Missão Prática | Nível 5 | Mundo 3

RPG0018 - Por que não paralelizar;

Simone Ramos de Jesus.

Matricula: 202208290965 – Turma: 2023.3

Polo Prado – Belo Horizonte – MG.

2023

RPG0018 - Por que não paralelizar;

Servidores e clientes baseados em Socket, com uso de Threads tanto no lado cliente quanto no lado servidor, acessando o banco de dados via JPA.

Objetivos da prática:

1 - Criar servidores Java com base em Sockets.

2 - Criar clientes síncronos para servidores com base em Sockets.

3 - Criar clientes assíncronos para servidores com base em Sockets.

4 - Utilizar Threads para implementação de processos paralelos.

5 - No final do exercício, o aluno terá criado um servidor Java baseado em Socket, com acesso ao banco de dados via JPA, além de utilizar os recursos nativos do Java para implementação de clientes síncronos e assíncronos. As Threads serão usadas tanto no servidor, para viabilizar múltiplos clientes paralelos, quanto no cliente, para implementar a resposta assíncrona.

Analise:

1 - Como funcionam as classes Socket e ServerSocket?

Socket: Fornece um endpoint para comunicação em rede.

ServerSocket: Espera por solicitações de clientes e cria sockets para comunicação.

2 - Qual a importância das portas para a conexão com servidores?

Portas identificam processos em um dispositivo. Permitem várias comunicações simultâneas em um único dispositivo.

3 - Para que servem as classes de entrada e saída?

Permitem a leitura (InputStream) e escrita (OutputStream) de dados em Java.

4 - ObjectInputStream e ObjectOutputStream, e por que os objetos transmitidos devem ser serializáveis?

Facilitam a serialização e desserialização de objetos Java.

Objetos devem ser serializáveis para serem convertidos em bytes para transmissão.

5 - Por que, mesmo utilizando as classes de entidades JPA no cliente, foi possível garantir o isolamento do acesso ao banco de dados?

Acesso ao banco é tratado pelo servidor, mantendo a lógica de negócios isolada do cliente.

6 - Como as Threads podem ser utilizadas para o tratamento assíncrono das respostas enviadas pelo servidor?

Threads permitem processamento simultâneo de várias operações.

Úteis para tratar respostas do servidor sem bloquear a execução.

7 - Para que serve o método invokeLater, da classe SwingUtilities?

Utilizado em GUIs Swing para executar tarefas na Thread de despacho de eventos.

Evita problemas de concorrência em interfaces gráficas.

8 - Como os objetos são enviados e recebidos pelo Socket Java?

Objetos são convertidos em bytes usando ObjectOutputStream para enviar e ObjectInputStream para receber.

9 - Compare a utilização de comportamento assíncrono ou síncrono nos clientes com Socket Java, ressaltando as características relacionadas ao bloqueio do processamento.

Assíncrono: Permite execução simultânea de operações sem bloquear.

Síncrono: Operações são realizadas em sequência, podendo bloquear o processamento.

Conclusão:

As classes Socket e ServerSocket em Java facilitam a comunicação em rede.

A utilização de portas é crucial para identificar processos em dispositivos e permitir a comunicação simultânea.

As classes de entrada e saída são essenciais para manipular dados. ObjectInputStream e ObjectOutputStream são úteis para serializar objetos, enquanto Threads possibilitam o tratamento assíncrono de respostas do servidor.

O método invokeLater da classe SwingUtilities evita problemas de concorrência em GUIs.

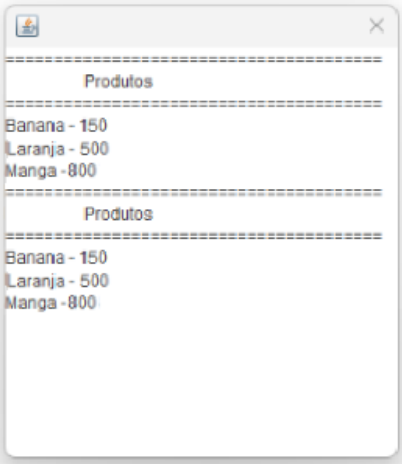
O envio e recebimento de objetos pelo Socket envolvem a conversão em bytes.

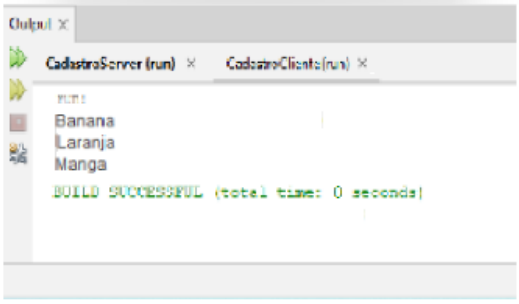
O uso de comportamento assíncrono no Socket permite execução simultânea, enquanto o síncrono pode resultar em bloqueio do processamento.

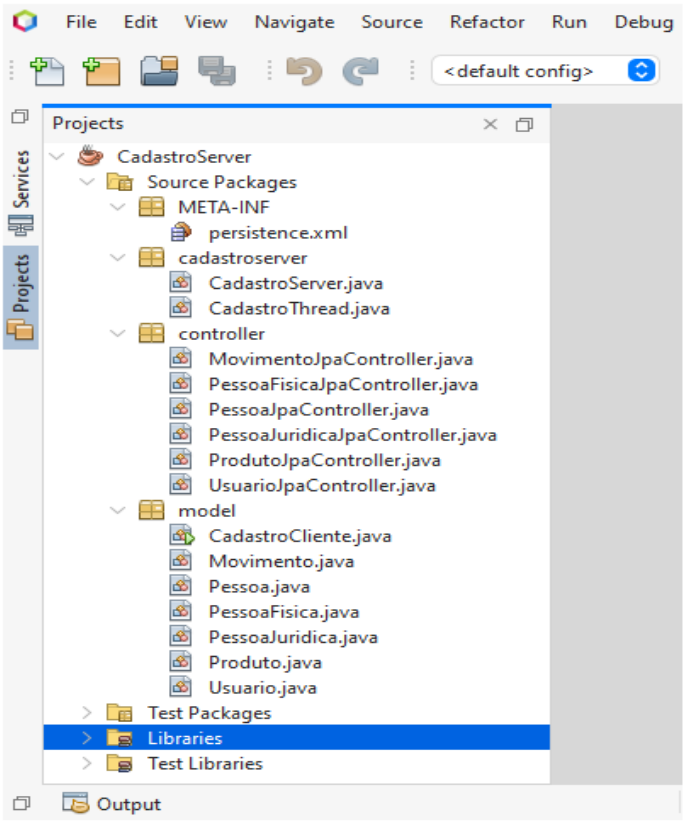
Em projetos com JPA, as classes de entidades garantem o isolamento do acesso ao banco de dados, mantendo a lógica de negócios no servidor.

Resultado da execução dos códigos:

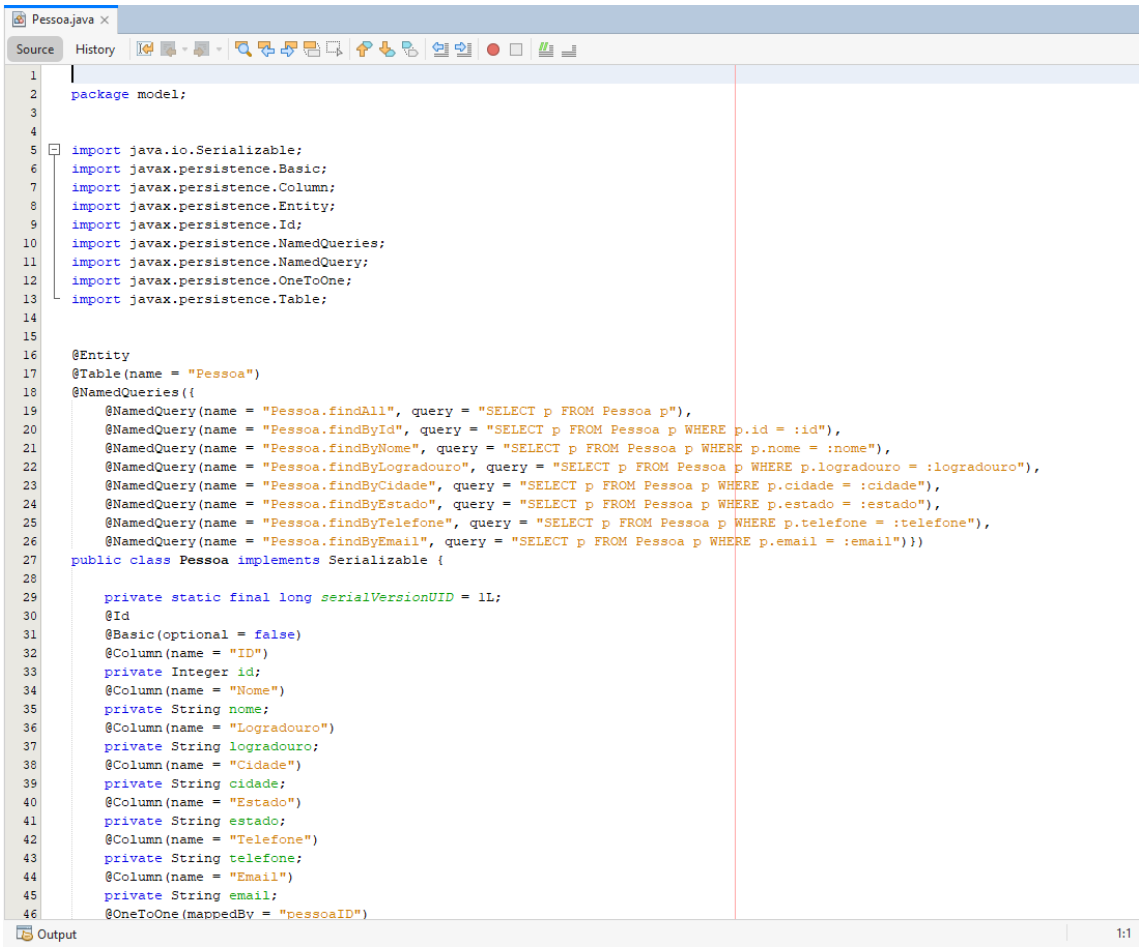
```
run:
================================================
                Menu de Opções
================================================
L - Listar
X - Finalizar
E - Entrada
S - Saída
Escolha uma opção: L
================================================
                Menu de Opções
================================================
L - Listar
X - Finalizar
E - Entrada
S - Saída
Escolha uma opção: E
Digite o ID da pessoa: 7
Digite o ID do produto: 1
Digite a quantidade: 50
Digite o valor unitário: 3
================================================
                Menu de Opções
================================================
L - Listar
X - Finalizar
E - Entrada
S - Saída
Escolha uma opção: L
================================================
                Menu de Opções
================================================
L - Listar
X - Finalizar
E - Entrada
S - Saída
Escolha uma opção:
```







Pessoa.java:


Pessoa.java
Source
History
package model;
import java.io.Serializable;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.OneToOne;
import javax.persistence.Table;
@Entity
@Table(name = "Pessoa")
@NamedQueries({
@NamedQuery(name = "Pessoa.findAll", query = "SELECT p FROM Pessoa p"),
@NamedQuery(name = "Pessoa.findById", query = "SELECT p FROM Pessoa p WHERE p.id = :id"),
@NamedQuery(name = "Pessoa.findByNome", query = "SELECT p FROM Pessoa p WHERE p.nome = :nome"),
@NamedQuery(name = "Pessoa.findByLogradouro", query = "SELECT p FROM Pessoa p WHERE p.logradouro = :logradouro"),
@NamedQuery(name = "Pessoa.findByCidade", query = "SELECT p FROM Pessoa p WHERE p.cidade = :cidade"),
@NamedQuery(name = "Pessoa.findByEstado", query = "SELECT p FROM Pessoa p WHERE p.estado = :estado"),
@NamedQuery(name = "Pessoa.findByTelefone", query = "SELECT p FROM Pessoa p WHERE p.telefone = :telefone"),
@NamedQuery(name = "Pessoa.findByEmail", query = "SELECT p FROM Pessoa p WHERE p.email = :email")})
public class Pessoa implements Serializable {
private static final long serialVersionUID = 1L;
@Id
@Basic(optional = false)
@Column(name = "ID")
private Integer id;
@Column(name = "Nome")
private String nome;
@Column(name = "Logradouro")
private String logradouro;
@Column(name = "Cidade")
private String cidade;
@Column(name = "Estado")
private String estado;
@Column(name = "Telefone")
private String telefone;
@Column(name = "Email")
private String email;
@OneToOne(mappedBy = "pessoaID")
Output
1:1

Pessoa.java

Source | History

```
44      @Column(name = "Email")
45      private String email;
46      @OneToOne(mappedBy = "pessoaID")
47      private PessoaJuridica pessoaJuridica;
48      @OneToOne(mappedBy = "pessoaID")
49      private PessoaFisica pessoaFisica;
50
51      public Pessoa() {}
52
53
54      public Pessoa(Integer id) {
55          this.id = id;
56      }
57      public Integer getId() {
58          return id;
59      }
60      public void setId(Integer id) {
61          this.id = id;
62      }
63      public String getNome() {
64          return nome;
65      }
66      public void setNome(String nome) {
67          this.nome = nome;
68      }
69      public String getLogradouro() {
70          return logradouro;
71      }
72      public void setLogradouro(String logradouro) {
73          this.logradouro = logradouro;
74      }
75      public String getCidade() {
76          return cidade;
77      }
78      public void setCidade(String cidade) {
79          this.cidade = cidade;
80      }
81      public String getEstado() {
82          return estado;
83      }
84      public void setEstado(String estado) {
85          this.estado = estado;
86      }
87      public String getTelefone() {
88          return telefone;
```

Pessoa.java

Source | History

```
96      public void setEmail(String email) {
97          this.email = email;
98      }
99      public PessoaJuridica getPessoaJuridica() {
100         return pessoaJuridica;
101     }
102     public void setPessoaJuridica(PessoaJuridica pessoaJuridica) {
103         this.pessoaJuridica = pessoaJuridica;
104     }
105     public PessoaFisica getPessoaFisica() {
106         return pessoaFisica;
107     }
108     public void setPessoaFisica(PessoaFisica pessoaFisica) {
109         this.pessoaFisica = pessoaFisica;
110     }
111     @Override
113     public int hashCode() {
114         int hash = 0;
115         hash += (id != null ? id.hashCode() : 0);
116         return hash;
117     }
119     @Override
120     public boolean equals(Object object) {
121         // TODO: Warning - this method won't work in the case the id fields are not set
122         if (!(object instanceof Pessoa)) {
123             return false;
125         }
126         Pessoa other = (Pessoa) object;
127         if ((this.id == null && other.id != null) || (this.id != null && !this.id.equals(obj:other.id))) {
128             return false;
129         }
131         return true;
132     }
133     @Override
134     public String toString() {
135         return "model.Pessoa[ id=" + id + " ]";
136     }
137     public Object getPessoaJuridicaCollection() {
        throw new UnsupportedOperationException(message:"Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templat
    }
}
```

PessoaFisica.java:

