**Engenharia de Sistemas e Computação**
**Lista 2 – Características das linguagens de programação I**

**Professor:** Giomar Sequeiros
**Período:** 2025 – II

**Instruções**:

- Criar um projeto para cada questão (identificado com o número da questão, exemplo Q1), o projeto deverá conter os arquivos fonte (extensão .java dentro da pasta src).
- Envie apenas a pasta src de cada projeto (questão) de forma compactada (exemplo Q1.zip). A pasta src contém o código fonte completo (pacotes e classes). Ao todo deverão ser enviados 7 arquivos .zip
- Cada questão deve ser implementada em uma ou mais classes conforme especificado no problema e deve ir acompanhada de uma classe de teste (em arquivos separados, exemplo Veiculo.java, TesteVeiculo.java). Crie pacotes se necessário.
- O **código** deve estar devidamente **comentado e indentado**.

**Q1. (1 ponto)** Crie uma classe chamada **Triangulo** contendo os três lados como atributos (a, b e c) e os seguintes métodos:

a) Método **construtor** para inicializar o triângulo, desde que seja válido. Obs. Use o método do item b.
b) Método **ehValido** que retorna verdadeiro se o triângulo é válido (o lado maior deve ser menor que a soma dos outros lados), senão retorna falso.
c) Método **tipoTriangulo** que retorna o tipo do triângulo. Se possuir os 3 lados iguais, é **equilátero**. Se possuir apenas 2 lados iguais, é **isósceles** e se possuir os 3 lados com valores diferentes é **escaleno**.
d) Método **calculaPerimetro** que retorna o perímetro do triângulo.
e) Método **calculaArea** que retorna a área do triângulo. A área do triângulo pode ser aproximada pela fórmula de Heron, conforme a Equação abaixo, onde *a*, *b* e *c* são os lados do triângulo e *p* é o seu semiperímetro (perímetro/2).

$$Area = \sqrt{p \times (p - a) \times (p - b) \times (p - c)}$$

f) Crie uma classe de teste para testar a sua classe com os diferentes métodos e situações:

**Q2. (1 ponto)** Escreva uma classe **Ponto2D** que represente um ponto no plano cartesiano com atributos x, y para as coordenadas do ponto. A classe deve oferecer os seguintes métodos:

a) Construtores que permitam a inicialização do ponto:
   i. Por default (sem parâmetros) na origem do espaço 2D;
   ii. Num local indicado por dois parâmetros (indicando o valor de abcissa e ordenada do ponto que está sendo criado);
b) Método que permita calcular a distância euclidiana de um ponto com outro, definido pela expressão abaixo:
   Sejam os pontos $P_1(x_1, y_1)$ e $P_2(x_2,y_2)$, a distância euclidiana está dado por:

$$d = \sqrt{\left(x_2 - x_1\right)^2 + \left(y_2 - y_1\right)^2}$$

c) Crie uma classe de teste para testar a sua classe com os diferentes métodos e situações:

**Q3. (1 ponto)** Considere os seguintes objetos geométricos: retângulo, caixa, círculo e cilindro. Cada objeto geométrico deve possuir métodos para obter seu perímetro (figuras 2D), sua área (externa, no caso das 3D) e volume (figuras 3D). Crie um diagrama de classes que modele objetos geométricos de forma a aproveitar ao máximo suas características comuns. Depois implemente em Java o modelo criado. As classes devem possuir construtores parametrizados, os quais devem ser utilizados para inicialização dos atributos dos objetos. Considere as informações abaixo para o cálculo do perímetro, área e volume dos objetos. Finalmente crie uma classe de teste que instancie objetos das classes e invoque as funcionalidades implementadas.

- Retângulo:
  - Área = base*altura,
  - Perímetro = 2*base + 2*altura
- Caixa:
  - Volume = base1*base2*altura,
  - Área = 2*(base1*base2 + base1*altura + base2*altura)
- Círculo:
  - Área = $3.14*(raio)^2$,
  - Perímetro = 2*3.14*raio
- Cilindro:
  - Volume = $3.14*(raio)^2*altura$,
  - Área = $2*3.14*(raio)^2 + 2*3.14*raio*altura$

**Q4. (1 ponto)** Crie uma classe Funcionario com os atributos privados nome, cpf e salario; e um construtor que deve receber como parâmetro nome e cpf. Todos os atributos devem ter propriedades (getters) definidos.

