FACULDADE DE COMPUTAÇÃO E INFORMÁTICA
Linguagem de Programação II
PROVA PARCIAL 1 – 02/10/2015
Prof. Tomaz Mikio Sasaki

NOME: **GABARITO** Turma: TIA:

Exemplos:

Os trechos de código a seguir são exemplos que podem ou não ser úteis na resolução dos exercícios da prova.

A execução da classe **Exemplo1** abaixo lê um arquivo texto chamado **gasto-mes-atual.txt**, imprime os gastos do usuário na tela do computador e apresenta o total dos gastos.

```
public class Exemplo1 {
    public static void main(String[] args)
           throws FileNotFoundException, IOException {
        FileReader fr =
               new FileReader("gastos-mes-atual.txt");
        BufferedReader br = new BufferedReader(fr);
        String linha;
        int total = 0;
        while ( (linha = br.readLine()) != null ) {
            String[] partes = linha.split(":");
            String descricao = partes[0];
            String categoria = partes[1];
            int valor = Integer.parseInt(partes[2]);
            System.out.println(categoria + " - R$ " +
                           valor + " - " + descricao);
            total += valor;
        }
        System.out.println("Total dos gastos: R$ " +
                                              total);
        br.close();
    }
}
```

A execução da classe **Exemplo2** abaixo solicita que o usuário forneça os dados de um gasto, cria uma instância da classe Gasto e serializa esta instância para o arquivo **gastos-mes-atual.dat**.

```
public class Exemplo2 {
    public static void main(String[] args)
                        throws FileNotFoundException,
IOException {
        Scanner entrada = new Scanner(System.in);
        System.out.print("Descrição: ");
        String d = entrada.nextLine();
        System.out.print("Categoria: ");
        String c = entrada.nextLine();
        System.out.print("Valor (em reais): ");
        int v = entrada.nextInt();
        ObjectOutputStream oos =
         new ObjectOutputStream(
         new FileOutputStream("gastos-mes-atual.dat"));
        String linha;
        Gasto gasto = new Gasto(d,c,v);
        oos.writeObject(gasto);
        br.close();
        oos.close();
    }
}
```

A execução da classe **Exemplo3** abaixo solicita as informações necessárias (url e pasta de destino) para efetuar um download e inicia o download em uma *thread* separada. A tarefa de download é definida na classe **RunnableDownload** (cujo código não está sendo fornecido aqui).

```
public class Exercicio8 {

    private static Scanner entrada = new
Scanner(System.in);

    public static void main(String[] args) {
        boolean finalizar = false;
        while (!finalizar) {
            System.out.println("MENU INICIAL");
            System.out.println("(1) Novo download");
            System.out.println("(2) Finalizar");
            int opcao = entrada.nextInt();
            entrada.nextLine();
            switch (opcao) {
                case 1:
                    menuDownload();
                    break;
                case 2:
                    finalizar = true;
                    break;
                default:
                    System.out.println(
                                    "Opção inválida!");
            }
        }
    }
    private static void menuDownload() {
        System.out.println("\nDOWNLOAD");
        System.out.print("- URL: ");
        String url = entrada.nextLine();
        System.out.print("- Pasta de destino: ");
        String pasta = entrada.nextLine();
        Runnable r = new RunnableDownload(url, pasta);
        Thread t = new Thread(r);
        t.start();
    }
}
```

Questão 1 (2 pontos) - Você precisa desenvolver um programa que leia os dados de um pedido. As informações do pedido são armazenadas em um arquivo texto chamado **itens_pedido.txt**, onde cada linha corresponde a um item do pedido e possui o formato

produto*|*quantidade*|*preço unitário

Exemplo de conteúdo do arquivo pedido.txt:

```
Sabão em pó|5|15.50
Leite|7|5.00
Arroz Catete|2|35.00
Refrigerante 2L|5|5.30
```

Escreva um programa em Java que leia o arquivo **itens_pedido.txt** e apresente na tela do computador todos os itens do pedido no formato:

Qtd:***quantidade*** - Produto:***produto*** - Subtotal: R$ ***subtotal***

Onde o subtotal é a quantidade do produto multiplicada pelo seu preço unitário. Para o exemplo de arquivo fornecido anteriormente, o resultado na tela deveria ser:

```
Qtd:5 - Produto:Sabão em pó - Subtotal: R$ 77.50
Qtd:7 - Produto:Leite - Subtotal: R$ 35.00
Qtd:2 - Produto:Arroz Catete - Subtotal: R$ 70.00
Qtd:5 - Produto:Refrigerante 2L - Subtotal: R$ 26.50
```