PessoaFisica.java

Source | History

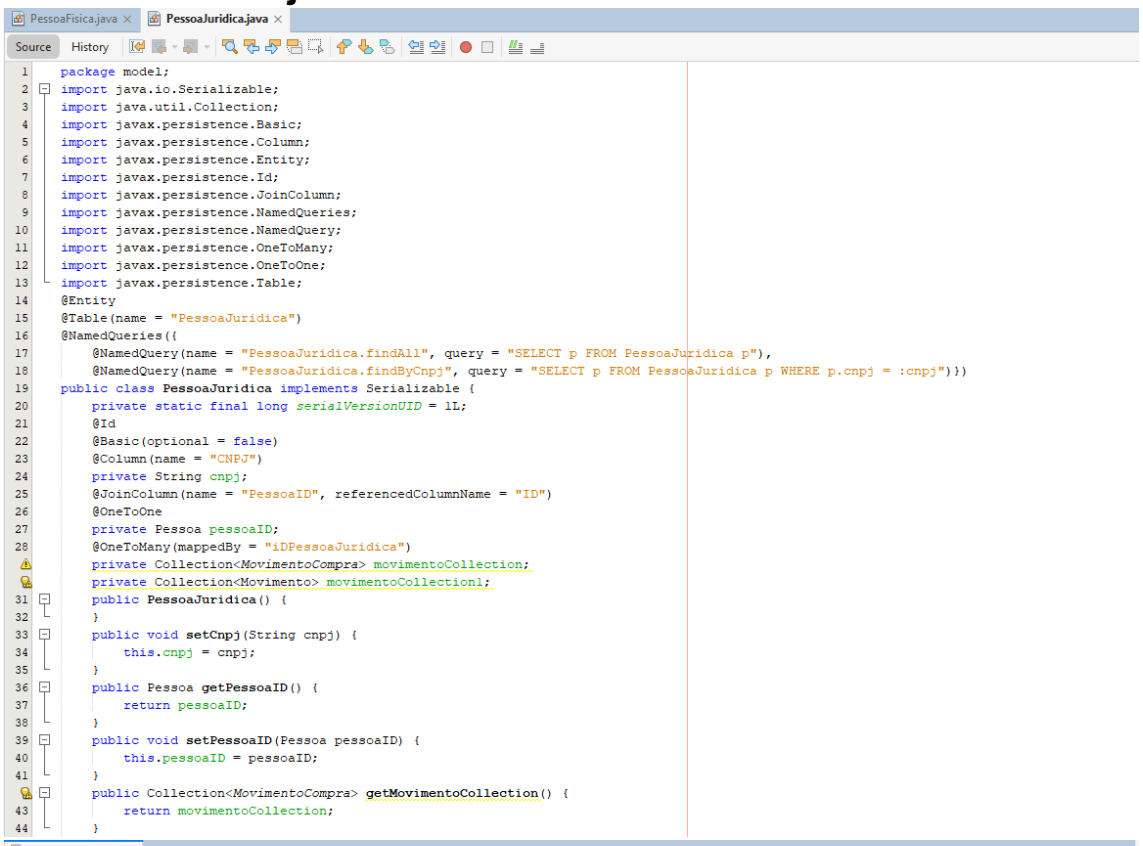
```java
package model;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.Collection;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.OneToMany;
import javax.persistence.OneToOne;
import javax.persistence.Table;
@Entity
@Table(name = "PessoaFisica")
@NamedQueries({
    @NamedQuery(name = "PessoaFisica.findAll", query = "SELECT p FROM PessoaFisica p"),
    @NamedQuery(name = "PessoaFisica.findByCpf", query = "SELECT p FROM PessoaFisica p WHERE p.cpf = :cpf")})
public class PessoaFisica implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @Column(name = "CPF")
    private String cpf;
    @OneToMany(mappedBy = "idPessoaFisica")
    private Collection<Movimento> movimentoCollection;
    @JoinColumn(name = "PessoaID", referencedColumnName = "ID")
    @OneToOne
    private Pessoa pessoaID;
    public PessoaFisica() {
    }
    public PessoaFisica(String cpf) {
        this.cpf = cpf;
    }
    public String getCpf() {
        return cpf;
    }
    public void setCpf(String cpf) {
        this.cpf = cpf;
    }
    public Collection<Movimento> getMovimentoCollection() {
        return movimentoCollection;
    }
```

PessoaFisica.java

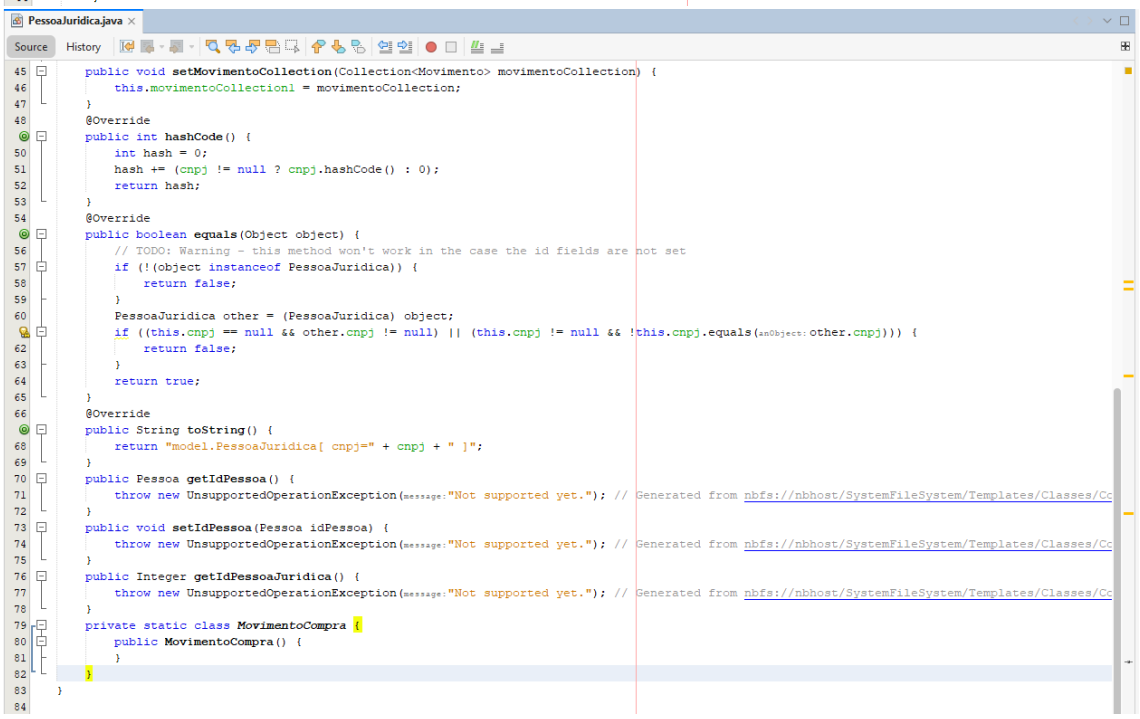
Source | History

```java
    public void setMovimento(Collection<Movimento> movimentoCollection) {
        this.movimentoCollection = movimentoCollection;
    }
    public void setPessoaID(Pessoa pessoaID) {
        this.pessoaID = pessoaID;
    }
    @Override
    public int hashCode() {
        int hash = 0;
        hash += (cpf != null ? cpf.hashCode() : 0);
        return hash;
    }
    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof PessoaFisica)) {
            return false;
        }
        PessoaFisica other = (PessoaFisica) object;
        if ((this.cpf == null && other.cpf != null) || (this.cpf != null && !this.cpf.equals(anObject: other.cpf))) {
            return false;
        }
        return true;
    }
    @Override
    public String toString() {
        return "model.PessoaFisica[ cpf=" + cpf + " ]";
    }
    public Pessoa getPessoaID() {
        throw new UnsupportedOperationException(message:"Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Cc
    }
    public void setMovimentoCollection(ArrayList<Movimento> arrayList) {
        throw new UnsupportedOperationException(message:"Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Cc
    }
    public void setMovimentoCollection(Collection<Movimento> attachedMovimentoCollection) {
        throw new UnsupportedOperationException(message:"Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Cc
    }
}
```

PessoaJuridica.java:

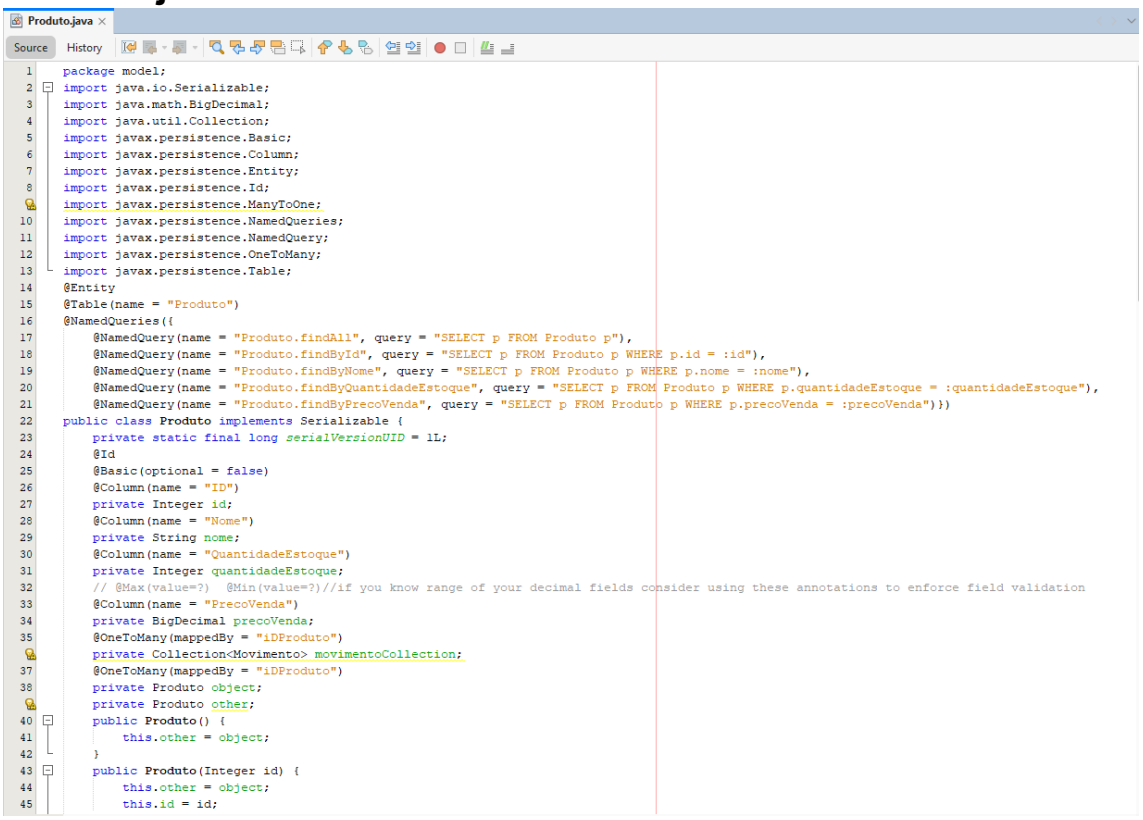


```java
package model;
import java.io.Serializable;
import java.util.Collection;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.OneToMany;
import javax.persistence.OneToOne;
import javax.persistence.Table;
@Entity
@Table(name = "PessoaJuridica")
@NamedQueries({
    @NamedQuery(name = "PessoaJuridica.findAll", query = "SELECT p FROM PessoaJuridica p"),
    @NamedQuery(name = "PessoaJuridica.findByCnpj", query = "SELECT p FROM PessoaJuridica p WHERE p.cnpj = :cnpj")})
public class PessoaJuridica implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @Column(name = "CNPJ")
    private String cnpj;
    @JoinColumn(name = "PessoaID", referencedColumnName = "ID")
    @OneToOne
    private Pessoa pessoaID;
    @OneToMany(mappedBy = "iDPessoaJuridica")
    private Collection<MovimentoCompra> movimentoCollection;
    private Collection<Movimento> movimentoCollection1;
    public PessoaJuridica() {
    }
    public void setCnpj(String cnpj) {
        this.cnpj = cnpj;
    }
    public Pessoa getPessoaID() {
        return pessoaID;
    }
    public void setPessoaID(Pessoa pessoaID) {
        this.pessoaID = pessoaID;
    }
    public Collection<MovimentoCompra> getMovimentoCollection() {
        return movimentoCollection;
    }
```



```java
    public void setMovimentoCollection(Collection<Movimento> movimentoCollection) {
        this.movimentoCollection1 = movimentoCollection;
    }
    @Override
    public int hashCode() {
        int hash = 0;
        hash += (cnpj != null ? cnpj.hashCode() : 0);
        return hash;
    }
    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof PessoaJuridica)) {
            return false;
        }
        PessoaJuridica other = (PessoaJuridica) object;
        if ((this.cnpj == null && other.cnpj != null) || (this.cnpj != null && !this.cnpj.equals(anObject: other.cnpj))) {
            return false;
        }
        return true;
    }
    @Override
    public String toString() {
        return "model.PessoaJuridica[ cnpj=" + cnpj + " ]";
    }
    public Pessoa getIdPessoa() {
        throw new UnsupportedOperationException(message:"Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Cc
    }
    public void setIdPessoa(Pessoa idPessoa) {
        throw new UnsupportedOperationException(message:"Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Cc
    }
    public Integer getIdPessoaJuridica() {
        throw new UnsupportedOperationException(message:"Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/Cc
    }
    private static class MovimentoCompra {
        public MovimentoCompra() {
        }
    }
}
```

Produto.java:


Produto.java

Source History

```
1  package model;
2  import java.io.Serializable;
3  import java.math.BigDecimal;
4  import java.util.Collection;
5  import javax.persistence.Basic;
6  import javax.persistence.Column;
7  import javax.persistence.Entity;
8  import javax.persistence.Id;
9  import javax.persistence.ManyToOne;
10 import javax.persistence.NamedQueries;
11 import javax.persistence.NamedQuery;
12 import javax.persistence.OneToMany;
13 import javax.persistence.Table;
14 @Entity
15 @Table(name = "Produto")
16 @NamedQueries({
17     @NamedQuery(name = "Produto.findAll", query = "SELECT p FROM Produto p"),
18     @NamedQuery(name = "Produto.findById", query = "SELECT p FROM Produto p WHERE p.id = :id"),
19     @NamedQuery(name = "Produto.findByNome", query = "SELECT p FROM Produto p WHERE p.nome = :nome"),
20     @NamedQuery(name = "Produto.findByQuantidadeEstoque", query = "SELECT p FROM Produto p WHERE p.quantidadeEstoque = :quantidadeEstoque"),
21     @NamedQuery(name = "Produto.findByPrecoVenda", query = "SELECT p FROM Produto p WHERE p.precoVenda = :precoVenda")})
22 public class Produto implements Serializable {
23     private static final long serialVersionUID = 1L;
24     @Id
25     @Basic(optional = false)
26     @Column(name = "ID")
27     private Integer id;
28     @Column(name = "Nome")
29     private String nome;
30     @Column(name = "QuantidadeEstoque")
31     private Integer quantidadeEstoque;
32     // @Max(value=?)  @Min(value=?)//if you know range of your decimal fields consider using these annotations to enforce field validation
33     @Column(name = "PrecoVenda")
34     private BigDecimal precoVenda;
35     @OneToMany(mappedBy = "idProduto")
36     private Collection<Movimento> movimentoCollection;
37     @OneToMany(mappedBy = "idProduto")
38     private Produto object;
39     private Produto other;
40     public Produto() {
41         this.other = object;
42     }
43     public Produto(Integer id) {
44         this.other = object;
45         this.id = id;
```

Produto.java

Source | History

```
47      public Integer getId() {
48          return id;
49      }
50      public void setId(Integer id) {
51          this.id = id;
52      }
53      public String getNome() {
54          return nome;
55      }
56      public void setNome(String nome) {
57          this.nome = nome;
58      }
59      public Integer getQuantidadeEstoque() {
60          return quantidadeEstoque;
61      }
62      public void setQuantidadeEstoque(Integer quantidadeEstoque) {
63          this.quantidadeEstoque = quantidadeEstoque;
64      }
65      public BigDecimal getPrecoVenda() {
66          return precoVenda;
67      }
68      public void setPrecoVenda(BigDecimal precoVenda) {
69          this.precoVenda = precoVenda;
70      }
71      public void setMovimentoCollection(Collection<Movimento> movimentoCollection) {
72          this.movimentoCollection = movimentoCollection;
73      }
74      @Override
75      public int hashCode() {
76          int hash = 0;
77          hash += (id != null ? id.hashCode() : 0);
78          return hash;
79      }
80      @Override
81      public boolean equals(Object object) {
82          // TODO: Warning - this method won't work in the case the id fields are not set
83          if (!(object instanceof Produto)) {
84              return false;}
85          return false;
86  }
87      public Object getMovimentacaoCollection() {
88          throw new UnsupportedOperationException(message:"Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/C
89      }
90      public Iterable<Movimento> getMovimentoCollection() {
91          throw new UnsupportedOperationException(message:"Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/C
```

