UNIVERSIDADE FEDERAL DE SÃO CARLOS
DEPARTAMENTO DE COMPUTAÇÃO
Rodovia Washington Luiz, Km. 235 – Caixa Postal 676
Fone: (16) 33518230 – Fax: (16) 33518233
CEP 13565-905 – São Carlos – SP – Brasil

Exercício sobre Coesão

Programação Orientada a Objetos Avançada

1) Considere a classe baixo que foi retirada de um sistema de supermercado disponível no GitHub (https://github.com/catarinaribeir0/Supermarket).
Faça uma análise da coesão dessa classe. Pode-se perceber que ela possui vários problemas menores, como por exemplo o nome pouco significativo de alguns métodos e atributos. **Se você considera que esta classe possui coesão baixa, o que você faria para aumentar a coesão dela ?**

```
public class Estoque {


   private Map<Integer, Produto> identificador = new HashMap();
   private Map<Produto, Float> prateleiras = new HashMap();


   public Produto buscarProduto(Integer codProd) {
      return identificador.get(codProd);
   }
   public void gerarRelatorioEstoque() {
      FileWriter estoqueFinalDia;
      FileReader arquivo;
      DateFormat dat = DateFormat.getDateInstance(DateFormat.LONG, new Locale("pt", "BR"));
      try {
         estoqueFinalDia = new FileWriter(new File("RelatorioEstoque " + dat.format(new Date()) + ".txt"));
         arquivo = new FileReader("Produtos.txt");
         BufferedReader estoqueInicioDia = new BufferedReader(arquivo);
         PrintWriter escreverArquivo = new PrintWriter(estoqueFinalDia);
         escreverArquivo.write("|---------- CDL SUPERMERCADOS ----------|\n");
         escreverArquivo.write("|-------------------------------------------------|\n");
         escreverArquivo.write("|---------- RELATORIO DE ESTOQUE ----------|\n");
         escreverArquivo.write("|  DATA: " + dat.format(new Date()) + "                    |\n");
         escreverArquivo.write("|        ESTOQUE DISPONIVEL - INICIO DO DIA        |\n");
         escreverArquivo.write(" PRODUTO                          QUANTIDADE\n");
         String linha = estoqueInicioDia.readLine();
         DecimalFormat df = new DecimalFormat("0.###");
         do {
            escreverArquivo.write(" " + linha);
            for (int i = 0; i < 2; i++) {
```

```
                linha = estoqueInicioDia.readLine();
            }
            escreverArquivo.write("      "+ df.format(new Float(linha))+"\n");
            for (int i = 0; i < 2; i++) {
                linha = estoqueInicioDia.readLine();
            }
        } while (linha != null);
        escreverArquivo.write("|                                                  | \n");
        escreverArquivo.write("|          ESTOQUE DISPONIVEL - FIM DO DIA          | \n");
        escreverArquivo.write(" PRODUTO                              QUANTIDADE\n");
        Set<Map.Entry<Produto, Float>> entrySet = prateleiras.entrySet();
        for(Map.Entry<Produto,Float> entrada : entrySet){
            escreverArquivo.write(" " + entrada.getKey() + "      " + df.format(entrada.getValue().floatValue()) +
"\n");
        }
        escreverArquivo.write("|                                                  | \n");
        escreverArquivo.write("|--------------------------------------------------| \n");

        estoqueFinalDia.close();
      } catch (Exception e) {
        e.printStackTrace();
      }
  }
  //Calculos com unidades
  public float quantidadeEstoque(Produto produto) {
    return prateleiras.get(produto);
  }
  public void inserirProduto(float valor, String nome, Float quantidade, int tipo) {
    Produto produto = new Produto(valor, nome, tipo);
    this.identificador.put(produto.getCodigo(), produto);
    this.prateleiras.put(produto, quantidade);
 }
  public void inserirPrateleira(Produto produto, Float quantidade) {
    Float quant = prateleiras.get(produto);
    quant += quantidade;
    prateleiras.put(produto, quant);
  }
  public boolean retirarPrateleira(Produto produto, Float quantidade) {
```

```
      Float quant = prateleiras.get(produto);

      quant -= quantidade;

      if (quant < 0) {

         return false;

      }

     prateleiras.put(produto, quant);

      return true;

   }

}
```

2) Todos os métodos abaixo encontram-se em uma única classe, mas nitidamente a classe possui baixa coesão. Refatore essa classe para que as classes resultantes tenham alta coesão.

   - gerarNotaFiscal()
   - void calcularImposto()
   - float getValorDoImposto()
   - aplicarDescontoNoPedido()
   - calcularDesconto();
   - removerProdutoPedido()
   - inserirProdutoPedido()
   - boolean estePedidoFoiEntregue()
   - getValorFrete()
   - calcularValorFrete()

3) Aplique a métrica LCOO na classe Point e demonstre todos os passos da aplicação para se obter o resultado final.

4) Aplique a métrica LCOO na classe abaixo.

```
public class C {

        private int att1;
        private String att2;

        public m1(){

                usa os atributos att1 e att2;

        }

        public m2() {
```

```
            usa os atributos att1 e att2;


        }

        public m3() {

            usa os atributos att1 e att2;

        }
```

5) Aplique a métrica LCOO na classe abaixo.

```
public class C {

        private int att1;
        private String att2;
        private int att3;
        private String att4;


        public m1(){
            usa os atributos att1 e att2;
        }

        public m2() {
            usa os atributos att1 e att2;
        }

        public m3() {
            usa os atributos att1 e att2;
        }

        public m4(){
            usa os atributos att3 e att4;
        }

        public m5() {
            usa os atributos att3 e att4;
        }

        public m6() {
            usa os atributos att3 e att4;

        }
```