Lista de Exercícios 02 – Revisão Prova01
CCF313 Programação Orientada a Objetos

QUESTÕES

1) Explique os princípios fundamentais da Programação Orientada a Objetos (POO) e por que eles são importantes no desenvolvimento de software.

2) Linguagens como Java, Python e C++ são consideradas linguagens orientada a objetos enquanto a linguagem C, por exemplo, é considerada apenas como linguagem estruturada. O que significa cada uma? Quais as vantagens de se utilizar Programação Orientada a Objetos em relação à estruturada?

3) Diferencie entre objetos e classes em POO. Forneça exemplos para ilustrar a diferença.

4) Escreva um programa em Java que modele um objeto do tipo ventilador:
   a. criar uma classe chamada "Ventilador" com atributos como velocidade, modelo, ano um estado para ligado. Inclua um construtor para inicializar esses atributos e métodos de acesso para recuperar informações sobre o ventilador e métodos para aumentar e diminuir em um nível a velocidade.
   b. Coloque um Main no seu programa para criar uma instância da classe "Ventilador" que você definiu na questão anterior. Inicialize os atributos dele corretamente e imprima os detalhes do ventilador na tela.

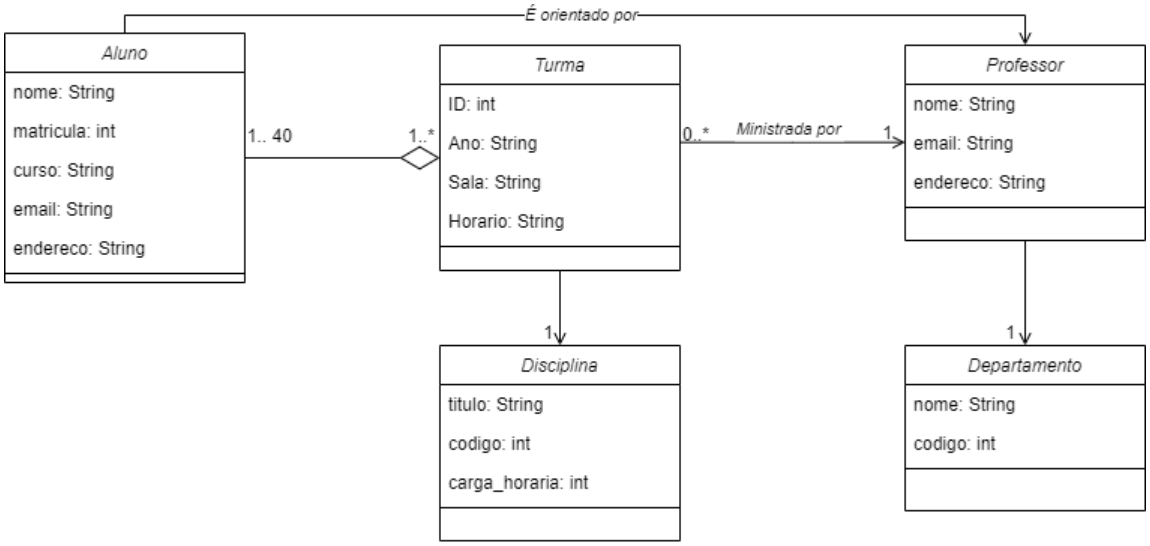
5) Crie uma classe Automóvel que possua os seguintes atributos e mpetodos: modelo, ano, quilometragem, combustível, e quilômetros_por_litro.
   - abastecer (litros): acrescenta X litros ao tanque do automóvel
   - trafegar (distancia): simula uma viagem, acrescentando X (distancia) à quilometragem do automóvel e consumindo X litros do tanque levando em consideração o atributo quilômetros_por_litro
   - combustivelNoTanque(): retorna a quantidade de combustível presente no automóvel
   - quilometragemCarro(): retorna a quilometragem do automóvel

6) Implemente em Java uma classe chamada "Produto" que represente um produto em uma loja com atributos como nome, preço e quantidade em estoque. Crie métodos para aumentar e diminuir o estoque e preço do produto.
   a. Crie uma outra classe Pedido que contenha um ID, o nome de um cliente, o valor do pedido, e uma lista de objetos Produto.
   b. Construa o método adicionaProduto. Este método deve receber um objeto tipo Produto e acrescentar este objeto ao Pedido.
   c. Construa o método calculaPrecoFinal. Este método deve somar o preco de cada produto e acrescentar 10% no valor da compra

d. Construa o método finalizarPedido. Este método deve imprimir na tela um recibo e descontar do estoque de cada produto o item comprado se possível.
e. Crie uma classe Main para testar seu código. Ela deve conter 2 objetos do tipo Produto e um Pedido. Adicione os dois produtos ao pedido, calcule o valor da compra e finalize o pedido.