Produto.java

Source | History

```
89      }
90      public Iterable<Movimento> getMovimentoCollection() {
91          throw new UnsupportedOperationException(message:"Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/C
92      }
93      public void setMovimentacaoCollection(Object movimentacaoCollectionNew) {
94          throw new UnsupportedOperationException(message:"Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/C
95      }
96      public Integer getIdProduto() {
97          throw new UnsupportedOperationException(message:"Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/C
98      }
99      public int getQuantidade() {
100         throw new UnsupportedOperationException(message:"Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/C
101     }
102     public void setQuantidade(int novaQuantidade) {
103         throw new UnsupportedOperationException(message:"Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/C
104     }
105     private static class MovimentoCompra {
106
107         public MovimentoCompra() {
108         }
109     }
110        class id {
111         public id() {
112         }
113         private static class equals {
114
115             public equals() {
116             }
117         }
118     }
119     @Override
120     public String toString() {
121         return "model.Produto[ id=" + id + " ]";
122     }
123 }
```

Usuário.java:

Usuario.java

Source | History

```java
package model;
import java.io.Serializable;
import java.util.Collection;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.OneToMany;
import javax.persistence.Table;
@Entity
@Table(name = "Usuario")
@NamedQueries({
    @NamedQuery(name = "Usuario.findAll", query = "SELECT u FROM Usuario u"),
    @NamedQuery(name = "Usuario.findById", query = "SELECT u FROM Usuario u WHERE u.id = :id"),
    @NamedQuery(name = "Usuario.findByLogin", query = "SELECT u FROM Usuario u WHERE u.login = :login"),
    @NamedQuery(name = "Usuario.findBySenha", query = "SELECT u FROM Usuario u WHERE u.senha = :senha")})
public class Usuario implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @Column(name = "ID")
    private Integer id;
    @Column(name = "Login")
    private String login;
    @Column(name = "Senha")
    private String senha;
    @OneToMany(mappedBy = "idUsuario")
    private Collection<Movimento> movimentoCollection;
    public Usuario() {
    }
    public Usuario(Integer id) {
        this.id = id;
    }
    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getLogin() {
        return login;
    }
    public void setLogin(String login) {
        this.login = login;
```

Usuario.java

Source | History

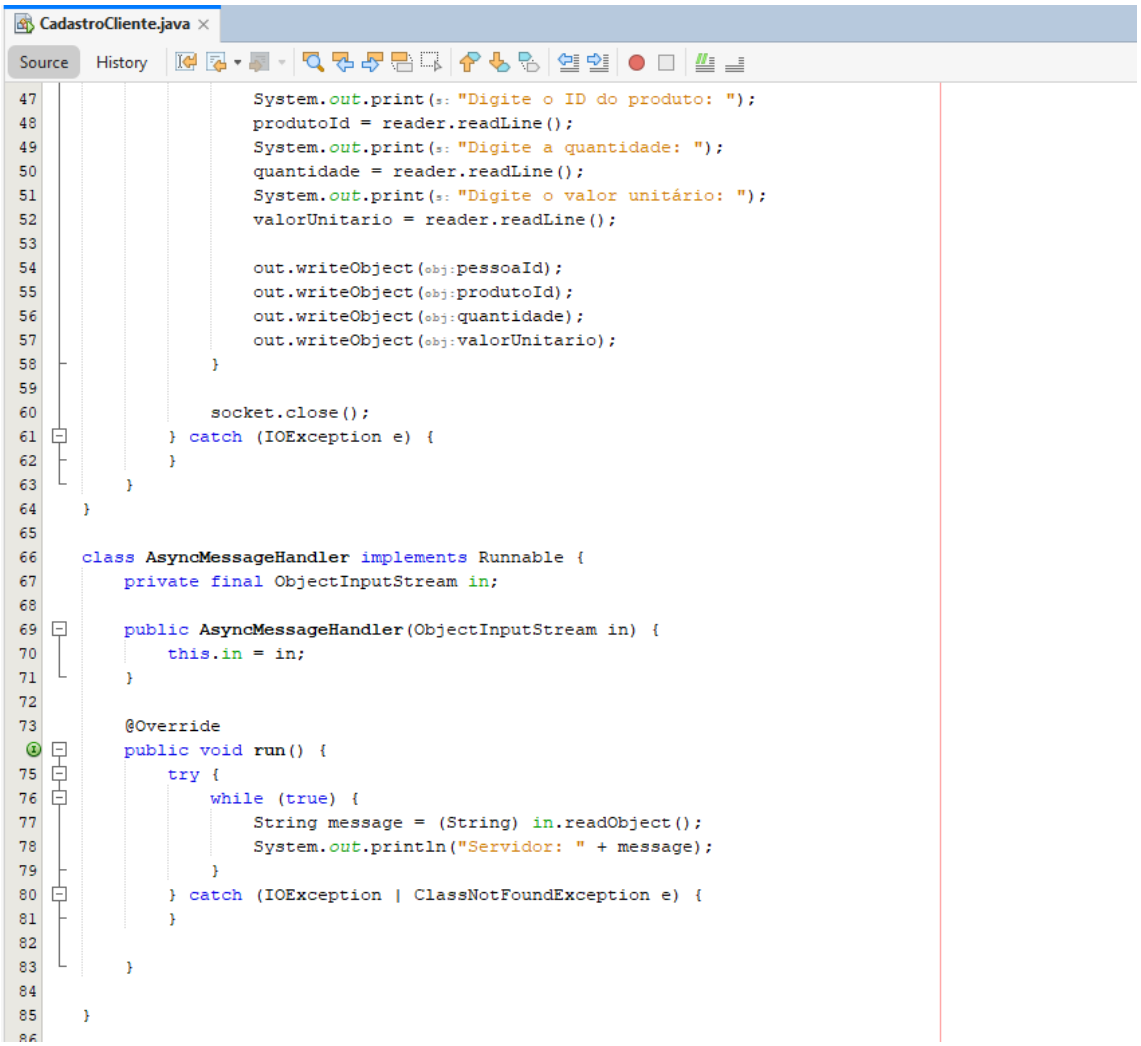
```java
    }
    public String getLogin() {
        return login;
    }
    public void setLogin(String login) {
        this.login = login;
    }
    public String getSenha() {
        return senha;
    }
    public void setSenha(String senha) {
        this.senha = senha;
    }
    public Collection<Movimento> getMovimentoCollection() {
        return movimentoCollection;
    }
    public void setMovimentoCollection(Collection<Movimento> movimentoCollection) {
        this.movimentoCollection = movimentoCollection;
    }
    @Override
    public int hashCode() {
        int hash = 0;
        hash += (id != null ? id.hashCode() : 0);
        return hash;
    }
    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof Usuario)) {
            return false;
        }
        Usuario other = (Usuario) object;
        if ((this.id == null && other.id != null) || (this.id != null && !this.id.equals(obj:other.id))) {
            return false;
        }
        return true;
    }
    @Override
    public String toString() {
        return "model.Usuario[ id=" + id + " ]";
    }
}
```

CadastroCliente.java:

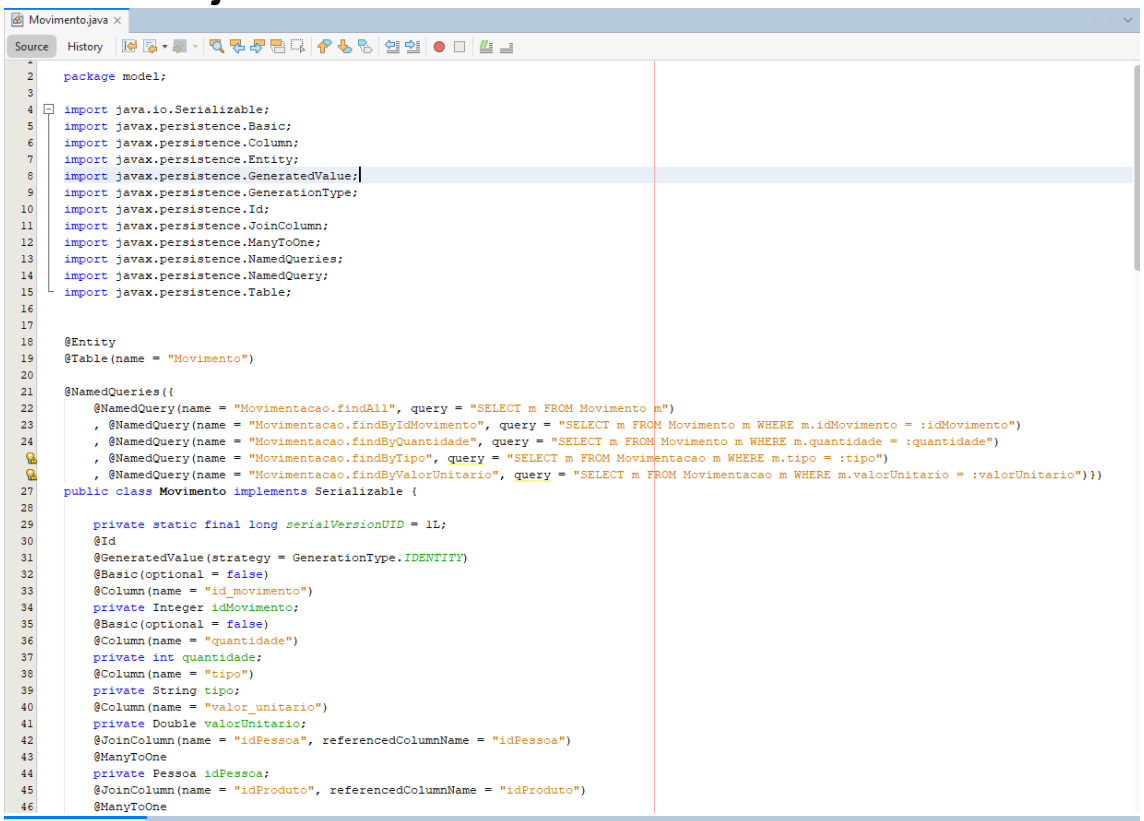
CadastroCliente.java

Source History

```
2
3   import java.io.*;
4   import java.net.*;
5   import java.util.Scanner;
6
7   public class CadastroCliente{
8
9       public static void main(String[] args) {
10          try {
11              Socket socket = new Socket(host: "localhost", port: 4321);
12              ObjectOutputStream out = new ObjectOutputStream(out:socket.getOutputStream());
13              ObjectInputStream in = new ObjectInputStream(in: socket.getInputStream());
14
15              BufferedReader reader = new BufferedReader(new InputStreamReader(in: System.in));
16
17              Scanner scanner = new Scanner(source: System.in);
18              String login = "op1";
19              String senha = "op1";
20
21              out.writeObject(obj:login);
22              out.writeObject(obj:senha);
23
24
25              Thread asyncThread = new Thread(new AsyncMessageHandler(in));
26              asyncThread.start();
27
28              String command;
29              while (true) {
30                  System.out.println(x: "Opções: L (Listar), E (Entrada), S (Saída), X (Finalizar)");
31                  System.out.print(s: "Digite um comando: ");
32                  command = scanner.nextLine();
33                  out.writeObject(obj:command);
34
35                  if (command.equalsIgnoreCase(anotherString: "X")) {
36                      break;
37                  }
38
39                  if (command.equalsIgnoreCase(anotherString: "L")) {
40                      continue;
41                  }
42
43                  String pessoaId, produtoId, quantidade, valorUnitario;
44
45                  System.out.print(s: "Digite o ID da pessoa: ");
46                  pessoaId = reader.readLine();
```


CadastroCliente.java
Source
History
System.out.print(s: "Digite o ID do produto: ");
produtoId = reader.readLine();
System.out.print(s: "Digite a quantidade: ");
quantidade = reader.readLine();
System.out.print(s: "Digite o valor unitário: ");
valorUnitario = reader.readLine();
out.writeObject(obj:pessoaId);
out.writeObject(obj:produtoId);
out.writeObject(obj:quantidade);
out.writeObject(obj:valorUnitario);
}
socket.close();
} catch (IOException e) {
}
}
}
class AsyncMessageHandler implements Runnable {
private final ObjectInputStream in;
public AsyncMessageHandler(ObjectInputStream in) {
this.in = in;
}
@Override
public void run() {
try {
while (true) {
String message = (String) in.readObject();
System.out.println("Servidor: " + message);
}
} catch (IOException | ClassNotFoundException e) {
}
}
}

Movimento.java:



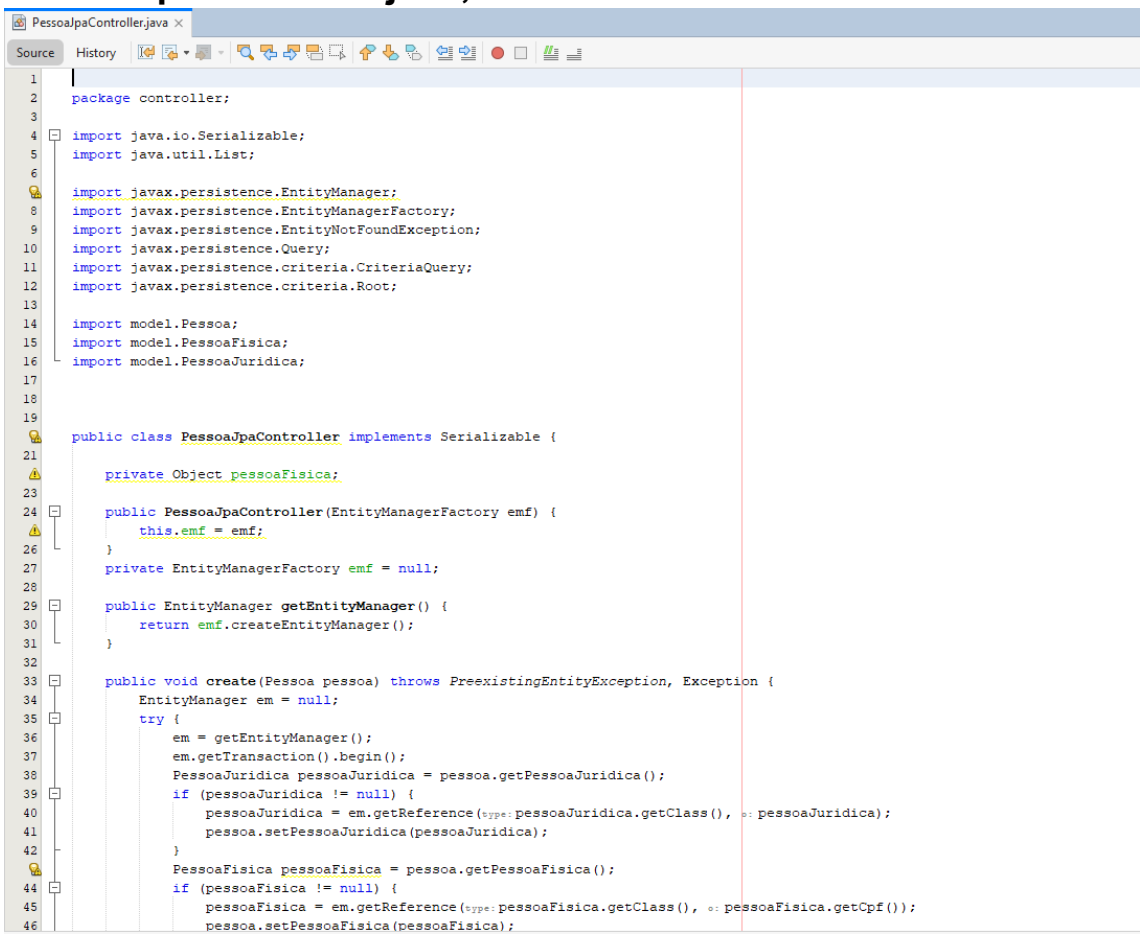
```java
package model;

import java.io.Serializable;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;


@Entity
@Table(name = "Movimento")

@NamedQueries({
    @NamedQuery(name = "Movimentacao.findAll", query = "SELECT m FROM Movimento m")
    , @NamedQuery(name = "Movimentacao.findByIdMovimento", query = "SELECT m FROM Movimento m WHERE m.idMovimento = :idMovimento")
    , @NamedQuery(name = "Movimentacao.findByQuantidade", query = "SELECT m FROM Movimento m WHERE m.quantidade = :quantidade")
    , @NamedQuery(name = "Movimentacao.findByTipo", query = "SELECT m FROM Movimentacao m WHERE m.tipo = :tipo")
    , @NamedQuery(name = "Movimentacao.findByValorUnitario", query = "SELECT m FROM Movimentacao m WHERE m.valorUnitario = :valorUnitario")})
public class Movimento implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "id_movimento")
    private Integer idMovimento;
    @Basic(optional = false)
    @Column(name = "quantidade")
    private int quantidade;
    @Column(name = "tipo")
    private String tipo;
    @Column(name = "valor_unitario")
    private Double valorUnitario;
    @JoinColumn(name = "idPessoa", referencedColumnName = "idPessoa")
    @ManyToOne
    private Pessoa idPessoa;
    @JoinColumn(name = "idProduto", referencedColumnName = "idProduto")
    @ManyToOne
    private Produto idProduto;
    @JoinColumn(name = "idUsuario", referencedColumnName = "idUsuario")
    @ManyToOne
    private Usuario idUsuario;

    public Movimento() {
    }

    public Movimento(Integer idMovimento) {
        this.idMovimento = idMovimento;
    }

    public Movimento(Integer idMovimento, int quantidade) {
        this.idMovimento = idMovimento;
        this.quantidade = quantidade;
    }

    public Integer getIdMovimento() {
        return idMovimento;
    }

    public void setIdMovimento(Integer idMovimento) {
        this.idMovimento = idMovimento;
    }

    public int getQuantidade() {
        return quantidade;
    }

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

    public String getTipo() {
        return tipo;
    }

    public void setTipo(String tipo) {
        // Aqui você pode definir a lógica para validar o tipo da movimentação
        // Certifique-se de validar o valor de 'tipo' para garantir que seja "E" ou "S"
        if ("E".equals(tipo) || "S".equals(tipo)) {
            this.tipo = tipo;
        } else {
            throw new IllegalArgumentException("O tipo de movimentação deve ser 'E' ou 'S'");
        }
    }
}
```

