



Sistema de Carrinho de Compras em Java: Procedimento Simples e Detalhado

O sistema de carrinho de compras é uma aplicação fundamental em e-commerce, permitindo aos usuários adicionar, remover e gerenciar produtos antes de finalizar a compra.

Este procedimento apresenta a construção de um carrinho de compras em Java, utilizando classes para modelar produtos e o carrinho, além de um sistema simples de interação com o usuário.

Estrutura do Projeto

1. Requisitos e Preparação do Ambiente

- **Ferramentas Necessárias:**
 - Java JDK (versão 8 ou superior).
 - IDE como IntelliJ IDEA, Eclipse ou Visual Studio Code.
 - Gerenciador de dependências, como Maven ou Gradle, se necessário.

2. Estrutura do Pacote

A estrutura do pacote será a seguinte:

```
EducaCiência
├── Carrinho.java
├── Produto.java
├── Sistema.java
├── CarrinhoCompra.java
└── CarrinhoTest.java
```

3. Classes do Projeto

Classe Produto

A classe Produto representa os itens que serão adicionados ao carrinho. Esta classe deve incluir atributos como id, nome, preço e métodos para manipulação desses dados.



Exemplo de Código:

```
package EducaCiência;

public class Produto {
    private int id;
    private String nome;
    private double preco;
    private int quantidade;

    public Produto(int id, String nome, double preco, int quantidade) {
        this.id = id;
        this.nome = nome;
        this.preco = preco;
        this.quantidade = quantidade;
    }

    public int getId() {
        return id;
    }

    public String getNome() {
        return nome;
    }

    public double getPreco() {
        return preco;
    }

    public int getQuantidade() {
        return quantidade;
    }

    public void setQuantidade(int quantidade) {
        this.quantidade = quantidade;
    }

    @Override
    public String toString() {
        return "Produto{" +
            "id=" + id +
            ", nome=" + nome + "\n" +
            ", preco=" + preco +
            ", quantidade=" + quantidade +
            "}";
    }
}
```

Classe Carrinho

A classe Carrinho é responsável por armazenar os produtos e calcular o total da compra. Esta classe deve ter métodos para adicionar, remover e listar produtos.

Exemplo de Código:

```
package EducaCiência;
```



```
import java.util.ArrayList;
import java.util.List;

public class Carrinho {
    private List<Produto> produtos;

    public Carrinho() {
        this.produtos = new ArrayList<>();
    }

    public void adicionarProduto(Produto produto) {
        produtos.add(produto);
        System.out.println("Produto adicionado: " + produto.getNome());
    }

    public void removerProduto(int produtold) {
        produtos.removeIf(produto -> produto.getId() == produtold);
        System.out.println("Produto removido.");
    }

    public double calcularTotal() {
        return produtos.stream()
            .mapToDouble(p -> p.getPreco() * p.getQuantidade())
            .sum();
    }

    public void listarProdutos() {
        if (produtos.isEmpty()) {
            System.out.println("O carrinho está vazio.");
        } else {
            produtos.forEach(System.out::println);
        }
    }
}
```

4. Classe Principal Sistema

A classe Sistema será responsável pela interação com o usuário, permitindo que ele adicione e remova produtos do carrinho.

Exemplo de Código:

```
package EducaCiência;

import java.util.Scanner;

public class Sistema {
    private Carrinho carrinho;
    private Scanner scanner;

    public Sistema() {
        this.carrinho = new Carrinho();
        this.scanner = new Scanner(System.in);
    }

    public void iniciar() {
        int opcao;
```



```
do {
    System.out.println("1. Adicionar Produto");
    System.out.println("2. Remover Produto");
    System.out.println("3. Listar Produtos");
    System.out.println("4. Calcular Total");
    System.out.println("5. Sair");
    System.out.print("Escolha uma opção: ");
    opcao = scanner.nextInt();

    switch (opcao) {
        case 1:
            adicionarProduto();
            break;
        case 2:
            removerProduto();
            break;
        case 3:
            carrinho.listarProdutos();
            break;
        case 4:
            System.out.println("Total: R$ " + carrinho.calcularTotal());
            break;
        case 5:
            System.out.println("Saindo do sistema.");
            break;
        default:
            System.out.println("Opção inválida.");
    }
} while (opcao != 5);
}

private void adicionarProduto() {
    System.out.print("ID do produto: ");
    int id = scanner.nextInt();
    System.out.print("Nome do produto: ");
    String nome = scanner.next();
    System.out.print("Preço do produto: ");
    double preco = scanner.nextDouble();
    System.out.print("Quantidade: ");
    int quantidade = scanner.nextInt();

    Produto produto = new Produto(id, nome, preco, quantidade);
    carrinho.adicionarProduto(produto);
}

private void removerProduto() {
    System.out.print("ID do produto a remover: ");
    int id = scanner.nextInt();
    carrinho.removerProduto(id);
}
}
```