Ao final da listagem, o programa deverá apresentar também o valor total do pedido.

```
public class Questao1 {
    public static void main(String[] args) throws FileNotFoundException, IOException {

      FileReader fr = new FileReader("itens_pedido.txt");
      BufferedReader br = new BufferedReader(fr);
      double total = 0;
      String linha;
      while ( (linha = br.readLine()) != null ) {
        String[] p = linha.split("|");
        String descricao = p[0];
        int quantidade = Integer.parseInt(p[1]);
        double pUnitario = Double.parseDouble(p[2]);
        double subTotal = quantidade * pUnitario;
        String msg = "Qtd:" + quantidade + " - ";
        msg += "Produto:" + descricao + " - ";
        msg += "Subtotal: R$ " + subTotal;
        System.out.println(msg);
        total += subTotal;
      }
      br.close();
      System.out.println("Total do pedido: R$ " + total);
}
```

Questão 2 (2 pontos) - Desenvolva um programa em Java que leia o arquivo **itens_pedido.txt** (especificado na Questão 1) e salve os seus dados em um arquivo utilizando o recurso de serialização. Para isso:

a) Declare a classe **ItemPedido** com os atributos, métodos e o que mais for necessário para que suas instâncias possam ser serializadas (não é necessário escrever os **getters** e **setters**; coloque apenas um comentário para indicar que eles existem).

```
public class ItemPedido implements Serializable {
    private String descricao;
    private int quantidade;
    private double precoUnitario;

    public ItemPedido(String d, int q, double p) {
        descricao = d;
        quantidade = q;
        precoUnitario = p;
    }

    // getters, setters, construtor default
}
```

b) Escreva um programa em Java que leia o arquivo **itens_pedido.txt**, crie as instâncias de **ItemPedido** correspondentes e serialize estes objetos para o arquivo **itens_pedido.dat**.

```
public class Questao2b {
    public static void main(String[] args) throws FileNotFoundException, IOException {

    BufferedReader br = new BufferedReader(
                        new FileReader("itens_pedido.txt"));
    FileOutputStream fos = new FileOutputStream(
                                      "itens_pedido.dat");
    ObjectOutputStream oos = new ObjectOutputStream(fos);
    String linha;
    while ( (linha = br.readLine()) != null) {
      String[] p = linha.split("|");
      String desc = p[0];
      int qtd = Integer.parseInt(p[1]);
      double pUnit = Double.parseDouble(p[2]);
      ItemPedido item = new ItemPedido(desc, qtd, pUnit);
      oos.writeObject(item);
    }
    br.close();
    oos.close();
}
```

Questão 3 (2 pontos) - Sua aplicação deverá persistir os dados dos itens do pedido em uma tabela na base de dados Java DB. A tabela foi criada com o seguinte script:

```
CREATE TABLE itens_pedido (
    produto VARCHAR(255) NOT NULL PRIMARY KEY,
    quantidade INT NOT NULL,
    preco_unitario DECIMAL(8,2) NOT NULL
);
```

Complete os trechos em branco de forma que o programa cumpra o seu propósito (as importações de classes foram omitidas propositalmente).

O programa a seguir deverá consultar os dados da tabela **itens_pedido**, imprimir as informações na tela do computador, solicitar as informações de um novo item do pedido e inseri-las na tabela.

```
public class AppSelectTimes {
    public static void main(String[] args)
            throws ClassNotFoundException, SQLException {
        Class.forName("org.apache.derby.jdbc.ClientDriver");

        Connection conexao;
        String url = "jdbc:derby://127.0.0.1:1527/times";
        String usuario = "app";
        String senha = "app";

        conexao = DriverManager.getConnection(url, usuario, senha);

        PreparedStatement stSelect;

        String sqlSelect = "SELECT * FROM itens_pedido ";
        stSelect = conexao.prepareStatement(sqlSelect);

        ResultSet  resultados = stSelect.executeQuery ();

        while (resultados.next ()) {

           System.out.println("Produto: " + resultados.getString("produto"));

           System.out.println("Quantidade: " + resultados.getInt("quantidade"));

           System.out.println("Preço unitário: " + resultados.getBigDecimal("preco_unitario"));
           System.out.println("--------------------");
        }
        PreparedStatement stInsert;

        String sqlInsert = "INSERT INTO itens_pedido VALUES (?,?,?)";
        stInsert = conexao.prepareStatement(sqlInsert);

        Scanner entrada;
        String sValor;
        entrada = new Scanner(System.in);
        System.out.println("Cadastro de um novo item do pedido:");
        System.out.print("Produto: ");
        String produto = entrada.nextLine();
        System.out.print("Quantidade: ");
        sValor = entrada.nextLine();
        int qtde = Integer.parseInt(sValor);
        System.out.print("Preço unitário: ");
        sValor = entrada.nextLine();
        BigDecimal preco = new BigDecimal(sValor);

        stInsert.setString(1, produto);

        stInsert.setInt(2, qtde);

        stInsert.setBigDecimal(3, preco);

        stInsert.executeUpdate();
        conexao.close();
    }
}
```