Movimento.java

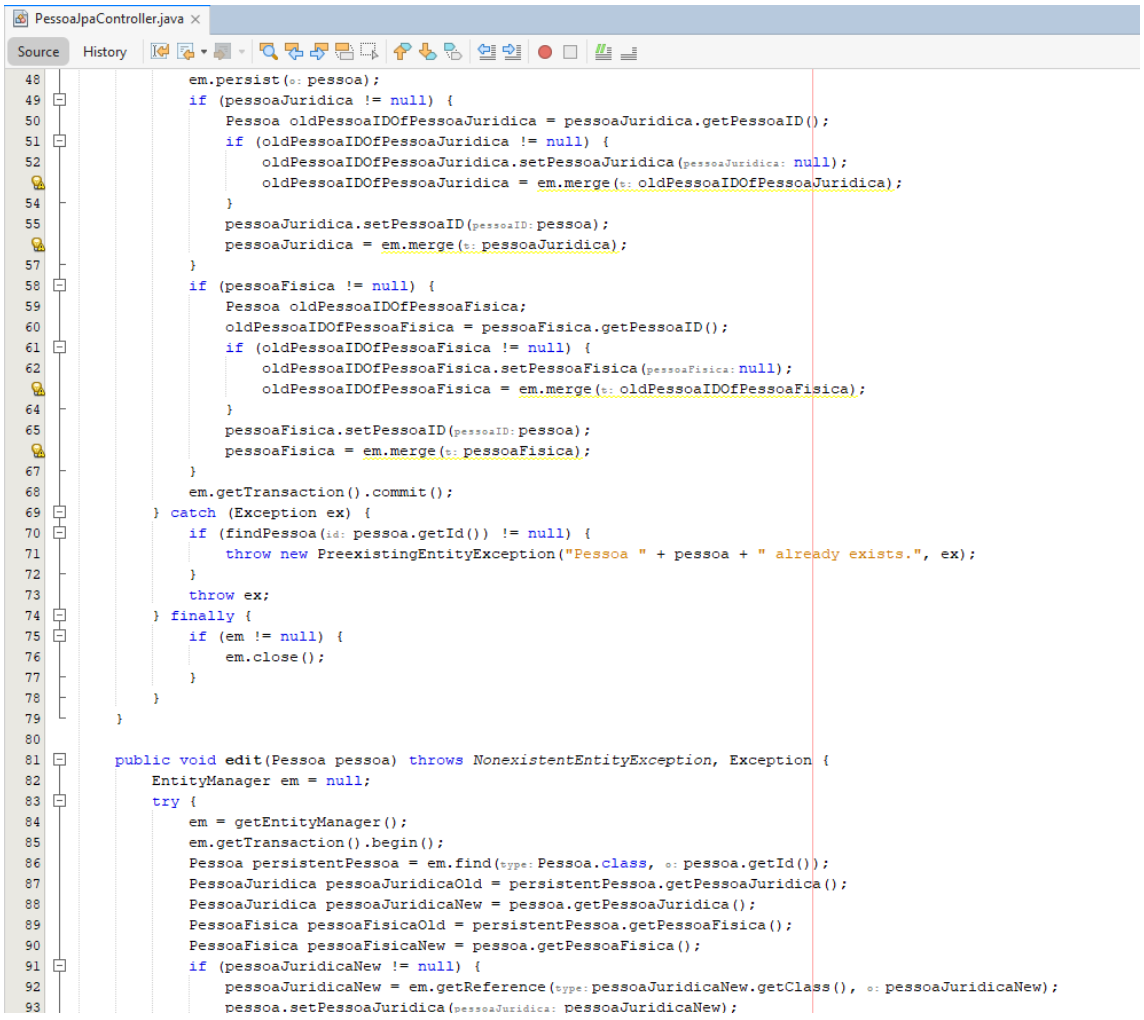
Source | History

```
94      public Double getValorUnitario() {
95          return valorUnitario;
96      }
97
98      public void setValorUnitario(Double valorUnitario) {
99          this.valorUnitario = valorUnitario;
100     }
101
102     public Pessoa getIdPessoa() {
103         return idPessoa;
104     }
105
106     public void setIdPessoa(Pessoa idPessoa) {
107         this.idPessoa = idPessoa;
108     }
109
110     public Produto getIdProduto() {
111         return idProduto;
112     }
113
114     public void setIdProduto(Produto idProduto) {
115         this.idProduto = idProduto;
116     }
117
118     public Usuario getIdUsuario() {
119         return idUsuario;
120     }
121
122     public void setIdUsuario(Usuario idUsuario) {
123         this.idUsuario = idUsuario;
124     }
125
126     @Override
        public int hashCode() {
128         int hash = 0;
129         hash += (idMovimento != null ? idMovimento.hashCode() : 0);
130         return hash;
131     }
132
133     @Override
        public boolean equals(Object object) {
135         if (!(object instanceof Movimento)) {
136             return false;
137         }
138         Movimento other = (Movimento) object;
137     }
138     Movimento other = (Movimento) object;
139     return !((this.idMovimento == null && other.idMovimento != null) || (this.idMovimento != null && !this.idMovimento.equals(obj:other.idMovi
140 }
141
142 @Override
    public String toString() {
144     return "cadastroserver.Movimento[ idMovimento=" + idMovimento + " ]";
145 }
146
147 public void setIDPessoaFisica(PessoaFisica pessoaFisica) {
148     throw new UnsupportedOperationException(message:"Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/C
149 }
150 }
151
```

PessoaJpaController.java;




```java

package controller;

import java.io.Serializable;
import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityNotFoundException;
import javax.persistence.Query;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;

import model.Pessoa;
import model.PessoaFisica;
import model.PessoaJuridica;



public class PessoaJpaController implements Serializable {

    private Object pessoaFisica;

    public PessoaJpaController(EntityManagerFactory emf) {
        this.emf = emf;
    }
    private EntityManagerFactory emf = null;

    public EntityManager getEntityManager() {
        return emf.createEntityManager();
    }

    public void create(Pessoa pessoa) throws PreexistingEntityException, Exception {
        EntityManager em = null;
        try {
            em = getEntityManager();
            em.getTransaction().begin();
            PessoaJuridica pessoaJuridica = pessoa.getPessoaJuridica();
            if (pessoaJuridica != null) {
                pessoaJuridica = em.getReference(type: pessoaJuridica.getClass(), o: pessoaJuridica);
                pessoa.setPessoaJuridica(pessoaJuridica);
            }
            PessoaFisica pessoaFisica = pessoa.getPessoaFisica();
            if (pessoaFisica != null) {
                pessoaFisica = em.getReference(type: pessoaFisica.getClass(), o: pessoaFisica.getCpf());
                pessoa.setPessoaFisica(pessoaFisica);
```


PessoaJpaController.java
Source
History
em.persist(o: pessoa);
if (pessoaJuridica != null) {
Pessoa oldPessoaIDOfPessoaJuridica = pessoaJuridica.getPessoaID();
if (oldPessoaIDOfPessoaJuridica != null) {
oldPessoaIDOfPessoaJuridica.setPessoaJuridica(pessoaJuridica: null);
oldPessoaIDOfPessoaJuridica = em.merge(o: oldPessoaIDOfPessoaJuridica);
}
pessoaJuridica.setPessoaID(pessoaID: pessoa);
pessoaJuridica = em.merge(o: pessoaJuridica);
}
if (pessoaFisica != null) {
Pessoa oldPessoaIDOfPessoaFisica;
oldPessoaIDOfPessoaFisica = pessoaFisica.getPessoaID();
if (oldPessoaIDOfPessoaFisica != null) {
oldPessoaIDOfPessoaFisica.setPessoaFisica(pessoaFisica: null);
oldPessoaIDOfPessoaFisica = em.merge(o: oldPessoaIDOfPessoaFisica);
}
pessoaFisica.setPessoaID(pessoaID: pessoa);
pessoaFisica = em.merge(o: pessoaFisica);
}
em.getTransaction().commit();
} catch (Exception ex) {
if (findPessoa(id: pessoa.getId()) != null) {
throw new PreexistingEntityException("Pessoa " + pessoa + " already exists.", ex);
}
throw ex;
} finally {
if (em != null) {
em.close();
}
}
}
public void edit(Pessoa pessoa) throws NonexistentEntityException, Exception {
EntityManager em = null;
try {
em = getEntityManager();
em.getTransaction().begin();
Pessoa persistentPessoa = em.find(type: Pessoa.class, o: pessoa.getId());
PessoaJuridica pessoaJuridicaOld = persistentPessoa.getPessoaJuridica();
PessoaJuridica pessoaJuridicaNew = pessoa.getPessoaJuridica();
PessoaFisica pessoaFisicaOld = persistentPessoa.getPessoaFisica();
PessoaFisica pessoaFisicaNew = pessoa.getPessoaFisica();
if (pessoaJuridicaNew != null) {
pessoaJuridicaNew = em.getReference(type: pessoaJuridicaNew.getClass(), o: pessoaJuridicaNew);
pessoa.setPessoaJuridica(pessoaJuridica: pessoaJuridicaNew);

PessoaJpaController.java

Source | History

```
94                }
95                if (pessoaFisicaNew != null) {
96                    pessoaFisicaNew = em.getReference(type: pessoaFisicaNew.getClass(), o: pessoaFisicaNew.getCpf());
97                    pessoa.setPessoaFisica(pessoaFisica: pessoaFisicaNew);
98                }
99                pessoa = em.merge(o: pessoa);
100               if (pessoaJuridicaOld != null && !pessoaJuridicaOld.equals(object: pessoaJuridicaNew)) {
101                   pessoaJuridicaOld.setPessoaID(pessoaID: null);
102                   pessoaJuridicaOld = em.merge(o: pessoaJuridicaOld);
103               }
104               if (pessoaJuridicaNew != null && !pessoaJuridicaNew.equals(object: pessoaJuridicaOld)) {
105                   Pessoa oldPessoaIDOfPessoaJuridica = pessoaJuridicaNew.getPessoaID();
106                   if (oldPessoaIDOfPessoaJuridica != null) {
107                       oldPessoaIDOfPessoaJuridica.setPessoaJuridica(pessoaJuridica: null);
108                       oldPessoaIDOfPessoaJuridica = em.merge(o: oldPessoaIDOfPessoaJuridica);
109                   }
110                   pessoaJuridicaNew.setPessoaID(pessoaID: pessoa);
111                   pessoaJuridicaNew = em.merge(o: pessoaJuridicaNew);
112               }
113               if (pessoaFisicaOld != null && !pessoaFisicaOld.equals(object: pessoaFisicaNew)) {
114                   pessoaFisicaOld.setPessoaID(pessoaID: null);
115                   pessoaFisicaOld = em.merge(o: pessoaFisicaOld);
116               }
117               if (pessoaFisicaNew != null && !pessoaFisicaNew.equals(object: pessoaFisicaOld)) {
118                   Pessoa oldPessoaIDOfPessoaFisica = pessoaFisicaNew.getPessoaID();
119                   if (oldPessoaIDOfPessoaFisica != null) {
120                       oldPessoaIDOfPessoaFisica.setPessoaFisica(pessoaFisica: null);
121                       oldPessoaIDOfPessoaFisica = em.merge(o: oldPessoaIDOfPessoaFisica);
122                   }
123                   pessoaFisicaNew.setPessoaID(pessoaID: pessoa);
124                   pessoaFisicaNew = em.merge(o: pessoaFisicaNew);
125               }
126               em.getTransaction().commit();
127           } catch (Exception ex) {
128               String msg = ex.getLocalizedMessage();
129               if (msg == null || msg.length() == 0) {
130                   Integer id = pessoa.getId();
131                   if (findPessoa(id) == null) {
132                       throw new NonexistentEntityException("The pessoa with id " + id + " no longer exists.");
133                   }
134               }
135               throw ex;
136           } finally {
137               if (em != null) {
138                   em.close();
139               }
```

PessoaJpaController.java

Source | History

```
143       public void destroy(Integer id) throws NonexistentEntityException {
144           EntityManager em = null;
145           try {
146               em = getEntityManager();
147               em.getTransaction().begin();
148               Pessoa pessoa;
149               try {
150                   pessoa = em.getReference(type: Pessoa.class, o: id);
151                   pessoa.getId();
152               } catch (EntityNotFoundException enfe) {
153                   throw new NonexistentEntityException("The pessoas with id " + id + " no longer exists.", enfe);
154               }
155               PessoaJuridica pessoaJuridica = pessoa.getPessoaJuridica();
156               if (pessoaJuridica != null) {
157                   pessoaJuridica.setPessoaID(pessoaID: null);
158                   pessoaJuridica = em.merge(o: pessoaJuridica);
159               }
160               PessoaFisica pessoasFisica = pessoa.getPessoaFisica();
161               if (pessoaFisica != null) {
162                   pessoaFisica = em.merge(o: pessoaFisica);
163               }
164               em.remove(o: pessoa);
165               em.getTransaction().commit();
166           } finally {
167               if (em != null) {
168                   em.close();
169               }
170           }
171       }
172
173       public List<Pessoa> findPessoaEntities() {
174           return findPessoaEntities(all:true, maxResults: -1, firstResult:-1);
175       }
176
177       public List<Pessoa> findPessoaEntities(int maxResults, int firstResult) {
178           return findPessoaEntities(all:false, maxResults, firstResult);
179       }
180
181       private List<Pessoa> findPessoaEntities(boolean all, int maxResults, int firstResult) {
182           EntityManager em = getEntityManager();
183           try {
184               CriteriaQuery cq = em.getCriteriaBuilder().createQuery();
185               cq.select(slctn: cq.from(type: Pessoa.class));
186               Query q = em.createQuery(cq);
187               if (!all) {
188                   q.setMaxResults(i: maxResults);
```

