Universidade Federal de Ouro Preto - UFOP
Instituto de Ciências Exatas e Biológicas - ICEB
Departamento de Computação - DECOM
Disciplina: Engenharia de Software II – BCC323
Alunos: Mateus Lana, Ricardo David e Thiago Santana
Matrículas: 15.1.4340, 15.2.4999 e 15.1.4313

TRABALHO PRÁTICO - SISTEMA DE VENDAS: NoMoreDodoiWeb

1.0 REQUISITOS

1.0.1 Requisitos Funcionais

- **RF01** – O sistema permitirá efetuar cadastro de produtos, como por exemplo, remédios que é o foco principal do sistema.
- **RF02** – O sistema permitirá realizar a venda de produtos, registrando o lucro obtido.
- **RF03** – O sistema permitirá controlar o estoque de produtos.
- **RF04** – O sistema permitirá gerar um relatório de vendas.
- **RF05** – O sistema permitirá a emissão de um comprovante de pagamento ao cliente.
- **RF06** – O sistema permitirá gerar relatórios contendo os produtos cadastrados.
- **RF07** – O sistema permitirá consultar o valor de um determinado produto pelo nome ou código.
- **RF08** – O sistema permitirá excluir um produto que não será mais vendido pela loja.
- **RF09** – O sistema permitirá registrar a forma de Pagamento (cartão, cheque ou dinheiro) do cliente.
- **RF10** – O sistema mostrará o valor do troco após a venda.
- **RF11** – O sistema permitirá registrar a data e a hora em que cada produto da farmácia foi vendido.

1.0.2 Requisitos Não Funcionais

- **RNF01** – As informações do sistema serão armazenadas em um banco de dados.
- **RNF02** – O sistema deverá rodar em qualquer sistema operacional.
- **RNF03** – O sistema não deverá demorar mais de 10 segundos para responder.
- **RNF04** – O sistema deverá ter uma interface amigável, a fim de ser de fácil utilização;
- **RNF05** – O sistema deverá ser fortemente orientado a baixo acoplamento e alta coesão;

1.0.3 Regras de Negócio

- **RN01** – O cliente poderá realizar o pagamento por meio de cartão de crédito, cheque ou dinheiro.
- **RN02** – Os clientes que comprarem uma determinada quantidade de produtos terão descontos.
- **RN03** – Os clientes só podem comprar no máximo em 10 parcelas.

2.0 PROJETO DE TESTES DA CLASSE VENDA: PARTIÇÃO POR EQUIVALÊNCIA

Caso de Teste	Entrada	Condições	RE	RO	Avaliação RO
Teste Válido					
1.0	Total comprado: 49,00	Total comprado <R$ 50,00; Sem desconto;	49,00	49,00	**Correto**
1.1	Total comprado 51,00	Total comprado >R$50,00 e <= 100,00; Desconto de 5%	48,45	48,45	**Correto**
1.2	Total comprado: 101,00	Total comprado >R$100,00; Desconto de 10%	90,90	90,90	**Correto**
Teste Inválido					
1.3	Total comprado: 100,00	Total comprado >R$100,00; Desconto de 10%	90,00	100,00	**Inválido** A função deveria ser implementada para total_comprado >= 100
Teste Inválido					
2.0	Itens comprado: Acarbose, ELOCON, SOTRET	A quantidade total de medicamentos comprados é 3	3	4	**Inválido** O contador foi iniciado de maneira inadequada

Na classe Venda foram inseridos dois erros: a inicialização errada de um contador, gerando um valor final de quantidade de produtos vendidos errado; e um operador booleano de “>” que deveria ser “>=”, provocando erro no cálculo dos descontos. Em ambos os casos o erro foi detectado.

2. IMPLEMENTAÇÃO DOS TESTES

```
package main.java.nomoredodoiweb;
import java.util.Arrays;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;
public class VendaTest{
	@Test
	public void testeSemDesconto(){
		Produto prod = new  Produto(0, 49, 1.00, "Beractanto");
		Venda venda = new
	Venda(Arrays.asList(prod),"15/06/2018","13:30");
		assertEquals((Double) 49.00, venda.getValorTotal()); //não  deve
		haver qualquer desconto
	}

	@Test
	public void testeCincoPorcentoDesconto(){
		Produto prod = new  Produto(0, 51, 1.00, "Beractanto");
		Venda venda = new
	Venda(Arrays.asList(prod),"15/06/2018","14:30");
		assertEquals((Double) 48.45, venda.getValorTotal()); //desconto
	de  5%
	}

	@Test
	public void testeDezPorcentoDesconto(){
		Produto prod = new Produto(0, 101, 1.00, "Beractanto");
		Venda venda = new
	Venda(Arrays.asList(prod),"15/06/2018","15:30");
		assertEquals((Double) 90.9, venda.getValorTotal());  //
	desconto  de  10%
	}

	@Test
	public void testeDezPorcentoDescontoLimite(){
		Produto prod = new Produto(0, 100, 1.00, "Beractanto");
		Venda venda = new
	Venda(Arrays.asList(prod),"15/06/2018","16:30");
		assertEquals((Double) 90.0, venda.getValorTotal());
	}

	@Test
	public void testeQuantidadeProdutos(){
		Venda venda  = new Venda(Arrays.asList(
		new Produto(0, 1, 1.00, "Acarbose"),
		new Produto(0, 1, 1.00, "ELOCON"),
		new Produto(0, 1, 1.00, "SOTRET")),"15/06/2018","17:30");
		assertEquals((Integer) 3, venda.getQuantidadeItens());
	}
}
```

