INSTITUTO FEDERAL Espírito Santo Campus Serra	**Curso de Engenharia de Controle e Automação**			
	Componente Curricular: Programação Orientada a Objetos.			
	Professor: Hilário Tomaz Alves de Oliveira			
	Semestre: 2022.1	**Período:** 3°	**Turma:** Noite	**Data de Entrega:** 26/06/2022

Lista de Exercícios 4 – Programação Orientada a Objetos com Python

Observações:

- A solução de cada questão que necessita de implementação de código deve estar contida em um arquivo de código na linguagem Python (extensão .py). Para padronizar utilize o seguinte padrão de nomenclatura.
  - l#NumeroListaq#NumeroQuestao.py
    - Nos quais:
      - #NumeroLista deve ser trocado pelo número da lista;
      - #NumeroQuestao deve ser trocado pelo número da questão.
  - **Exemplos:** l1q1.py, l3q4.py, l5q10.py, e assim sucessivamente
- As questões objetivas ou subjetivas podem ser resolvidas em um único arquivo texto contendo de forma explícita o número da questão e sua respectiva resposta.
- Ao final cada aluno deve enviar uma pasta compactada contendo todos os arquivos com suas soluções.

Questão 1. Descreva algumas diferenças básicas entre os paradigmas de programação estruturada e programação orientada a objetos.

Questão 2. Considerando os conceitos elementares sobre programação orientada a objetos estudados, descreve o que você entende sobre:

- Classe;

- Objeto;
- Atributo;
- Método;
- Herança;
- Polimorfismo

Questão 3. O paradigma de Programação Orientada a Objetos (POO) diz respeito a um padrão de desenvolvimento que é seguido por muitas linguagens, tais como: Python, Java, C++, entre outras. O paradigma de POO se baseia em quatro pilares. Assinale a alternativa que não representa um desses pilares.

a) Dividir para conquistar
b) Classes e objetos
c) Encapsulamento.
d) Herança.
e) Polimorfismo.

Questão 4. No contexto de programação orientada a objetos, métodos de acesso do tipo setter têm a finalidade primária de:

a) Obter o valor de um atributo.
b) Modificar o valor de um atributo.
c) Inicializar os valores de atributos de um objeto assim que ele é instanciado.
d) Contar quantas vezes o valor de um atributo foi obtido.
e) Salvar o valor de um atributo de um objeto em um banco de dados.

Questão 5. Crie uma classe para representar um jogador de futebol, com os atributos nome, posição, data de nascimento, nacionalidade, altura e peso. Crie os métodos públicos necessários para sets e gets e também um método para imprimir todos os dados do jogador. Crie um método para calcular a idade do jogador e outro método para mostrar quanto tempo falta para o jogador se aposentar. Para isso, considere que os jogadores da posição de defesa se aposentam em média aos 40 anos, os jogadores de meio-campo aos 38 e os atacantes aos 35.

Questão 6. Identifique e implemente na linguagem Python classes junto com seus atributos e métodos para o seguinte cenário: “O supermercado vende diferentes tipos de

produtos. Cada produto tem um preço e uma quantidade em estoque. Um pedido de um cliente é composto de itens, onde cada item especifica o produto que o cliente deseja e a respectiva quantidade. Esse pedido pode ser pago em dinheiro, cheque ou cartão."

Questão 7. Com base no cenário descrito a seguir, identifique as classes relevantes para o desenvolvimento do sistema solicitado, junto com seus atributos e seus relacionamentos e as implemente usando a linguagem Python.

> João é o diretor de um grande hospital e deseja implantar um sistema de prontuário eletrônico para gerenciar as informações das consultas realizadas em todo o hospital. Para isso, ele contrata a sua empresa para realizar o desenvolvimento desse sistema. Após várias entrevistas, João explica que o sistema deve contemplar módulos para manter as informações de todos os usuários do sistema (médicos, pacientes, atendentes e administradores) e também de todas as informações relacionadas as consultas dos pacientes (data, hora, prescrição do médico, medicamentos prescritos, exames solicitados, entre outras). Um paciente assim que entra no hospital precisa se dirigir à uma das recepções para ser atendido por um atendente. O atendente verifica se o paciente já possui cadastro no hospital, em caso positivo, ele apenas atualiza os seus dados cadastrais, caso seja necessário. Caso o paciente não possua cadastro, o atendente deverá cadastrar os dados do paciente. No início da consulta, o médico busca no seu computador o histórico de informações do paciente em atendimento. Com base no histórico do paciente e nas informações da consulta sendo realizada, o médico faz sua prescrição e insere todas as novas informações no prontuário eletrônico para futuras consultas. Caso seja necessário, o médico já pode deixar agendado uma nova data para uma consulta de retorno.

Questão 8. Crie uma classe Elevador para armazenar as informações de um elevador de 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:

- **Inicializar**: que deve receber como parâmetros a capacidade do elevador e o total de andares no prédio (os elevadores sempre começam no térreo e vazio);
- **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);

OBS: Encapsular todos os atributos da classe (crie os métodos set e get).

Em um arquivo separado, instancie um objeto da classe Elevador e teste os métodos criados.