7) Implemente em Java duas classes, "Animal" e "Cachorro". A classe "Cachorro" deve herdar da classe "Animal". Construa a classe "Animal" para incluir um método chamado "fazerSom()". Substitua este método na classe "Cachorro" para que o cachorro faça um som específico. Adicione um método especializado na classe "Cachorro" que imprima uma mensagem específica "Isto é um cachorro". Crie no Main do programa uma instância de "Animal" e uma instância de "Cachorro", chame o método "fazerSom()" e também o método específico criado para a classe Cachorro.

8) Considere o Diagrama de Classes abaixo, implemente o código Java correspondente ao diagrama, note que as classes não possuem métodos, não é necessário implementar nenhum método para as classes. Implemente apenas os atributos listados e as relações estabelecidas, criando novos atributos em cada classe quando for necessário para modelar o problema da forma adequada.



9) Crie uma classe chamada "Pessoa" com atributos privados, como nome, gênero, idade, altura e peso. Implemente métodos públicos de acesso (getters e setters) para esses atributos.
a. Implemente uma nova classe que herda Pessoa, chamada Cliente. Este cliente possui um atributo novo privado chamado id, um outro chamado cadastro e ainda um booleano estaCadastrado. Um método cadastrar (String cadastro) que recebe uma String, atribui ela ao Cliente e ativa o booleano.
b. Implemente uma terceira classe chamada Paciente que herda diretamente de Cliente. Esta nova classe possui dois novos métodos: um novo cadastrar que substitui o anterior que além de atribuir cadastro ao cliente, imprime na tela que é um paciente sendo cadastrado;
c. além disso, um novo método calculaIMC que utiliza da informação de altura e idade para calcular o IMC da pessoa em questão
d. Demonstre como usar esses métodos e como acessar e modificar os atributos de um objeto "Paciente" e de um "Cliente" em um Main no programa.

10) Considere o seguinte cenário em que você está desenvolvendo um sistema de gerenciamento de biblioteca em Java. Você tem duas classes principais, Biblioteca e Livro. As classes estão implementadas da seguinte forma:

```
public class Biblioteca {
	public Livro[] livros;
	public int numLivros;

	 public Biblioteca(int tamanhoMaximo) {
	    livros = new Livro[tamanhoMaximo];
	    numLivros = 0;
	}
	public void adicionarLivro(String titulo, String autor) {
		livros[numLivros] = new Livro();
		livros[numLivros].titulo = titulo;
		livros[numLivros].autor = autor;
		numLivros++;
	}
	public void listarLivros(int totalLivros) {
		System.out.println("Livros na biblioteca:");
		for (int i = 0; i < totalLivros; i++) {
			System.out.println("Título: " + livros[i].titulo +
					"  Autor:" + livros[i].autor);
		}
	}
}

public class Livro {
	public String titulo;
	public String autor;

	public Livro( ) {
		System.out.println("Livro vazio criado");
	}
	public Livro(String titulo, String autor ) {
	this.titulo = titulo;
	this.autor = autor;
	}

}

public class Main {
	public static void main(String[] args) {
		Biblioteca biblioteca = new Biblioteca(2);
		biblioteca.adicionarLivro("Java Programming", "John Smith");
		biblioteca.adicionarLivro("Introduction to OOP", "Alice Johnson");

		int totalLivros = biblioteca.numLivros;
		if (totalLivros < livros.length) {
			biblioteca.adicionarLivro("Data Structures", "Bob Davis");
		}
		totalLivros = biblioteca.numLivros;
		biblioteca.listarLivros(totalLivros);
	}
}
```

No entanto, você percebeu que essa implementação está altamente acoplada, contendo vários problemas ligados a acoplamento de classes. Sua tarefa é melhorar a implementação das classes Biblioteca e Livro para reduzir o acoplamento.

a. Descreva brevemente o que significa "acoplamento" em programação e por que é importante reduzi-lo em um sistema.
b. Reimplemente as classes Biblioteca e Livro de forma a reduzir o acoplamento entre elas. Você pode propor uma nova estrutura de classes ou fazer alterações na estrutura existente (novos métodos, atributos, etc) para torná-la mais flexível e menos acoplada.
c. Explique como sua nova implementação das classes Biblioteca e Livro reduz o acoplamento e quais benefícios isso traz para o sistema.