3.0 IMPLEMENTAÇÃO DAS CLASSES UTILIZADAS

3.0.1 Classe Produto

```
package main.java.nomoredodoiweb;

public class Produto{
	private Integer codigo;
	private Integer quantidade;
	private Double preco;  //  preco  unitario
	private Double precoTotal;  //  preco  de  todos  os  produtos
	private String nome;

	public Produto(Integer codigo, Integer quantidade, Double preco, String
	nome){
		this.codigo = codigo;
		this.quantidade = quantidade;
		this.preco = preco;
		this.precoTotal  =  preco  *  quantidade;
		this.nome = nome;
	}
	public void vende(Integer quantidade){
		this.quantidade -= quantidade;
	}

	public double getPreco(){
		return preco;
	}

	public Double getPrecoTotal(){
		return  this.precoTotal;
	}

	public Integer getQuantidade(){
		return this.quantidade;
	}

	public boolean equals(Produto  p){
	return (p.codigo.equals(this.codigo) && p.preco.equals(this.preco));
	}
}
```

3.0.2 Classe Venda

```
package main.java.nomoredodoiweb;
import java.util.List;
import java.util.ArrayList;

public class Venda{
	private String data;
	private String hora;
	private List<Produto> produtos = new ArrayList<Produto>();

	public Venda(List<Produto> produtos, String data, String hora){
		this.produtos = produtos;
		this.data = data;
		this.hora = hora;
	}
	public String getData(){
		return data;
	}

	public String getHora(){
		return hora;
	}
	public  Integer  getQuantidadeItens(){
		Integer cont = 1; //ERRO AQUI, INICIALIZACAO  ERRADA
		for  (Produto prod : produtos){
			cont += prod.getQuantidade();
		}
		return cont;
	}
	public Double getValorTotal(){
		Double cont  =  0.0;
		for (Produto prod : produtos){
			cont += prod.getPrecoTotal();
		}
		if (cont >= 50.0 && cont < 100){ // desconto de 5%
			cont -= cont * 0.05;
		}else if (cont > 100){
		//ERRO  AQUI,  DEVE  SER  >= // desconto de 10%
			cont -= cont * 0.1;
		}
		return cont;
	}
}
```

3.0.3 Classe VendasRealizadas

```
package main.java.nomoredodoiweb;
import java.util.List;
import java.util.ArrayList;

public class VendasRealizadas{

	private static List<Venda> vendas = new ArrayList<Venda>();

	VendasRealizadas(){}

	public static void adicionaVenda(Venda venda){
		vendas.add(venda);
	}

	public static Integer totalVendasRealizadas(){
		return vendas.size();
	}

	public static Integer quantidadeItensVendidos()
	{
		Integer cont = 0;
		for (Venda venda : vendas){
			cont += venda.getQuantidadeItens();
		}
		return cont;
	}

	public static Double lucroObtido(){
		Double cont = 0.0;
		for (Venda venda : vendas){
			cont += venda.getValorTotal();
		}
		return cont;
	}

	public static String dataUltimaVenda(){
		String data = "";
		for (Venda venda : vendas){
		data = venda.getData();
		}
		return data;
	}
	public static String horaUltimaVenda(){
		String hora  =  "";
		for (Venda  venda  :  vendas){
			hora = venda.getHora();
		}
		return hora;
	}
}
```

3.0.4 Classe Principal

```
package main.java.nomoredodoiweb;
import java.text.DateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

public  class  Main{

      public static void main(String[]args){

            VendasRealizadas historico = new VendasRealizadas();
            Produto prod1 = new Produto(1,10,2.5,"Produto1");
            Produto prod2 = new Produto(2,20, 5.0, "Produto 2");
            Produto prod3 = new Produto(3,30,7.5,"Produto 3");
            List<Produto> produtos = new ArrayList<Produto>();

            produtos.add(prod1);
            produtos.add(prod2);
            produtos.add(prod3);

            List<Produto> produtos1 = new ArrayList<Produto>();
            produtos1.add(prod1);
            produtos1.add(prod3);

            Date d = new Date();
            String dataStr =
         java.text.DateFormat.getDateInstance(DateFormat.MEDIUM).format(d);
            String hora = "15:30";

            Venda venda = new Venda(produtos,dataStr,hora);
            Venda venda1 = new Venda(produtos1,"15/06/2018","14:30");

            historico.adicionaVenda(venda);
            historico.adicionaVenda(venda1);

            System.out.println(historico.totalVendasRealizadas());
            System.out.println(historico.lucroObtido());
            System.out.println(historico.quantidadeItensVendidos());
            System.out.println("Ultima venda:
            "+historico.dataUltimaVenda()+"
                                        "+historico.horaUltimaVenda());
      }
}
```