sets** para os atributos.

- Escreva um programa que:
    - Adicione o máximo de pessoas no elevador e depois tente adicionar mais uma
    - Suba até o último andar e depois tente subir mais um andar
    - Desça até o térreo e depois tente descer mais um andar
    - Removas todas as pessoas do elevador e depois tente remover mais uma

Questão 03 (Valor 2,0):

- Implemente uma classe que simule o comportamento de um **Tamagushi** (Bichinho Eletrônico):
    - Atributos: **Nome, Fome, Tédio** e **Idade**.
    - Métodos:
        - **Construtor**
        - **gets** e **sets** para os atributos
        - **void brincar (double minutos)** // permite que o usuário especifique por quanto tempo ele brinca com o bichinho, fazendo com que o nível de tédio caia.
        - **void alimentar (double quantidade)** // permite que o usuário especifique quanto de comida ele fornece ao bichinho, fazendo com que o nível de fome caia.
        - **getHumor()** // Explicação abaixo
    - Obs1: Os níveis de **fome** e **tédio** devem estar entre **0 e 100**
    - Obs2: Além dos atributos, existe mais uma informação que devemos levar em consideração, o **Humor** do nosso tamagushi, este humor é uma **combinação entre os atributos Fome e Tédio**, e, portanto, pode variar entre **0 e 200**, ou seja, é um campo calculado, então **não devemos criar um atributo para armazenar esta informação** por que ela pode ser calculada a qualquer momento através do método **getHumor()**
- Escreva um programa que:
    - Crie um **objeto** do tipo **Tamagushi**
    - Através de **prints e scanners** permita ao usuário escolher se deseja **brincar, alimentar, ou ver o nível de humor** do seu bichinho
    - Obs1: Sempre que o usuário escolher a opção brincar, **deve ser perguntado quantos minutos o usuário irá passar brincando com o bichinho**, esse valor deve ser passado para o método **brincar (double minutos)**, e após sua chamada **deve ser mostrado o nível de tédio**.
    - Obs2: Sempre que o usuário escolher a opção alimentar, **deve ser perguntado a quantidade de alimento que o usuário dará ao bichinho**, esse valor deve ser passado para o método **alimentar (double quantidade)**, e após sua chamada **deve ser mostrado o nível de fome**.

Questão 04 (Valor 3,5):

- Implemente uma classe que simule o comportamento de uma **BombaCombustível** de um posto de gasolina, essa classe deve conter os seguintes atributos:

    - **id**
    - **tipoCombustivel**
    - **valorLitro**
    - **quantidadeCombustivel**
- E os seguintes métodos:
    - **Construtor**
    - **abastecerPorValor( )** – método onde é informado o valor a ser abastecido e mostra a quantidade de litros que foi colocada no veículo
    - **abastecerPorLitro( )** – método onde é informado a quantidade em litros de combustível e mostra o valor a ser pago pelo cliente.
    - **gets** e **sets** para os atributos (**exceto setId**)
    - ***OBS: Sempre que acontecer um abastecimento é necessário atualizar a quantidade de combustível total na bomba.***
- Escreva um programa que:
    - Crie um **ArrayList** de **BombaCombustivel**.
    - Através de **prints e scanners** permita ao administrador do posto de gasolina escolher se deseja **registrar** uma nova bomba de combustível ou **editar** uma bomba de combustível existente.
    - Obs1: Sempre que o usuário escolher a opção registrar uma nova bomba, **deve ser perguntado o tipo de combustível, o valor do litro e quantos litros a bomba terá inicialmente**, esses valores, juntamente com um **id** gerado **a seu critério**, devem ser usados para a criação de um **objeto** do tipo **BombaCombustivel** que será armazenado no **ArrayList**.
    - Obs2: Sempre que o usuário escolher a opção editar uma bomba de combustível, **deve ser perguntado o id da bomba que o usuário deseja editar**, em seguida, devem ser apresentadas ao usuário 5 opções:
        - **Editar Tipo de Combustível** – pergunta ao usuário qual será o novo tipo de combustível e chama o método **setTipoCombustível().**
        - **Editar Valor do Litro** - pergunta ao usuário qual será o novo valor do litro e chama o método **setValorLitro()**
        - **Editar Quantidade de Combustível na bomba** – pergunta ao usuário qual é a quantidade atualizada de combustível na bomba e chama o método **setQuantidadeCombustível()**
        - Registrar um abastecimento por valor – pergunta ao usuário o valor abastecido e chama o método **abastecerPorValor()**
        - Registrar um abastecimento por litro – pergunta ao usuário quantos litros foram abastecidos e chama o método **abastecerPorLitro()**