5. Classe CarrinhoCompra

A classe CarrinhoCompra é a entrada principal da aplicação, onde o sistema é inicializado.

Exemplo de Código:



```
package EducaCiência;
```

```
public class CarrinhoCompra {  
    public static void main(String[] args) {  
        Sistema sistema = new Sistema();  
        sistema.iniciar();  
    }  
}
```

6. Saída (Output)

Ao executar o sistema, a interação pode ser semelhante ao seguinte:

```
1. Adicionar Produto  
2. Remover Produto  
3. Listar Produtos  
4. Calcular Total  
5. Sair  
Escolha uma opção: 1  
ID do produto: 1  
Nome do produto: Produto A  
Preço do produto: 50.0  
Quantidade: 2  
Produto adicionado: Produto A
```

```
1. Adicionar Produto  
2. Remover Produto  
3. Listar Produtos  
4. Calcular Total  
5. Sair  
Escolha uma opção: 3  
Produto{id=1, nome='Produto A', preco=50.0, quantidade=2}
```

```
1. Adicionar Produto  
2. Remover Produto  
3. Listar Produtos  
4. Calcular Total  
5. Sair  
Escolha uma opção: 4  
Total: R$ 100.0
```

7. Testes Unitários

Os testes unitários são fundamentais para garantir que cada parte do sistema funcione conforme o esperado. Vamos criar uma classe `CarrinhoTest` utilizando o JUnit.

Exemplo de Código:

```
package EducaCiência;
```

```
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.Test;
```

```
import static org.junit.jupiter.api.Assertions.*;
```



```
public class CarrinhoTest {
    private Carrinho carrinho;

    @BeforeEach
    public void setUp() {
        carrinho = new Carrinho();
    }

    @Test
    public void testAdicionarProduto() {
        Produto produto = new Produto(1, "Produto A", 50.0, 2);
        carrinho.adicionarProduto(produto);
        assertEquals(100.0, carrinho.calcularTotal());
    }

    @Test
    public void testRemoverProduto() {
        Produto produto = new Produto(1, "Produto A", 50.0, 2);
        carrinho.adicionarProduto(produto);
        carrinho.removerProduto(1);
        assertEquals(0.0, carrinho.calcularTotal());
    }

    @Test
    public void testCalcularTotal() {
        carrinho.adicionarProduto(new Produto(1, "Produto A", 50.0, 2));
        carrinho.adicionarProduto(new Produto(2, "Produto B", 30.0, 1));
        assertEquals(130.0, carrinho.calcularTotal());
    }

    @Test
    public void testListarProdutosVazio() {
        // Aqui podemos verificar se o carrinho está vazio
        // Uma abordagem para testar é capturar a saída e verificar o texto impresso
    }
}
```

8. Execução e Testes

Para executar o sistema:

1. Compile todas as classes.
2. Execute a classe CarrinhoCompra para iniciar a aplicação.
3. Para rodar os testes, você pode utilizar o framework JUnit na sua IDE, ou executar os testes via linha de comando, se configurado corretamente.

Neste procedimento, abordamos o desenvolvimento de um sistema de carrinho de compras em Java.

O sistema permite ao usuário adicionar, remover produtos e calcular o total da compra de maneira simples e eficiente.



Além disso, implementamos testes unitários para garantir a funcionalidade das classes criadas.

Essa estrutura pode ser expandida para incluir funcionalidades adicionais, como persistência de dados e integração com um sistema de pagamento.