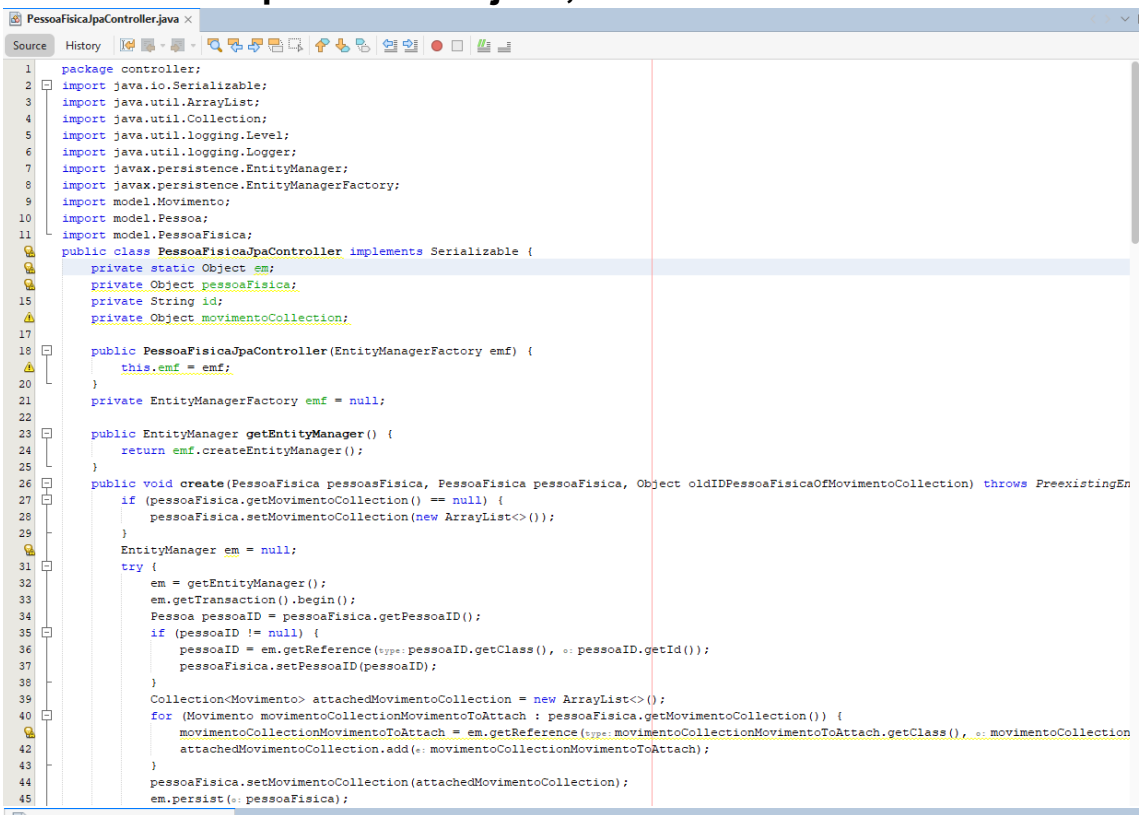
PessoaJpaController.java

Source | History

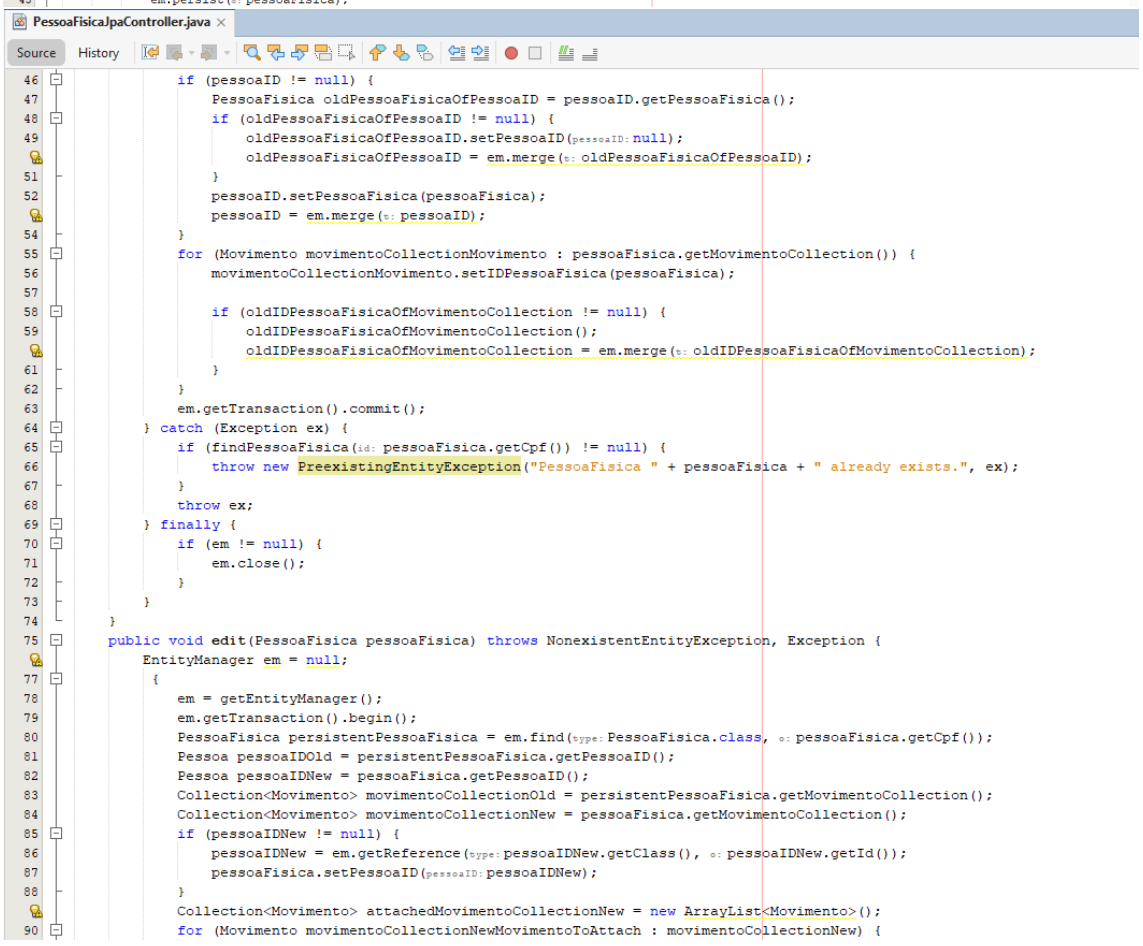
```
            return q.getResultList();
        } finally {
            em.close();
        }
    }

    public Pessoa findPessoa(Integer id) {
        EntityManager em = getEntityManager();
        try {
            return em.find(type: Pessoa.class, o: id);
        } finally {
            em.close();
        }
    }

    public int getPessoaCount() {
        EntityManager em = getEntityManager();
        try {
            CriteriaQuery cq = em.getCriteriaBuilder().createQuery();
            Root<Pessoa> rt = cq.from(type: Pessoa.class);
            cq.select(slctn: em.getCriteriaBuilder().count(exprsn: rt));
            Query q = em.createQuery(cq);
            return ((Long) q.getSingleResult()).intValue();
        } finally {
            em.close();
        }
    }

    private static class PreexistingEntityException extends Exception {

        public PreexistingEntityException() {
        }

        private PreexistingEntityException(String string, Exception ex) {
            throw new UnsupportedOperationException(message:"Not supported yet."); // Generated from nbfs:
        }
    }

    private static class NonexistentEntityException extends Exception {

        public NonexistentEntityException() {
        }

        private NonexistentEntityException(String string) {
            throw new UnsupportedOperationException(message:"Not supported yet."); // Generated from nbfs:
```


```
    private static class PreexistingEntityException extends Exception {

        public PreexistingEntityException() {
        }

        private PreexistingEntityException(String string, Exception ex) {
            throw new UnsupportedOperationException(message:"Not supported yet."); // Generated frc
        }
    }

    private static class NonexistentEntityException extends Exception {

        public NonexistentEntityException() {
        }

        private NonexistentEntityException(String string) {
            throw new UnsupportedOperationException(message:"Not supported yet."); // Generated frc
        }

        private NonexistentEntityException(String string, EntityNotFoundException enfe) {
            throw new UnsupportedOperationException(message:"Not supported yet."); // Generated frc
        }
```

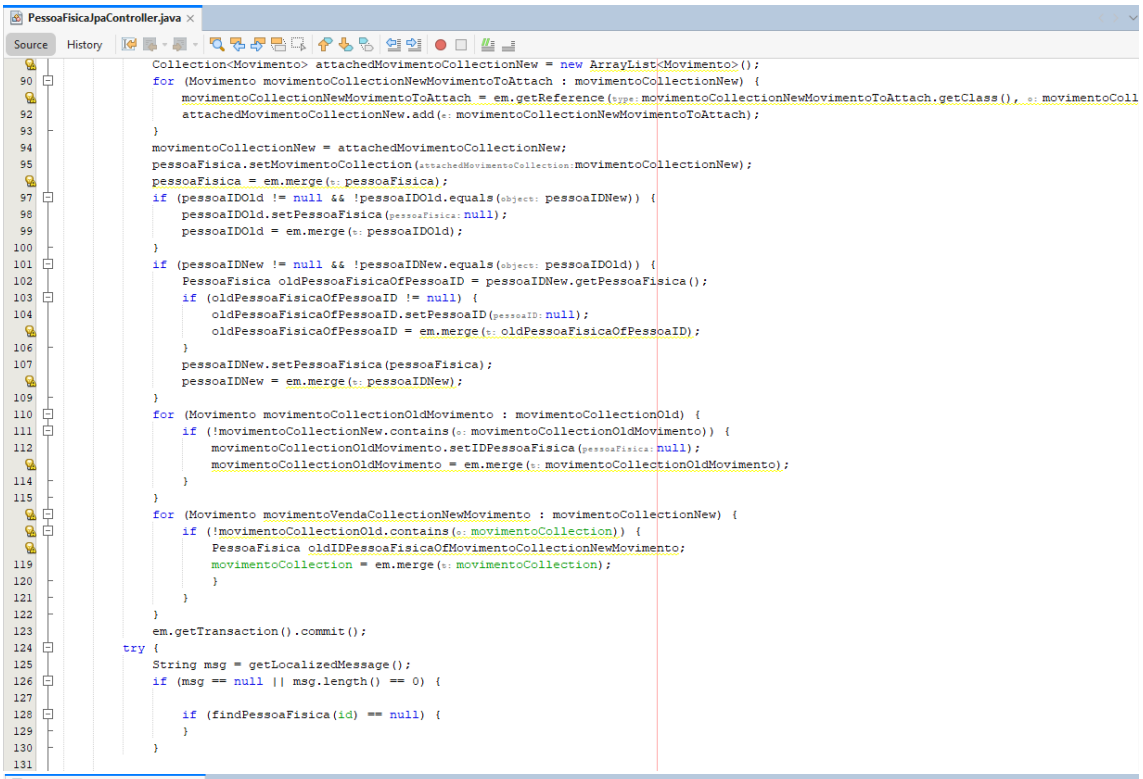
PessoaFisicaJpaController.java;