a) Crie uma classe TrabalhadorAssalariado e outra TrabalhadorHorista, ambas herdando de Funcionario
b) A classe TrabalhadorAssalariado possui o método definirSalario(salario) que recebe um valor como parâmetro e o atribui ao atributo salario.
c) A classe TrabalhadorHorista possui os atributos privados valorHora e horasTrabalhadasMes, com seus respectivos métodos de acesso (getters e setters).
d) A classe TrabalhadorHorista também possui o método calcularPagamento(), que ao ser invocado deve calcular o valor do salário e preencher este atributo. O salário é obtido multiplicando-se as horas trabalhadas no mês pelo valor da hora. Crie validações que verifiquem se os atributos necessários para o cálculo do salário estão preenchidos. Se não estiverem, emita um aviso.
e) Crie uma classe de teste e no método main crie instâncias de cada uma das classes e invoque os métodos implementados.

**Q5. (1 ponto)** Considere o projeto **ContaHeranca** desenvolvido na Aula 11 (Veja os slides do Classroom).

a) Adicione a classe ContaPoupanca derivada de conta. Esta classe possui uma constante que representa um índice de reajuste mensal. Acrescente um método reajustar que permita atualizar o saldo da conta somando o percentual determinado pelo índice.
b) Adicione o método transferência na classe Conta que receba uma conta destino como parâmetro e um valor. Depois sobrescreva o método na classe ContaEspecial.
c) Crie uma classe Banco que possua um conjunto de contas e permita determinar a quantidade de contas e imprimir todas as contas que pertencem a ele.
d) Modifique a classe Teste de modo a provar todas as funcionalidades implementadas

**Q6. (2,5 pontos)** Criar um programa que permita analisar **o banco de dados Iris** e permita criar um **modelo** de **aprendizado máquina** capaz **caracterizar** uma flor Iris dadas as medidas das pétalas e sépalas.

O banco de dados está disponível em um arquivo de texto **iris.csv** (formato csv - valores separados com vírgulas) disponível no link a seguir:
https://drive.google.com/file/d/1dGHLZoD5-4cnKv2F_tO7Zpw5DxmlA6Ss/view?usp=share_link

Este banco de dados contém um conjunto de 151 linhas com cinco colunas (atributos) listados a seguir:
- Comprimento da sépala
- Largura da sépala,
- Comprimento da pétala,
- Largura da pétala
- Tipo

A continuação mostra-se as 4 primeiras linhas do arquivo **iris.csv**

```
comprimento_sepala,largura_sepala,comprimento_petala,largura_petala,tipo
5.1,3.5,1.4,0.2,setosa
4.9,3.0,1.4,0.2,setosa
4.7,3.2,1.3,0.2,setosa
```

A primeira linha contém apenas os nomes das colunas. A partir da segunda linha são armazenados 5 valores (separados por vírgula). Os quatro primeiros valores são numéricos e representam as dimensões de uma flor íris (em centímetros), o último valor é categórico podendo ser: setosa, versicolor e virginica. Ao todo são 150 registros de medições com 50 registros para cada tipo de flor de íris (veja figura abaixo)


Acrescente as seguintes funcionalidades:

a) Crie uma **classe** chamada **Iris** com atributos comprimento_sepala, largura_sepala, comprimento_petala, largura_pelata e tipo. Os quatro primeiros atributos armazenam dados numéricos que representam as dimensões da flor iris e o atributo tipo armazena uma String que pode tomar um dos valores a seguir: “Setosa”, “Versicolor”, “Virginica” que representa o tipo de iris. Adicione os construtores, métodos getter e setter e o método toString() que deve retornar uma string concatenando os valores dos atributos separados por vírgula. Similar ao mostrado abaixo:

    5.1,3.5,1.4,.2,Setosa

b) Na classe Iris crie um método calcularDistância que receba um objeto do tipo iris. Este método calcula a **distância de Manhattan** entre duas instâncias do tipo iris e é definido por:

distancia (iris1, iris2) = |comprimento_sepala1 - comprimento_sepal2 | + | largura_sepala1- largura_sepala2 | + | comprimento_petala1 - comprimento_petala2| + | largura_pelata2 - largura_pelata2 |

Ou seja, é igual ao somatório dos valores absolutos da diferença dos atributos dos objetos (Sem considerar a classe)

c) Crie uma classe **DBIris** que tenha o atributo dataset do tipo ArrayList<Iris> que permita armazenar um conjunto de objetos do tipo iris. Crie um **construtor** que aceita um arquivo de texto como parâmetro e permita inicializar o atributo dataset com todos os dados do arquivo. Crie o método **imprimir()** que permita mostrar todos os elementos do arquivo.