Desenvolvendo um Sistema de Carrinho de Compras em Java: Procedimento Detalhado

O sistema de carrinho de compras é uma aplicação fundamental em e-commerce, permitindo aos usuários adicionar, remover e gerenciar produtos antes de finalizar a compra.

Este procedimento apresenta a construção de um carrinho de compras em Java, utilizando classes para modelar produtos e o carrinho, além de um sistema simples de interação com o usuário.

Estrutura do Projeto

1. Requisitos e Preparação do Ambiente

- **Ferramentas Necessárias:**
 - Java JDK (versão 8 ou superior).
 - IDE como IntelliJ IDEA, Eclipse ou Visual Studio Code.
 - Servidor de Aplicação GlassFish.
 - Banco de Dados H2.
 - Dependências para JSP (JavaServer Pages) e JDBC (Java Database Connectivity).

2. Estrutura do Pacote

A estrutura do pacote será a seguinte:

```
EducaCiência
├── model
│   ├── Carrinho.java
│   └── Produto.java
├── controller
│   └── CarrinhoController.java
├── view
│   └── carrinho.jsp
├── CarrinhoCompra.java
└── CarrinhoTest.java
```



3. Classes do Projeto

Classe Produto

A classe Produto representa os itens que serão adicionados ao carrinho. Esta classe deve incluir atributos como id, nome, preco e métodos para manipulação desses dados.

Exemplo de Código:

```
package EducaCiência.model;

public class Produto {
    private int id;
    private String nome;
    private double preco;
    private int quantidade;

    public Produto(int id, String nome, double preco, int quantidade) {
        this.id = id;
        this.nome = nome;
        this.preco = preco;
        this.quantidade = quantidade;
    }

    public int getId() {
        return id;
    }

    public String getNome() {
        return nome;
    }

    public double getPreco() {
        return preco;
    }

    public int getQuantidade() {
        return quantidade;
    }

    public void setQuantidade(int quantidade) {
        this.quantidade = quantidade;
    }

    @Override
    public String toString() {
        return "Produto{" +
            "id=" + id +
            ", nome=" + nome + " " +
            ", preco=" + preco +
            ", quantidade=" + quantidade +
            "}";
    }
}
```




Classe Carrinho

A classe Carrinho é responsável por armazenar os produtos e calcular o total da compra. Esta classe deve ter métodos para adicionar, remover e listar produtos.

Exemplo de Código:

```
package EducaCiência.model;

import java.util.ArrayList;
import java.util.List;

public class Carrinho {
    private List<Produto> produtos;

    public Carrinho() {
        this.produtos = new ArrayList<>();
    }

    public void adicionarProduto(Produto produto) {
        produtos.add(produto);
    }

    public void removerProduto(int produtold) {
        produtos.removeIf(produto -> produto.getId() == produtold);
    }

    public double calcularTotal() {
        return produtos.stream()
            .mapToDouble(p -> p.getPreco() * p.getQuantidade())
            .sum();
    }

    public List<Produto> listarProdutos() {
        return produtos;
    }
}
```

Classe CarrinhoController

A classe CarrinhoController manipula as requisições da página JSP e interage com a lógica de negócios.

Exemplo de Código:

```
package EducaCiência.controller;

import EducaCiência.model.Carrinho;
import EducaCiência.model.Produto;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
```



```
@WebServlet("/carrinho")
public class CarrinhoController extends HttpServlet {
    private Carrinho carrinho = new Carrinho();

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        request.setAttribute("produtos", carrinho.listarProdutos());
        request.getRequestDispatcher("/view/carrinho.jsp").forward(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
        String acao = request.getParameter("acao");
        if ("adicionar".equals(acao)) {
            int id = Integer.parseInt(request.getParameter("id"));
            String nome = request.getParameter("nome");
            double preco = Double.parseDouble(request.getParameter("preco"));
            int quantidade = Integer.parseInt(request.getParameter("quantidade"));

            Produto produto = new Produto(id, nome, preco, quantidade);
            carrinho.adicionarProduto(produto);
        } else if ("remover".equals(acao)) {
            int id = Integer.parseInt(request.getParameter("id"));
            carrinho.removerProduto(id);
        }
        response.sendRedirect("carrinho");
    }
}
```

Página JSP carrinho.jsp

A página JSP apresenta a interface do carrinho de compras ao usuário.