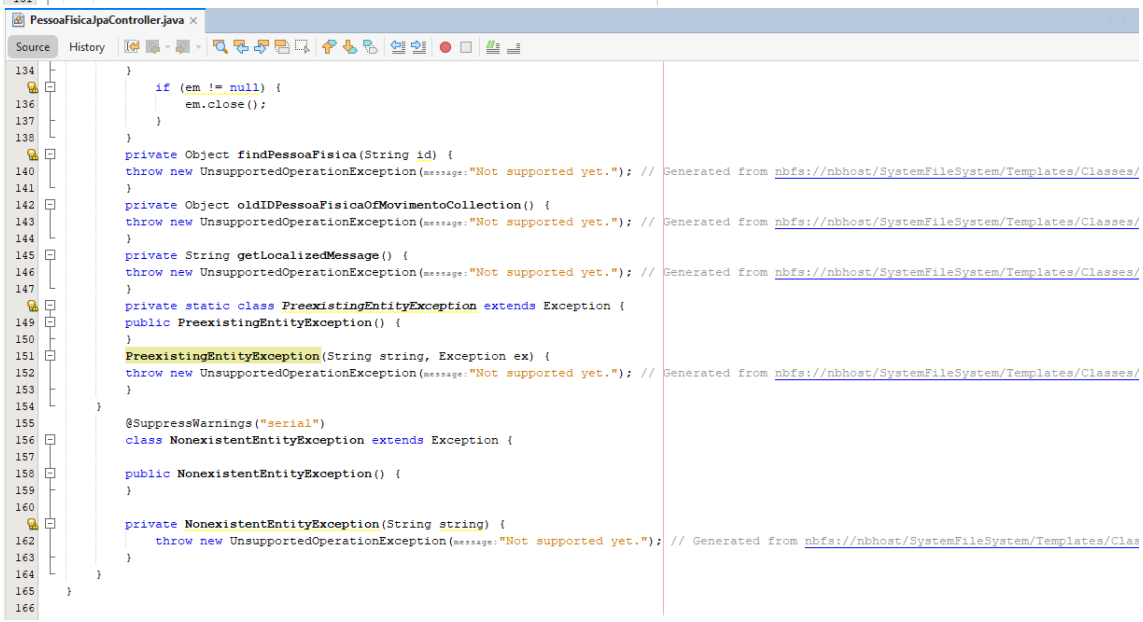
```java
package controller;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.Collection;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import model.Movimento;
import model.Pessoa;
import model.PessoaFisica;
public class PessoaFisicaJpaController implements Serializable {
    private static Object em;
    private Object pessoaFisica;
    private String id;
    private Object movimentoCollection;

    public PessoaFisicaJpaController(EntityManagerFactory emf) {
        this.emf = emf;
    }
    private EntityManagerFactory emf = null;

    public EntityManager getEntityManager() {
        return emf.createEntityManager();
    }
    public void create(PessoaFisica pessoasFisica, PessoaFisica pessoaFisica, Object oldIDPessoaFisicaOfMovimentoCollection) throws PreexistingEn
        if (pessoaFisica.getMovimentoCollection() == null) {
            pessoaFisica.setMovimentoCollection(new ArrayList<>());
        }
        EntityManager em = null;
        try {
            em = getEntityManager();
            em.getTransaction().begin();
            Pessoa pessoaID = pessoaFisica.getPessoaID();
            if (pessoaID != null) {
                pessoaID = em.getReference(type: pessoaID.getClass(), o: pessoaID.getId());
                pessoaFisica.setPessoaID(pessoaID);
            }
            Collection<Movimento> attachedMovimentoCollection = new ArrayList<>();
            for (Movimento movimentoCollectionMovimentoToAttach : pessoaFisica.getMovimentoCollection()) {
                movimentoCollectionMovimentoToAttach = em.getReference(type: movimentoCollectionMovimentoToAttach.getClass(), o: movimentoCollection
                attachedMovimentoCollection.add(o: movimentoCollectionMovimentoToAttach);
            }
            pessoaFisica.setMovimentoCollection(attachedMovimentoCollection);
            em.persist(o: pessoaFisica);
```



```java
            if (pessoaID != null) {
                PessoaFisica oldPessoaFisicaOfPessoaID = pessoaID.getPessoaFisica();
                if (oldPessoaFisicaOfPessoaID != null) {
                    oldPessoaFisicaOfPessoaID.setPessoaID(pessoaID: null);
                    oldPessoaFisicaOfPessoaID = em.merge(o: oldPessoaFisicaOfPessoaID);
                }
                pessoaID.setPessoaFisica(pessoaFisica);
                pessoaID = em.merge(o: pessoaID);
            }
            for (Movimento movimentoCollectionMovimento : pessoaFisica.getMovimentoCollection()) {
                movimentoCollectionMovimento.setIDPessoaFisica(pessoaFisica);

                if (oldIDPessoaFisicaOfMovimentoCollection != null) {
                    oldIDPessoaFisicaOfMovimentoCollection();
                    oldIDPessoaFisicaOfMovimentoCollection = em.merge(o: oldIDPessoaFisicaOfMovimentoCollection);
                }
            }
            em.getTransaction().commit();
        } catch (Exception ex) {
            if (findPessoaFisica(id: pessoaFisica.getCpf()) != null) {
                throw new PreexistingEntityException("PessoaFisica " + pessoaFisica + " already exists.", ex);
            }
            throw ex;
        } finally {
            if (em != null) {
                em.close();
            }
        }
    }
    public void edit(PessoaFisica pessoaFisica) throws NonexistentEntityException, Exception {
        EntityManager em = null;
        {
            em = getEntityManager();
            em.getTransaction().begin();
            PessoaFisica persistentPessoaFisica = em.find(type: PessoaFisica.class, o: pessoaFisica.getCpf());
            Pessoa pessoaIDOld = persistentPessoaFisica.getPessoaID();
            Pessoa pessoaIDNew = pessoaFisica.getPessoaID();
            Collection<Movimento> movimentoCollectionOld = persistentPessoaFisica.getMovimentoCollection();
            Collection<Movimento> movimentoCollectionNew = pessoaFisica.getMovimentoCollection();
            if (pessoaIDNew != null) {
                pessoaIDNew = em.getReference(type: pessoaIDNew.getClass(), o: pessoaIDNew.getId());
                pessoaFisica.setPessoaID(pessoaID: pessoaIDNew);
            }
            Collection<Movimento> attachedMovimentoCollectionNew = new ArrayList<Movimento>();
            for (Movimento movimentoCollectionNewMovimentoToAttach : movimentoCollectionNew) {
```



```
PessoaFisicaJpaController.java
Source  History
            Collection<Movimento> attachedMovimentoCollectionNew = new ArrayList<Movimento>();
            for (Movimento movimentoCollectionNewMovimentoToAttach : movimentoCollectionNew) {
                movimentoCollectionNewMovimentoToAttach = em.getReference(type: movimentoCollectionNewMovimentoToAttach.getClass(), o: movimentoColl
                attachedMovimentoCollectionNew.add(e: movimentoCollectionNewMovimentoToAttach);
            }
            movimentoCollectionNew = attachedMovimentoCollectionNew;
            pessoaFisica.setMovimentoCollection(attachedMovimentoCollection:movimentoCollectionNew);
            pessoaFisica = em.merge(o: pessoaFisica);
            if (pessoaIDOld != null && !pessoaIDOld.equals(object: pessoaIDNew)) {
                pessoaIDOld.setPessoaFisica(pessoaFisica: null);
                pessoaIDOld = em.merge(o: pessoaIDOld);
            }
            if (pessoaIDNew != null && !pessoaIDNew.equals(object: pessoaIDOld)) {
                PessoaFisica oldPessoaFisicaOfPessoaID = pessoaIDNew.getPessoaFisica();
                if (oldPessoaFisicaOfPessoaID != null) {
                    oldPessoaFisicaOfPessoaID.setPessoaID(pessoaID: null);
                    oldPessoaFisicaOfPessoaID = em.merge(o: oldPessoaFisicaOfPessoaID);
                }
                pessoaIDNew.setPessoaFisica(pessoaFisica);
                pessoaIDNew = em.merge(o: pessoaIDNew);
            }
            for (Movimento movimentoCollectionOldMovimento : movimentoCollectionOld) {
                if (!movimentoCollectionNew.contains(o: movimentoCollectionOldMovimento)) {
                    movimentoCollectionOldMovimento.setIDPessoaFisica(pessoaFisica: null);
                    movimentoCollectionOldMovimento = em.merge(o: movimentoCollectionOldMovimento);
                }
            }
            for (Movimento movimentoVendaCollectionNewMovimento : movimentoCollectionNew) {
                if (!movimentoCollectionOld.contains(o: movimentoCollection)) {
                    PessoaFisica oldIDPessoaFisicaOfMovimentoCollectionNewMovimento;
                    movimentoCollection = em.merge(o: movimentoCollection);
                    }
                }
            }
            em.getTransaction().commit();
        try {
            String msg = getLocalizedMessage();
            if (msg == null || msg.length() == 0) {

                if (findPessoaFisica(id) == null) {
                }
            }
```



```
PessoaFisicaJpaController.java
Source  History
        }
            if (em != null) {
                em.close();
            }
        }
        private Object findPessoaFisica(String id) {
        throw new UnsupportedOperationException(message:"Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/
        }
        private Object oldIDPessoaFisicaOfMovimentoCollection() {
        throw new UnsupportedOperationException(message:"Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/
        }
        private String getLocalizedMessage() {
        throw new UnsupportedOperationException(message:"Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/
        }
        private static class PreexistingEntityException extends Exception {
        public PreexistingEntityException() {
        }
        PreexistingEntityException(String string, Exception ex) {
        throw new UnsupportedOperationException(message:"Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Classes/
        }
    }
        @SuppressWarnings("serial")
        class NonexistentEntityException extends Exception {

        public NonexistentEntityException() {
        }

        private NonexistentEntityException(String string) {
            throw new UnsupportedOperationException(message:"Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Clas
        }
    }
}
```

PessoaJuridicaJpaController.java;

PessoaJuridicaJpaController.java

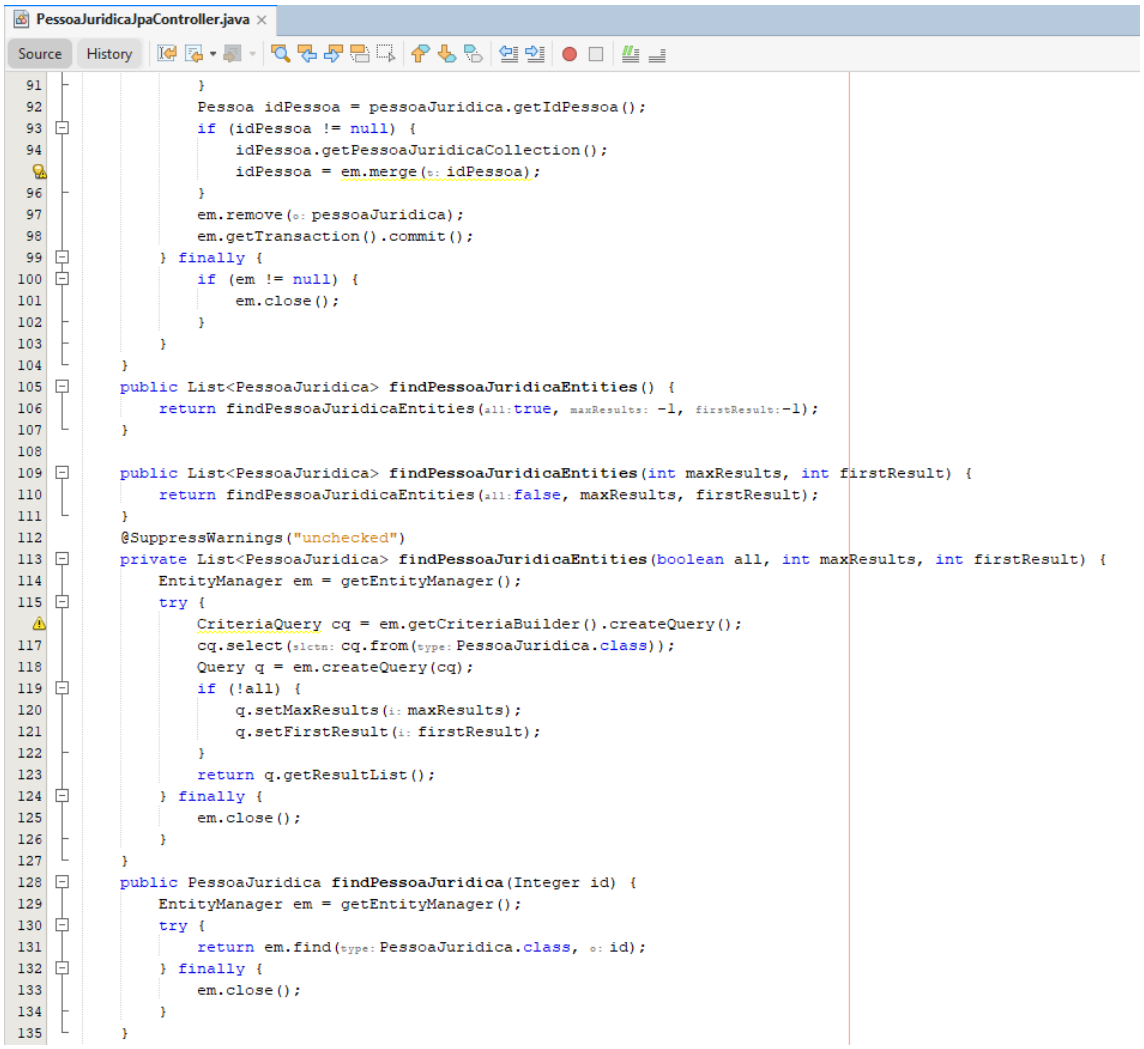
Source History

```
1  package controller;
2  import java.io.Serializable;
3  import java.util.List;
4  import javax.persistence.Query;
5  import javax.persistence.EntityManager;
6  import javax.persistence.EntityManagerFactory;
7  import javax.persistence.EntityNotFoundException;
8  import javax.persistence.criteria.CriteriaQuery;
9  import javax.persistence.criteria.Root;
10 import model.Pessoa;
11 import model.PessoaJuridica;
12 public class PessoaJuridicaJpaController implements Serializable {
13     private static final long serialVersionUID = 1L;
14     public PessoaJuridicaJpaController(EntityManagerFactory emf) {
15         this.emf = emf;
16     }
17     private EntityManagerFactory emf = null;
18     public EntityManager getEntityManager() {
19         return emf.createEntityManager();
20     }
21     public void create(PessoaJuridica pessoaJuridica) {
22         EntityManager em = null;
23         try {
24             em = getEntityManager();
25             em.getTransaction().begin();
26             Pessoa idPessoa = pessoaJuridica.getIdPessoa();
27             if (idPessoa != null) {
28                 idPessoa = em.getReference(type: idPessoa.getClass(), o: idPessoa.getId());
29                 pessoaJuridica.setIdPessoa(idPessoa);
30             }
31             em.persist(o: pessoaJuridica);
32             if (idPessoa != null) {
33                 idPessoa.getPessoaJuridicaCollection();
34                 idPessoa = em.merge(t: idPessoa);
35             }
36             em.getTransaction().commit();
37         } finally {
38             if (em != null) {
39                 em.close();
40             }
41         }
42     }
43     public void edit(PessoaJuridica pessoaJuridica) throws NonexistentEntityException, Exception {
44         EntityManager em = null;
45         try {
```

PessoaJuridicaJpaController.java

Source | History

```
46              em = getEntityManager();
47              em.getTransaction().begin();
48              PessoaJuridica persistentPessoaJuridica = em.find(type: PessoaJuridica.class, o: pessoaJuridica.getIdPessoaJuridica());
49              Pessoa idPessoaOld = persistentPessoaJuridica.getIdPessoa();
50              Pessoa idPessoaNew = pessoaJuridica.getIdPessoa();
51              if (idPessoaNew != null) {
52                  idPessoaNew = em.getReference(type: idPessoaNew.getClass(), o: idPessoaNew.getId());
53                  pessoaJuridica.setIdPessoa(idPessoa: idPessoaNew);
54              }
                pessoaJuridica = em.merge(o: pessoaJuridica);
56              if (idPessoaOld != null && !idPessoaOld.equals(object: idPessoaNew)) {
57                  idPessoaOld.getPessoaJuridicaCollection();
58                  idPessoaOld = em.merge(o: idPessoaOld);
59              }
60              if (idPessoaNew != null && !idPessoaNew.equals(object: idPessoaOld)) {
61                  idPessoaNew.getPessoaJuridicaCollection();
                    idPessoaNew = em.merge(o: idPessoaNew);
63              }
64              em.getTransaction().commit();
65          } catch (Exception ex) {
66              String msg = ex.getLocalizedMessage();
67              if (msg == null || msg.length() == 0) {
68                  Integer id = pessoaJuridica.getIdPessoaJuridica();
69                  if (findPessoaJuridica(id) == null) {
70                      throw new NonexistentEntityException("The pessoaJuridica with id " + id + " no longer exists.");
71                  }
72              }
73              throw ex;
74          } finally {
75              if (em != null) {
76                  em.close();
77              }
78          }
79      }
80      public void destroy(Integer id) throws NonexistentEntityException {
81          EntityManager em = null;
82          try {
83              em = getEntityManager();
84              em.getTransaction().begin();
85              PessoaJuridica pessoaJuridica;
86              try {
87                  pessoaJuridica = em.getReference(type: PessoaJuridica.class, o: id);
88                  pessoaJuridica.getIdPessoaJuridica();
89              } catch (EntityNotFoundException enfe) {
90                  throw new NonexistentEntityException("The pessoaJuridica with id " + id + " no longer exists.", enfe);
```


PessoaJuridicaJpaController.java
Source
History
91 }
92 Pessoa idPessoa = pessoaJuridica.getIdPessoa();
93 if (idPessoa != null) {
94 idPessoa.getPessoaJuridicaCollection();
idPessoa = em.merge(b: idPessoa);
96 }
97 em.remove(o: pessoaJuridica);
98 em.getTransaction().commit();
99 } finally {
100 if (em != null) {
101 em.close();
102 }
103 }
104 }
105 public List<PessoaJuridica> findPessoaJuridicaEntities() {
106 return findPessoaJuridicaEntities(all:true, maxResults: -1, firstResult:-1);
107 }
108
109 public List<PessoaJuridica> findPessoaJuridicaEntities(int maxResults, int firstResult) {
110 return findPessoaJuridicaEntities(all:false, maxResults, firstResult);
111 }
112 @SuppressWarnings("unchecked")
113 private List<PessoaJuridica> findPessoaJuridicaEntities(boolean all, int maxResults, int firstResult) {
114 EntityManager em = getEntityManager();
115 try {
CriteriaQuery cq = em.getCriteriaBuilder().createQuery();
117 cq.select(slctn: cq.from(type: PessoaJuridica.class));
118 Query q = em.createQuery(cq);
119 if (!all) {
120 q.setMaxResults(i: maxResults);
121 q.setFirstResult(i: firstResult);
122 }
123 return q.getResultList();
124 } finally {
125 em.close();
126 }
127 }
128 public PessoaJuridica findPessoaJuridica(Integer id) {
129 EntityManager em = getEntityManager();
130 try {
131 return em.find(type: PessoaJuridica.class, o: id);
132 } finally {
133 em.close();
134 }
135 }

PessoaJuridicaJpaController.java

```
130         try {
131             return em.find(type: PessoaJuridica.class, o: id);
132         } finally {
133             em.close();
134         }
135     }
136     @SuppressWarnings("unchecked")
137     public int getPessoaJuridicaCount() {
138         EntityManager em = getEntityManager();
139         try {
            CriteriaQuery cq = em.getCriteriaBuilder().createQuery();
141             @SuppressWarnings("unchecked")
142             Root<PessoaJuridica> rt = cq.from(type: PessoaJuridica.class);
143             cq.select(slctn: em.getCriteriaBuilder().count(exprsn: rt));
144             Query q = em.createQuery(cq);
145             return ((Number) q.getSingleResult()).intValue();
146         } finally {
147             em.close();
148         }
149     }
150     @SuppressWarnings("serial")
151         private static class NonexistentEntityException extends Exception {
            private NonexistentEntityException(String string, EntityNotFoundException enfe) {
153             throw new UnsupportedOperationException(message: "Not supported yet.");
154     }
            private NonexistentEntityException(String string) {
156             throw new UnsupportedOperationException(message: "Not supported yet."); // Generated
157         }
158 }
159
```

ProdutoJpaController.java;