d) Na classe DBIris Crie um método chamado de **calcularEstatisticas()** que imprima medidas estatísticas básicas dos elementos do dataset como: total de elementos, mínimo valor, máximo, média aritmética e desvio padrão de cada atributo numérico da classe Iris. Para o caso do atributo tipo deve imprimir a frequência. Por exemplo:

```
Resumo base de dados iris: 150 elementos
   comprimento_sepala:
     Minimo: 4.3
     Maximo: 7.9
     Média: 5.843
     Desvio padrão: 0.828
   largura_sepala:
.......
   comprimento_petala:
.......
   largura_pelata:
.......
   tipo:
Setosa: 50 ocorrências
Virgínica: 50 ocorrências
Versicolor: 50 ocorrências
```

e) Na classe DBIris Crie um método chamado de **knn(Iris iris, int k)** que receba um objeto do tipo iris com atributo tipo =”?” e um parâmetro k do tipo inteiro. Este método deve determinar a categoria ao qual o parâmetro de entrada pertence usando o algoritmo knn. O algoritmo KNN basicamente calcula a distância de Manhattan (implementado na classe iris) do objeto de entrada com todos os elementos da base de dados. Ordena os valores das distâncias em uma lista (de menor a maior) e verifica qual é o tipo majoritário nos primeiros k elementos ordenados.

**Dica.** Para facilitar os experimentos utilize k=1, ou seja, o método knn deve retornar o valor do atributo tipo do objeto que apresente a menor distância.

Finalmente crie uma classe teste para testar todas as funcionalidades implementadas com um menu amigável de opções.

Exemplo básico de uso

```
String url = “c:/iris.csv”;
DBIris db = new DBIris(url);
db.imprimir();
String classe = db.knn(new Iris(4.9, 3.1, 1.5, 0.4,”?”), 1);
System.out.println(“A classe é: ”+ classe);
```

**Observações**: Faça o tratamento de erros necessário usando os comandos try-catch-finally e throw

**Q7**. **(2,5 pontos)** Crie uma agenda de contatos usando arquivos com os seguintes requisitos:

- Crie os pacotes “persistencia”, “negocio” e “apresentacao"
- No pacote “negocio” crie uma classe Contato com atributos: id, nome, sobrenome, e-mail, celular, tipo (ex: trabalho, família, amigos, etc.). Na mesma classe, sobre escreva o método toString de modo a retornar uma cadeia da seguinte forma:

    1, “João”,”da Silva”,”joao@email.com”,”(21) 98765-4321”, “Família”

Ou seja, concatenar todos os atributos separados por vírgula. Atributos do tipo string deve estar entre aspas duplas.

- No pacote “persistencia” crie a interface IContatoDAO conforme o código a seguir:

```
import java.util.List;

public interface IContatoDAO {

   public void salvar(Contato c); //salva os dados de um contato em um arquivo de texto
   public void atualizar(Contato c); //altera os dados de um contato que está no arquivo de texto
   public void deletar(int id); //Apaga os dados de um contato que está no arquivo de texto
   public Contato getById(int id);  // Dado o id, recupera os dados de um contato
   public List<Contato>  getAll(); //Recupera todos os dados de um contato e retorna uma lista
}
```

- No pacote “persistencia” crie a classe ContatoDAO que implemente a interface IContatoDAO. Os métodos a serem implementados devem utilizar arquivos para persistir os dados. Por exemplo, o método salvar deve receber um objeto Contato e armazenar os dados no final de um arquivo. O método atualizar deve permitir modificar os dados de um contato cujo id coincida com o atributo id do parâmetro contato.

- No pacote “apresentacao” crie a classe AppCliente que permita testar todas as funcionalidades implementadas do IContatoDAO.

Exemplo de uso:

```
Contato c = new Contato(1, “João”,”da Silva”,”joao@email.com”,”(21) 98765-4321”, “Família” );
IContatoDAO dao = new ContatoDAO();
dao.salvar(c); //Os dados foram gravados em um arquivo de texto
```

**Dicas**:
- A classe ContatoDAO deve ter um atributo estático contendo a url do arquivo.
- Adicione uma classe com métodos estáticos que permita realizar operações de leitura e escrita de dados em arquivos
- Pode utilizar um ArrayList para carregar os dados do arquivo em memória e realizar as operações de atualização/remoção e finalmente gravar os dados no arquivo.
- Não é possível modificar o id de um contato