Questão 4 (2 pontos) - Vamos definir as seguintes interfaces:

```
import java.sql.Connection;

public interface ConexaoInterface {
    Connection getConnection();
    void close();
}
```

```
import java.util.List;

public interface ItemPedidoDaoInterface {
    List<ItemPedido> listarTudo();
    void inserir(ItemPedido item);
}
```

A classe **ItemPedido** é a que foi definida no item **a** da **Questão 2**.

Complemente o código a seguir de forma a implementar uma classe que encapsule as duas operações de persistência definidas na interface **ItemPedidoDaoInterface** para serem realizadas na tabela **itens_pedido** (definida na **Questão 3**) da base de dados.

```
public class ItemPedidoDaoRelacional implements ItemPedidoDaoInterface {
    private ConexaoInterface conexao;
    public ItemPedidoDaoRelacional(ConexaoInterface conexao) {
        this.conexao = conexao;
    }
    @Override
    public List<ItemPedido> listarTudo() {
        List<ItemPedido> itens;
        itens = new ArrayList<>();
        try {
            PreparedStatement st;

      String sql = "SELECT * FROM itens_pedido";
            st = conexao.getConnection().prepareStatement(sql);

           ResultSet resultados = st.executeQuery();
           while (resultados.next ()) {

             String d = resultados.getString("produto");
             int qtd = resultados.getInt("quantidade"));
             double p = resultados.getDouble("preco_unitario"));
             ItemPedido item = new ItemPedido(d, qtd, p);
             itens.add(item);
          }

        } catch (Exception ex) {
            ex.printStackTrace();
        }
        return itens;
    }
    @Override
    public void inserir(ItemPedido item) {
        try {
            PreparedStatement st;

      String sql = "INSERT INTO itens_pedido VALUES(?,?,?)";
            st = conexao.getConnection().prepareStatement(sql);

      st.setString(1, item.getDescricao());
      st.setInt(2, item.getQuantidade());
      st.setDouble(3, item.getPrecoUnitario());

      st.executeUpdate();

        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}
```

Questão 5 (1 ponto) - Um item do pedido não deveria ter um valor negativo no seu preço unitário.

a) Declare uma exceção do tipo ***checked*** chamada **PrecoException**.

```
public class PrecoException extends Exception {}
```

b) Altere a classe **ItemPedido** (a classe definida na Questão 2) de forma que o método *setter* correspondente ao preço unitário do item lance a exceção **PrecoException** caso o valor do preço fornecido como parâmetro seja negativo.

```
public class ItemPedido implements Serializable {

  public void setPrecoUnitario(double p) throws PrecoException {
    if (p < 0) {
      throw new PrecoException();
    }
    precoUnitario = p;
  }

// outros membros da classe
}
```

Questão 6 (1 ponto) - Um colega da sua equipe desenvolveu a classe **RunnableDownload**, que é uma implementação da interface **Runnable,** e faz o download do arquivo.

RunnableDownload
RunnableDownload(url:String,pasta:String)
run():void

Veja o código das duas classes abaixo:

```
public class Questao5a {
    public static void main(String[] args) {
        Runnable r1 = new RunnableDownload("http://www.mackenzie.br/cursos.pdf", "C:\Temp");
        Thread t1 = new Thread(r1);
        t1.start();
        Runnable r2 = new RunnableDownload("http://www.mackenzie.br/vestibular.pdf", "C:\Temp");
        Thread t2 = new Thread(r2);
        t2.start();
    }
}
```

```
public class Questao5b {
    public static void main(String[] args) {
        Runnable r1 = new RunnableDownload("http://www.mackenzie.br/cursos.pdf", "C:\Temp");
        r1.run();
        Runnable r2 = new RunnableDownload("http://www.mackenzie.br/cursos.pdf", "C:\Temp");
        r2.run();

    }
}
```

Explique qual será a diferença que o usuário perceberá ao comparar a execução destas duas classes.

Na classe Questao5a, os dois downloads serão executados "simultaneamente", enquanto que na classe Questao5b os downloads serão executados sequencialmente (o segundo download só terá início após o término do primeiro)