ProdutoJpaController.java

```
1   package controller;
2   import controller.PessoaFisicaJpaController.NonexistentEntityException;
3   import java.io.Serializable;
4   import java.util.ArrayList;
5   import java.util.Collection;
6   import java.util.List;
7   import javax.persistence.EntityManager;
8   import javax.persistence.EntityManagerFactory;
9   import javax.persistence.EntityNotFoundException;
10  import javax.persistence.Query;
11  import javax.persistence.TypedQuery;
12  import javax.persistence.criteria.CriteriaQuery;
13  import javax.persistence.criteria.Root;
14  import model.Movimento;
15  import model.Produto;
16  public class ProdutoJpaController implements Serializable {
17      private EntityManager movimentoCollectionMovimento;
18      private Iterable<Movimento> movimentoCollectionNew;
19      private Object oldIdProdutoOfMovimentoCollectionMovimentacao;
20      private Iterable<Movimento> movimentoCollectionOld;
21      private Object movimentacaoCollectionOld;
22      public ProdutoJpaController(EntityManagerFactory emf) {
23          this.emf = emf;
24      }
25      private EntityManagerFactory emf = null;
26      public EntityManager getEntityManager() {
27          return emf.createEntityManager();
28      }
29      public void create(Produto produto) {
30          if (produto.getMovimentacaoCollection() == null) {
31              produto.setMovimentoCollection(new ArrayList<Movimento>());
32          }
33          EntityManager em = null;
34          try {
35              em = getEntityManager();
36              em.getTransaction().begin();
37              Collection<Movimento> attachedMovimentacaoCollection = new ArrayList<Movimento>();
38              for (Movimento movimentacaoCollectionMovimentacaoToAttach : produto.getMovimentoCollection()) {
39                  movimentacaoCollectionMovimentacaoToAttach = em.getReference(type: movimentacaoCollectionMovimentacaoToAttach.getClass(),
40                  attachedMovimentacaoCollection.add(e: movimentacaoCollectionMovimentacaoToAttach);
41              }
42              produto.setMovimentoCollection(movimentoCollection: attachedMovimentacaoCollection);
43              em.persist(o: produto);
44              for (Movimento movimentoCollectionMovimentacao : produto.getMovimentoCollection()) {
45                  Produto oldIdProdutoOfMovimentacaoCollectionMovimentacao = movimentoCollectionMovimentacao.getIdProduto();
```

ProdutoJpaController.java

Source | History

```
                movimentoCollectionMovimento = em.merge(movimentoCollectionMovimento);
                if (oldIdProdutoOfMovimentoCollectionMovimentacao != null) {
                    oldIdProdutoOfMovimentacaoCollectionMovimentacao.getMovimentacaoCollection();
                    oldIdProdutoOfMovimentacaoCollectionMovimentacao = em.merge(oldIdProdutoOfMovimentacaoCollectionMovimentacao);
                }
            }
            em.getTransaction().commit();
        } finally {
            if (em != null) {
                em.close();
            }
        }
    }
    public void edit(Produto produto) throws NonexistentEntityException, Exception {
        EntityManager em = null;
        {
            em = getEntityManager();
            em.getTransaction().begin();
            Produto persistentProduto;
            persistentProduto = em.find(type: Produto.class, o: produto.getIdProduto());
            Iterable<Movimento> movimentacaoCollectionOld = persistentProduto.getMovimentoCollection();
            Object movimentacaoCollectionNew = produto.getMovimentacaoCollection();
            Collection<Movimento> attachedMovimentacaoCollectionNew = new ArrayList<Movimento>();
            for (Movimento movimentacaoCollectionNewMovimentacaoToAttach : movimentoCollectionNew) {
                movimentacaoCollectionNewMovimentacaoToAttach = em.getReference(type:movimentacaoCollectionNewMovimentacaoToAttach.getClass(), o:
                attachedMovimentacaoCollectionNew.add(e: movimentacaoCollectionNewMovimentacaoToAttach);
            }
            movimentacaoCollectionNew = attachedMovimentacaoCollectionNew;
            produto.setMovimentacaoCollection(movimentacaoCollectionNew);
            produto = em.merge(t: produto);
            for (Movimento movimentacaoCollectionOldMovimentacao : movimentoCollectionOld) {

                movimentacaoCollectionOldMovimentacao.setIdProduto(idProduto: null);
                movimentacaoCollectionOldMovimentacao = em.merge(t: movimentacaoCollectionOldMovimentacao);
            }
            }
            for (Movimento movimentoCollectionNewMovimento : movimentoCollectionNew) {

                Produto oldIdProdutoOfMovimentacaoCollectionNewMovimentacao = movimentoCollectionNewMovimento.getIdProduto();
                movimentoCollectionNewMovimento.setIdProduto(idProduto: produto);
                movimentoCollectionNewMovimento = em.merge(t: movimentoCollectionNewMovimento);
                if (oldIdProdutoOfMovimentacaoCollectionNewMovimentacao != null && !oldIdProdutoOfMovimentacaoCollectionNewMovimentacao.equa
                    oldIdProdutoOfMovimentacaoCollectionNewMovimentacao.getMovimentoCollection();
                    oldIdProdutoOfMovimentacaoCollectionNewMovimentacao = em.merge(t: oldIdProdutoOfMovimentacaoCollectionNewMovimentacao);
                } else {}
```

ProdutoJpaController.java

Source | History

```
    public void destroy(Integer id) throws NonexistentEntityException {
        EntityManager em = null;
        try {
            em = getEntityManager();
            em.getTransaction().begin();
            Produto produto = null;
            produto = em.getReference(type: Produto.class, o: id);
            produto.getIdProduto();
            Iterable<Movimento> movimentoCollection = produto.getMovimentoCollection();
            for (Movimento movimentoCollectionMovimento : movimentoCollection) {
                movimentoCollectionMovimento.setIdProduto(idProduto: null);
                movimentoCollectionMovimento = em.merge(t: movimentoCollectionMovimento);
            }
            em.remove(o: produto);
            em.getTransaction().commit();
        } finally {
            if (em != null) {
                em.close();
            }
        }
    }
    public List<Produto> findProdutoEntities() {
        return findProdutoEntities(all:true, maxResults: -1, firstResult:-1);
    }
    public List<Produto> findProdutoEntities(int maxResults, int firstResult) {
        return findProdutoEntities(all:false, maxResults, firstResult);
    }
    private List<Produto> findProdutoEntities(boolean all, int maxResults, int firstResult) {
        EntityManager em = getEntityManager();
        try {
            CriteriaQuery cq = em.getCriteriaBuilder().createQuery();
            cq.select(slctn: cq.from(type: Produto.class));
            Query q = em.createQuery(cq);
            if (!all) {
                q.setMaxResults(i: maxResults);
                q.setFirstResult(i: firstResult);
            }
            return q.getResultList();
        } finally {
            em.close();
        }
    }
    public Produto findProduto(Integer id) {
        EntityManager em = getEntityManager();
        try {
```

ProdutoJpaController.java

Source | History

```
139                 return em.find(type: Produto.class, o: id);
140             } finally {
141                 em.close();
142             }
143         }
144         public int getProdutoCount() {
145             EntityManager em = getEntityManager();
146             try {
                    CriteriaQuery cq = em.getCriteriaBuilder().createQuery();
                    Root<Produto> rt = cq.from(type: Produto.class);
                    cq.select(slctn: em.getCriteriaBuilder().count(exprsn: rt));
                    Query q = em.createQuery(cq);
                    return ((Long) q.getSingleResult()).intValue();
152             } finally {
153                 em.close();
154             }
155         }
156         private List<Produto> findProdutosEntities(boolean all, int maxResults, int firstResult) {
157         EntityManager em = getEntityManager();
158         try {
                CriteriaQuery cq = em.getCriteriaBuilder().createQuery();
                cq.select(slctn: cq.from(type: Produto.class));
                Query q = em.createQuery(cq);
162             if (!all) {
163             q.setMaxResults(i: maxResults);
164             q.setFirstResult(i: firstResult);
165             }
                return q.getResultList();
167         } finally {
168             em.close();}
169         }
170          public List<Produto> getListaProdutos() {
171         EntityManager em = getEntityManager();
172         try {
173             CriteriaQuery<Produto> cq = em.getCriteriaBuilder().createQuery(type: Produto.class);
174             cq.select(slctn: cq.from(type: Produto.class));
175             TypedQuery<Produto> query = em.createQuery(cq);
176             return query.getResultList();
177         } finally {
178             em.close();}
179             }
180         private static class movimentoCollectionMovimento {
181             private static void setIdProduto(Produto produto) {
182                 throw new UnsupportedOperationException(message:"Not supported yet."); // Generated fro
183             }
```

ProdutoJpaController.java

Source | History

```
163             q.setMaxResults(i: maxResults);
164             q.setFirstResult(i: firstResult);
165             }
166             return q.getResultList();
167         } finally {
168             em.close();}
169         }
170          public List<Produto> getListaProdutos() {
171         EntityManager em = getEntityManager();
172         try {
173             CriteriaQuery<Produto> cq = em.getCriteriaBuilder().createQuery(type: Produto.class);
174             cq.select(slctn: cq.from(type: Produto.class));
175             TypedQuery<Produto> query = em.createQuery(cq);
176             return query.getResultList();
177         } finally {
178             em.close();}
179             }
180         private static class movimentoCollectionMovimento {
181             private static void setIdProduto(Produto produto) {
182                 throw new UnsupportedOperationException(message:"Not supported yet."); // Generated from nbfs://nbhost/SystemFileSystem/Templates/Cl
183             }
184             public movimentoCollectionMovimento() {
185             }
186         }
187         private static class ex {
188             public ex() {
189             }
190         }
191     }
192
```

UsuarioJpaController.java;

UsuarioJpaController.java

Source History

```
143     public Usuario findUsuariosenha(String login, String senha) {
144         EntityManager em = getEntityManager();
145         try {
146             TypedQuery<Usuario> query = em.createQuery(
147                 string: "SELECT u FROM Usuario u WHERE u.login = :login AND u.senha = :senha"
148   , type: Usuario.class);
149
150
151             query.setParameter(string: "login", o: login);
152             query.setParameter(string: "senha", o: senha);
153             return query.getSingleResult(); // Retorna o usuário se encontrado
154         } catch (NoResultException e) {
155             return null; // Retorna null se não encontrar um usuário com as credenciais
156         } finally {
157             em.close();
158
159         }
160     }
161     public List<Usuario> findUsuarioEntities() {
162         return findUsuarioEntities(all:true, maxResults: -1, firstResult:-1);
163     }
164
165     public List<Usuario> findUsuarioEntities(int maxResults, int firstResult) {
166         return findUsuarioEntities(all:false, maxResults, firstResult);
167     }
168
169     private List<Usuario> findUsuarioEntities(boolean all, int maxResults, int firstResult) {
170         EntityManager em = getEntityManager();
171         try {
            CriteriaQuery cq = em.getCriteriaBuilder().createQuery();
            cq.select(slctn: cq.from(type: Usuario.class));
            Query q = em.createQuery(cq);
175             if (!all) {
176                 q.setMaxResults(i: maxResults);
177                 q.setFirstResult(i: firstResult);
178             }
            return q.getResultList();
180         } finally {
181             em.close();
182         }
183     }
184
185     public Usuario findUsuario(Integer login) {
186         EntityManager em = getEntityManager();
187         try {
```

UsuarioJpaController.java

Source History

```
2   package controller;
3
4   import java.io.Serializable;
5   import java.util.ArrayList;
6   import java.util.Collection;
7   import java.util.List;
    import javax.persistence.Query;
9   import javax.persistence.EntityNotFoundException;
10  import javax.persistence.criteria.CriteriaQuery;
11  import javax.persistence.criteria.Root;
12  import javax.persistence.EntityManager;
13  import javax.persistence.EntityManagerFactory;
14  import javax.persistence.NoResultException;
15  import javax.persistence.TypedQuery;
16
17
18
19  import model.Movimento;
20  import model.Usuario;
21
    public class UsuarioJpaController implements Serializable {
23
        private Object senha;
        private Object movimentoCollectionMovimento;
26
27      public UsuarioJpaController(EntityManagerFactory emf) {
            this.emf = emf;
29      }
30      private EntityManagerFactory emf = null;
31
32      public EntityManager getEntityManager() {
33          return emf.createEntityManager();
34      }
35
36      public void create(Usuario usuario) {
37          if (usuario.getMovimentoCollection() == null) {
                usuario.setMovimentoCollection(new ArrayList<Movimento>());
39          }
40          EntityManager em = null;
41          try {
42              em = getEntityManager();
43              em.getTransaction().begin();
                Collection<Movimento> attachedMovimentoCollection = new ArrayList<Movimento>();
45              for (Movimento movimentoCollectionMovimentoToAttach : usuario.getMovimentoCollection()) {
                    Movimento movimentoCollectionMovimentacaoToAttach = em.getReference(type: movimentoCollectionMovimentoToAttach.getClass(), o: movime
```

UsuarioJpaController.java

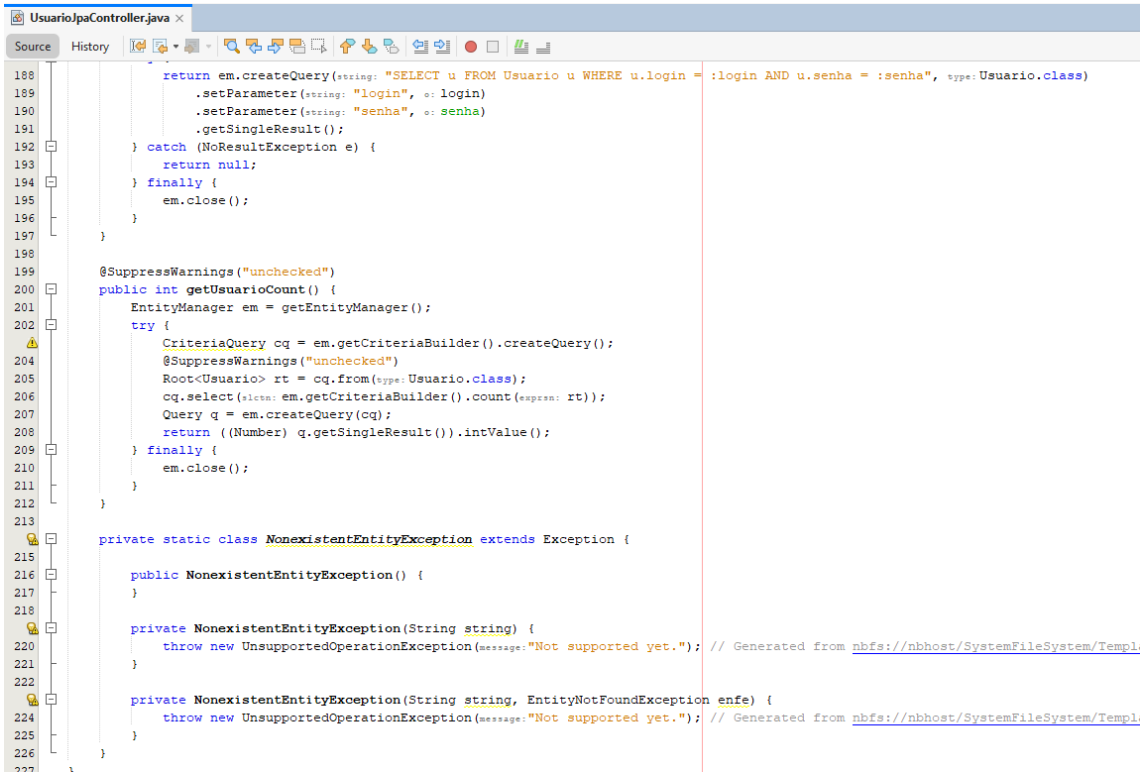
Source | History

```
                attachedMovimentoCollection.add(e: movimentoCollectionMovimentoToAttach);
            }
            usuario.setMovimentoCollection(movimentoCollection:attachedMovimentoCollection);
            em.persist(o: usuario);
            for (Movimento movimentoCollectionMovimento : usuario.getMovimentoCollection()) {
                Usuario oldIdUsuarioOfMovimentoCollectionMovimento = movimentoCollectionMovimento.getIdUsuario();
                movimentoCollectionMovimento.setIdUsuario(idUsuario: usuario);
                movimentoCollectionMovimento = em.merge(o: movimentoCollectionMovimento);
                if (oldIdUsuarioOfMovimentoCollectionMovimento != null) {
                    oldIdUsuarioOfMovimentoCollectionMovimento.getMovimentoCollection().remove(o: movimentoCollectionMovimento);
                    oldIdUsuarioOfMovimentoCollectionMovimento = em.merge(o: oldIdUsuarioOfMovimentoCollectionMovimento);
                }
            }
            em.getTransaction().commit();
        } finally {
            if (em != null) {
                em.close();
            }
        }
    }

    public void edit(Usuario usuario, EntityManager movimentoCollectionNewMovimentacaoToAttach) throws NonexistentEntityException, Exception {
        EntityManager em = null;
        try {
            em = getEntityManager();
            em.getTransaction().begin();
            Usuario persistentUsuario = em.find(type: Usuario.class, o: usuario.getId());
            Collection<Movimento> movimentacaoCollectionOld = persistentUsuario.getMovimentoCollection();
            Collection<Movimento> movimentoCollectionNew = usuario.getMovimentoCollection();
            Collection<Movimento> attachedMovimentoCollectionNew = new ArrayList<Movimento>();
            for (Movimento movimentoCollectionNewMovimentoToAttach : movimentoCollectionNew) {

                attachedMovimentoCollectionNew.add(e: movimentoCollectionNewMovimentoToAttach);
            }
            movimentoCollectionNew = attachedMovimentoCollectionNew;
            usuario.setMovimentoCollection(movimentoCollection:movimentoCollectionNew);
            usuario = em.merge(o: usuario);
            for (Movimento movimentoCollectionOldMovimento : movimentacaoCollectionOld) {
                if (!movimentoCollectionNew.contains(o: movimentoCollectionOldMovimento)) {
                    movimentoCollectionOldMovimento.setIdUsuario(idUsuario: null);
                    movimentoCollectionOldMovimento = em.merge(o: movimentoCollectionOldMovimento);
                }
            }
            for (Movimento movimentoCollectionNewMovimento : movimentoCollectionNew) {
                if (!movimentacaoCollectionOld.contains(o: movimentoCollectionNewMovimento)) {
```