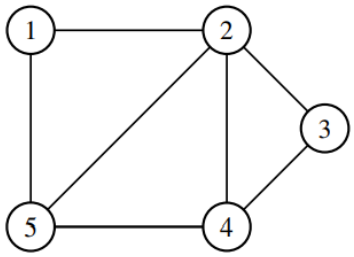
Questão 9 Crie uma classe chamada Fatura que possa ser utilizado por uma loja de suprimentos de informática para representar uma fatura de um item vendido na loja. Uma fatura deve incluir as seguintes informações como atributos: O número do item faturado, a descrição do item, a quantidade comprada do item e o preço unitário do item. Sua classe deve ter um construtor que inicialize os quatro atributos. Se a quantidade não for positiva, ela deve ser configurada como 0. Se o preço por item não for positivo ele deve ser configurado como 0.0. Forneça os métodos get/set para cada variável de instância. Além disso, forneça um método chamado calcular_valor_fatura que calcula o valor da fatura (isso é, multiplica a quantidade pelo preço por item) e depois retorna o valor. Escreva também um programa de teste (main) que demonstra as capacidades da classe Fatura.

Questão 10. Crie uma classe chamada Ingresso, que possui um valor em reais e um método imprimir_valor(). Crie uma classe IngressoVIP, que herda de Ingresso e possui um valor adicional. Crie um método que retorne o valor do ingresso VIP (com o adicional incluído). Crie um programa para criar as instâncias de Ingresso e IngressoVIP, mostrando a diferença de preços.

Questão 11. Grafo é uma estrutura de dados muito comum em computação, e os algoritmos sobre grafos são fundamentais para a área. Um grafo G = (V, A) consiste em:

- Um conjunto finito de pontos V. Os elementos de V são chamados de vértices de G.
- Um conjunto finito A de pares não ordenados de V, que são chamados de arestas de G.

Uma aresta a em A é um par não ordenado (v, w) de vértices v, w em V, que são chamados de extremidades de a. Uma aresta a em A é chamada de incidente com um vértice v em V, se v for uma extremidade de a. Um vértice v em V diz-se vizinho de outro vértice w em V se existir uma aresta a em A incidente com v e w. Um grafo pode ser representado por listas de adjacência ou por uma matriz de adjacência, como é ilustrado nas figuras a seguir:



(a) O grafo.

vértice	lista de adjacência
1	2, 5
2	1, 5
3	2, 4
4	2, 5, 3
5	4, 1, 2

(b) Listas de adjacência do grafo.

	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

(c) Matriz de adjacência do grafo.

Desenvolva na linguagem Python uma classe para representar grafos. Escolha entre a representação por listas de adjacência ou por matriz de adjacência. A classe deve oferecer as seguintes operações:

1. Inserir um vértice;
2. Inserir uma aresta entre dois vértices já inseridos no grado;
3. Listar todos os vértices;
4. Listar todas as arestas;
5. Determinar se dois vértices são vizinhos;
6. Determinar a lista de todos os vértices que são vizinhos de um dado vértice.

Considere que cada vértice é representado por um número inteiro. Escreva um script principal para testar sua classe.

Questão 12. Implemente na linguagem Python uma classe que representa um voo que acontece em determinada data e em determinado horário. Cada voo possui no máximo 200 passageiros, e a classe permite controlar a ocupação das vagas. A classe deve ter os seguintes métodos:

- **__init__** ➔ configura os dados do voo (recebidos como parâmetro) são: número do voo, cidade origem e cidade destino, data. Além disso, deve-se inicializar um

dicionário cujas as chaves devem ser os números dos assentos (1 até 200) e o valor deve representar se o assento está ou não vazio (todos os assentos inicialmente devem estar livres);

- **proximo_livre** ➔ retorna o número da próxima cadeira livre;
- **verifica_assento** ➔ verifica se o número da cadeira recebido como parâmetro está ocupada;
- **marcar_assento** ➔ ocupa determinada cadeira do voo, cujo número é recebido como parâmetro, e retorna verdadeiro se a cadeira ainda não estiver ocupada (operação foi bem sucedida) e falso caso contrário;
- **retornar_vagas** ➔ retorna o número de cadeiras vagas disponíveis (não ocupadas) no voo;

Teste sua classe com alguns exemplos.

Questão 13. Considere uma empresa que possui vários guichês para o atendimento de seus clientes. Cada cliente que chega recebe um número (sequencial, começando em 1) e fica aguardando em uma fila. Os guichês, um a cada vez, podem fazer a chamada do próximo cliente da fila, iniciando então o seu atendimento. Implemente um programa na linguagem Python para controlar o atendimento nesses guichês. O programa deve mostrar a todo instante a situação atual dos guichês, da fila e o número do próximo cliente. Seu programa deve também mostrar um menu contendo opções para inserir um novo cliente na fila, para o atendimento de um cliente da fila e para abertura/fechamento de um guichê. Observações: Os guichês deverão ser representados por uma classe que deve conter o número do guichê, o número do cliente que está em atendimento (caso não tenha nenhum ou se o guichê foi recém aberto deverá armazenar o valor None) e um histórico de todos os clientes que foram atendidos nesse guichê. Para cada cliente você deve armazenar o seu nome e o número da senha que ele recebeu.

MENU

1. **Inserir cliente na Fila**
2. **Abrir Guichê**
3. **Chamar próximo cliente**
4. **Encerrar Guichê**
5. **Fechar programa**

Observações:

- Na opção 1, seu programa deve solicitar o nome do cliente e atribuir automaticamente o seu número da senha.
- Na opção 2, seu programa deve automaticamente atribuir o número do guichê que será aberto e o inserir na lista de guichê. O número do guichê também deve ser sequencial.
- Na opção 3, você deve informar o número do guichê que atenderá o cliente e chamar o próximo cliente da fila. OBS: O cliente deve ser removido da fila de espera.
- Na opção 4, você deve solicitar o número do guichê que será fechado e o remover da lista de guichês.