Exemplo de Código:

```
jsp
<% @ page contentType="text/html; charset=UTF-8" language="java" %>
<% @ page import="EducaCiência.model.Produto" %>
<html>
<head>
    <title>Carrinho de Compras</title>
</head>
<body>
<h2>Carrinho de Compras</h2>
<form method="post" action="carrinho">
    <input type="hidden" name="acao" value="adicionar">
    ID: <input type="text" name="id"><br>
    Nome: <input type="text" name="nome"><br>
    Preço: <input type="text" name="preco"><br>
    Quantidade: <input type="text" name="quantidade"><br>
    <input type="submit" value="Adicionar Produto">
</form>
<hr>
<h3>Produtos no Carrinho</h3>
<ul>
    <c:forEach var="produto" items="${produtos}">
        <li>${produto.nome} - R$ ${produto.preco} x ${produto.quantidade}
            <form method="post" action="carrinho" style="display:inline;">
```



```
<input type="hidden" name="acao" value="remover">
<input type="hidden" name="id" value="${produto.id}">
<input type="submit" value="Remover">
</form>
</li>
</c:forEach>
</ul>
</body>
</html>
```

4. Configurando o Banco de Dados H2

1. **Adicionar Dependências:** Adicione a dependência do H2 no seu pom.xml (para Maven) ou build.gradle (para Gradle):

```
xml
<dependency>
  <groupId>com.h2database</groupId>
  <artifactId>h2</artifactId>
  <version>1.4.200</version>
</dependency>
```

2. **Criar Configuração de Banco de Dados:** Configure o H2 no seu servidor GlassFish. Adicione um novo recurso JDBC no console de administração do GlassFish, com a URL JDBC para o H2:

```
vbnet
jdbc:h2:mem:test;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE
```

5. Executando a Aplicação

1. **Implante o projeto no GlassFish.**
2. **Acesse a URL** correspondente ao CarrinhoController no navegador para interagir com o carrinho de compras.

6. Saída (Output)

Ao acessar a página JSP, a interação pode ser semelhante ao seguinte:

Carrinho de Compras
ID: [Campo para inserir ID]
Nome: [Campo para inserir Nome]
Preço: [Campo para inserir Preço]
Quantidade: [Campo para inserir Quantidade]
[Adicionar Produto]

Produtos no Carrinho
- Produto A - R\$ 50.0 x 2 [Remover]



7. Testes Unitários

Os testes unitários são fundamentais para garantir que cada parte do sistema funcione conforme o esperado. Vamos criar uma classe `CarrinhoTest` utilizando o JUnit.

Exemplo de Código

```
package EducaCiência;

import EducaCiência.model.Carrinho;
import EducaCiência.model.Produto;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

public class CarrinhoTest {
    private Carrinho carrinho;

    @BeforeEach
    public void setUp() {
        carrinho = new Carrinho();
    }

    @Test
    public void testAdicionarProduto() {
        Produto produto = new Produto(1, "Produto A", 50.0, 2);
        carrinho.adicionarProduto(produto);
        assertEquals(100.0, carrinho.calcularTotal());
    }

    @Test
    public void testRemoverProduto() {
        Produto produto = new Produto(1, "Produto A", 50.0, 2);
        carrinho.adicionarProduto(produto);
        carrinho.removerProduto(1);
        assertEquals(0.0, carrinho.calcularTotal());
    }
}
```

Este procedimento forneceu uma visão abrangente de como construir um sistema de carrinho de compras em Java utilizando o padrão MVC, com a integração do servidor GlassFish, banco de dados H2 e JSP.

A implementação de testes unitários também é fundamental para garantir a funcionalidade do sistema. Para um projeto mais robusto, considere adicionar funcionalidades como persistência de dados em um banco de dados relacional real, autenticação de usuário e melhorias na interface do usuário.

A utilização de frameworks como Spring MVC ou JSF pode ser uma evolução interessante para este projeto.

EducaCiência FastCode para a comunidade