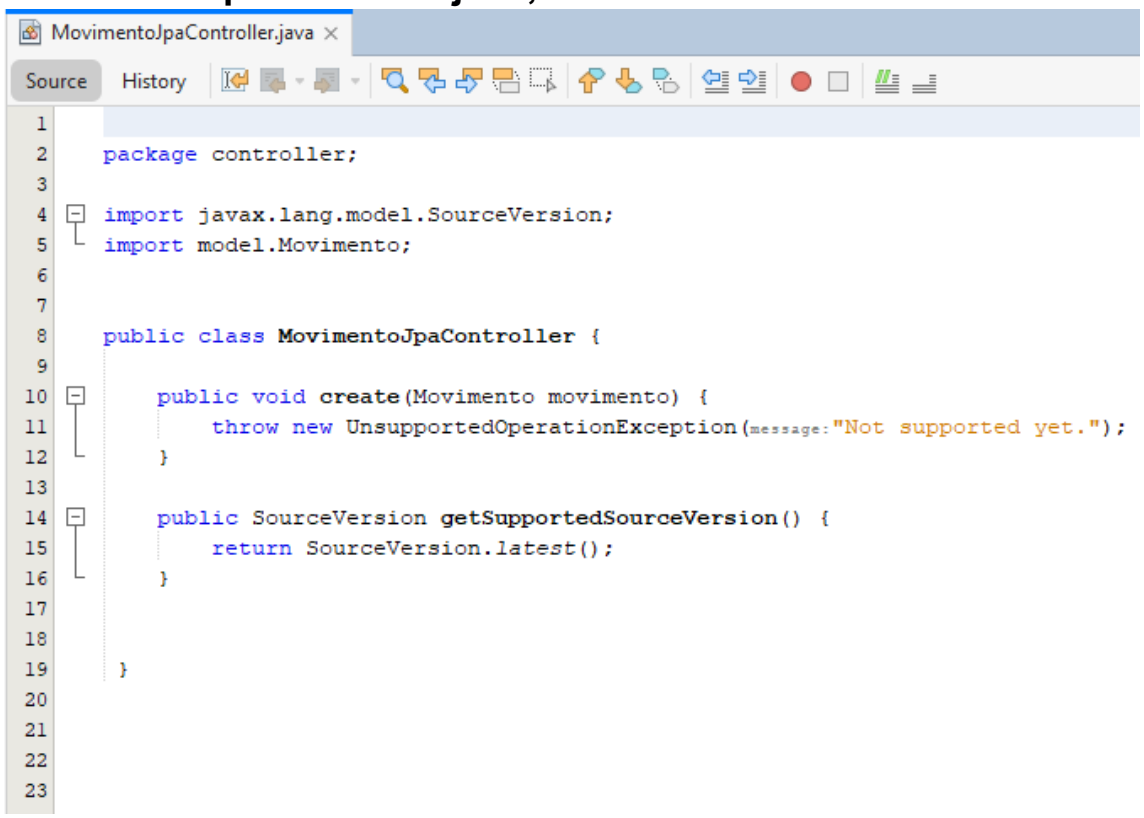
UsuarioJpaController.java

Source | History

```
                    movimentoCollectionNewMovimento.setIdUsuario(idUsuario: usuario);
                    movimentoCollectionNewMovimento = em.merge(o: movimentoCollectionNewMovimento);
                    if (oldIdUsuarioOfMovimentoCollectionNewMovimento != null && !oldIdUsuarioOfMovimentoCollectionNewMovimento.equals(object: usua
                        oldIdUsuarioOfMovimentoCollectionNewMovimento.getMovimentoCollection().remove(o: movimentoCollectionNewMovimento);
                        oldIdUsuarioOfMovimentoCollectionNewMovimento = em.merge(o: oldIdUsuarioOfMovimentoCollectionNewMovimento);
                    }
                }
            }
            em.getTransaction().commit();
        } catch (Exception ex) {
            String msg = ex.getLocalizedMessage();
            if (msg == null || msg.length() == 0) {
                Integer id = usuario.getId();
                if (findUsuario(login: id) == null) {
                    throw new NonexistentEntityException("The usuario with id " + id + " no longer exists.");
                }
            }
            throw ex;
        } finally {
            if (em != null) {
                em.close();
            }
        }
    }

    public void destroy(Integer id) throws NonexistentEntityException {
        EntityManager em = null;
        try {
            em = getEntityManager();
            em.getTransaction().begin();
            Usuario usuario;
            try {
                usuario = em.getReference(type: Usuario.class, o: id);
                usuario.getId();
            } catch (EntityNotFoundException enfe) {
                throw new NonexistentEntityException("The usuario with id " + id + " no longer exists.", enfe);
            }
            Collection<Movimento> movimentoCollection = usuario.getMovimentoCollection();
            for (Movimento movimentacaoCollectionMovimentacao : movimentoCollection) {

                movimentoCollectionMovimento = em.merge(o: movimentoCollectionMovimento);
            }
            em.remove(o: usuario);
            em.getTransaction().commit();
        } finally {
```



MovimentoJpaController.java;



CadastroServer.java:

CadastroServer.java

```java
package cadastroserver;
import java.io.Serializable;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
@Entity
public class CadastroServer implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    @Override
    public int hashCode() {
        int hash = 0;
        hash += (id != null ? id.hashCode() : 0);
        return hash;
    }
    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof CadastroServer)) {
            return false;
        }
        CadastroServer other = (CadastroServer) object;
        return !((this.id == null && other.id != null) || (this.id != null && !this.id.equals(obj:other.id)));
    }
    @Override
    public String toString() {
        return "cadastroserver.CadastroServer[ id=" + id + " ]";
    }
}
```

CadastroThread.java:

CadastroThread.java

```java
package cadastroserver;

import controller.MovimentoJpaController;
import controller.PessoaJpaController;
import controller.ProdutoJpaController;
import controller.UsuarioJpaController;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import model.Movimento;
import model.Pessoa;
import model.Produto;
import model.Usuario;


public class CadastroThread extends Thread {
    private final ProdutoJpaController ctrlProduto;
    private final UsuarioJpaController ctrlUsuario;
    private final MovimentoJpaController ctrlMovimento;
    private final PessoaJpaController ctrlPessoa;
    private final Socket socl;
    public CadastroThread(ProdutoJpaController ctrlProduto, UsuarioJpaController ctrlUsuario,
            MovimentoJpaController ctrlMovimento, PessoaJpaController ctrlPessoa, Socket socl) {
        this.ctrlProduto = ctrlProduto;
        this.ctrlUsuario = ctrlUsuario;
        this.ctrlMovimento = ctrlMovimento;
        this.ctrlPessoa = ctrlPessoa;
        this.socl = socl;
    }
    @Override
    public void run() {
        try (
                ObjectOutputStream saida = new ObjectOutputStream(out:socl.getOutputStream());
                ObjectInputStream entrada = new ObjectInputStream(in: socl.getInputStream())) {
            String login = (String) entrada.readObject();
            String senha = (String) entrada.readObject();
            Usuario usuario = ctrlUsuario.findUsuariosenha(login, senha);
            if (usuario == null) {
                System.out.println(x: "Usuário inválido. Conexão encerrada.");
                return;
            }
```

CadastroThread.java

Source | History

```
36          try {
37              ObjectOutputStream saida = new ObjectOutputStream(out:socl.getOutputStream());
38              ObjectInputStream entrada = new ObjectInputStream(in: socl.getInputStream())) {
39              String login = (String) entrada.readObject();
40              String senha = (String) entrada.readObject();
41              Usuario usuario = ctrlUsuario.findUsuariosenha(login, senha);
42              if (usuario == null) {
43                  System.out.println(x: "Usuário inválido. Conexão encerrada.");
44                  return;
45              }
46              while (true) {
47                  String comando = (String) entrada.readObject();
48                  if ("L".equals(anObject: comando)) {
49                      List<Produto> produtos = ctrlProduto.findProdutoEntities();
50                      saida.writeObject(obj:produtos);
51
52                  } else if ("E".equalsIgnoreCase(anotherString: comando)) {
53                      if (EntradaMovimento(entrada, usuario)) {
54                          saida.writeObject(obj:"EntradaMovimento realizada com sucesso.");
55                      } else {
56                          saida.writeObject(obj:"Erro ao realizar entrada.");
57                      }
58                  } else if ("S".equalsIgnoreCase(anotherString: comando)) {
59                      if (SaidaMovimento(entrada, usuario)) {
60                          saida.writeObject(obj:"Saida realizada com sucesso.");
61                      } else {
62                          saida.writeObject(obj:"Erro ao realizar saida.");
63                      }
64                  } else if ("X".equals(anObject: comando)) {
65                      saida.writeObject(obj:"SAINDO");
66                  }
67              }
68          } catch (IOException | ClassNotFoundException e) {
69          } catch (Exception ex) {
70              Logger.getLogger(name:CadastroThread.class.getName()).log(level: Level.SEVERE, msg:null, thrown: ex);
71          } finally {
72              try {
73                  socl.close();
74              } catch (IOException e) {
75              }
76          }
77      }
78      private boolean EntradaMovimento(ObjectInputStream entrada, Usuario usuario, Movimento movimento) throws IOException, ClassNotFoundException
79          Integer idPessoaObj = (Integer) entrada.readObject();
80          Integer idProdutoObj = (Integer) entrada.readObject();
```

CadastroThread.java

Source | History

```
81          Integer quantidadeObj = (Integer) entrada.readObject();
82          Double valorUnitarioObj = (Double) entrada.readObject();
83          int idPessoa = idPessoaObj;
84          int idProduto = idProdutoObj;
85          int quantidade = quantidadeObj;
86          double valorUnitario = valorUnitarioObj;
87          Pessoa pessoa = ctrlPessoa.findPessoa(id: idPessoa);
88          Produto produto = ctrlProduto.findProduto(id: idProduto);
89          if (pessoa == null || produto == null) {
90              System.out.println(x: "Pessoa ou Produto não encontrado. Movimento não registrado.");
91              return false;
92          }
93          if (quantidade <= 0) {
94              System.out.println(x: "Quantidade inválida. Movimento não registrado.");
95              return false;
96          }
97          Movimento movimentacao = new Movimento();
98          movimentacao.setIdUsuario(idUsuario: usuario);
99          movimentacao.setTipo(tipo: "E");
100         movimentacao.setIdPessoa(idPessoa:pessoa);
101         movimentacao.setIdProduto(idProduto: produto);
102         movimentacao.setQuantidade(quantidade);
103         movimentacao.setValorUnitario(valorUnitario);
104         int novaQuantidade = produto.getQuantidade() + quantidade;
105         try {
106             produto.setQuantidade(novaQuantidade);
107             ctrlProduto.edit(produto);
108         } catch (Exception ex) {
109             System.out.println(x: "Erro ao realizar a persistencia em produto.");
            ex.printStackTrace();
111             return false;
112         }
113         try {
114             ctrlMovimento.create(movimento);
115             return true;
116         } catch (Exception ex) {
117             System.out.println(x: "Erro ao realizar a persistencia em movimento.");
            ex.printStackTrace();
119             return false;
120         }
121     }
    private boolean SaidaMovimento(ObjectInputStream entrada, Usuario usuario, Movimento movimento) throws IOException, ClassNotFoundException
123         Integer idPessoaObj = (Integer) entrada.readObject();
124         Integer idProdutoObj = (Integer) entrada.readObject();
125         Integer quantidadeObj = (Integer) entrada.readObject();
```

CadastroThread.java

Source | History

```
126             Double valorUnitarioObj = (Double) entrada.readObject();
127             int idPessoa = idPessoaObj;
128             int idProduto = idProdutoObj;
129             int quantidade = quantidadeObj;
130             double valorUnitario = valorUnitarioObj;
131             Pessoa pessoa = ctrlPessoa.findPessoa(id: idPessoa);
132             Produto produto = ctrlProduto.findProduto(id: idProduto);
133             if (pessoa == null || produto == null) {
134                 System.out.println(x: "Pessoa ou Produto não encontrado. Movimento não registrado.");
135                 return false;
136             }
137             if (quantidade <= 0) {
138                 System.out.println(x: "Quantidade inválida. Movimento não registrado.");
139                 return false;
140             }
141             int novaQuantidade = produto.getQuantidade() - quantidade;
142             if (novaQuantidade >= 0) {
143                 Movimento movimentacao = new Movimento();
144                 movimentacao.setIdUsuario(idUsuario: usuario);
145                 movimentacao.setTipo(tipo: "S");
146                 movimentacao.setIdPessoa(idPessoa: pessoa);
147                 movimentacao.setIdProduto(idProduto: produto);
148                 movimentacao.setIdMovimento(idMovimento:quantidade);
149                 movimentacao.setValorUnitario(valorUnitario);
150                 try {
151                     produto.setQuantidade(novaQuantidade);
152                     ctrlProduto.edit(produto);
153                 } catch (Exception ex) {
154                     System.out.println(x: "Erro ao realizar a persistencia em produto.");
155                     return false;
156                 }
157                 try {
158                     ctrlMovimento.create(movimento);
159                     return true;
160                 } catch (Exception ex) {
161                     System.out.println(x: "Erro ao realizar a persistencia em movimento.");
162                     return false;
163                 }
164             } else {
165                 System.out.println(x: "Estoque insuficiente para a saida.");
166                 return false;
167             }
168         }
169
170         private boolean EntradaMovimento(ObjectInputStream entrada, Usuario usuario) {
```

```
169
170         private boolean EntradaMovimento(ObjectInputStream entrada, Usuario usuario) {
171             throw new UnsupportedOperationException(message:"Not supported yet."); // Generated from
172         }
173
174         private boolean SaidaMovimento(ObjectInputStream entrada, Usuario usuario) {
175             throw new UnsupportedOperationException(message:"Not supported yet."); // Generated from
176         }
177     }
```