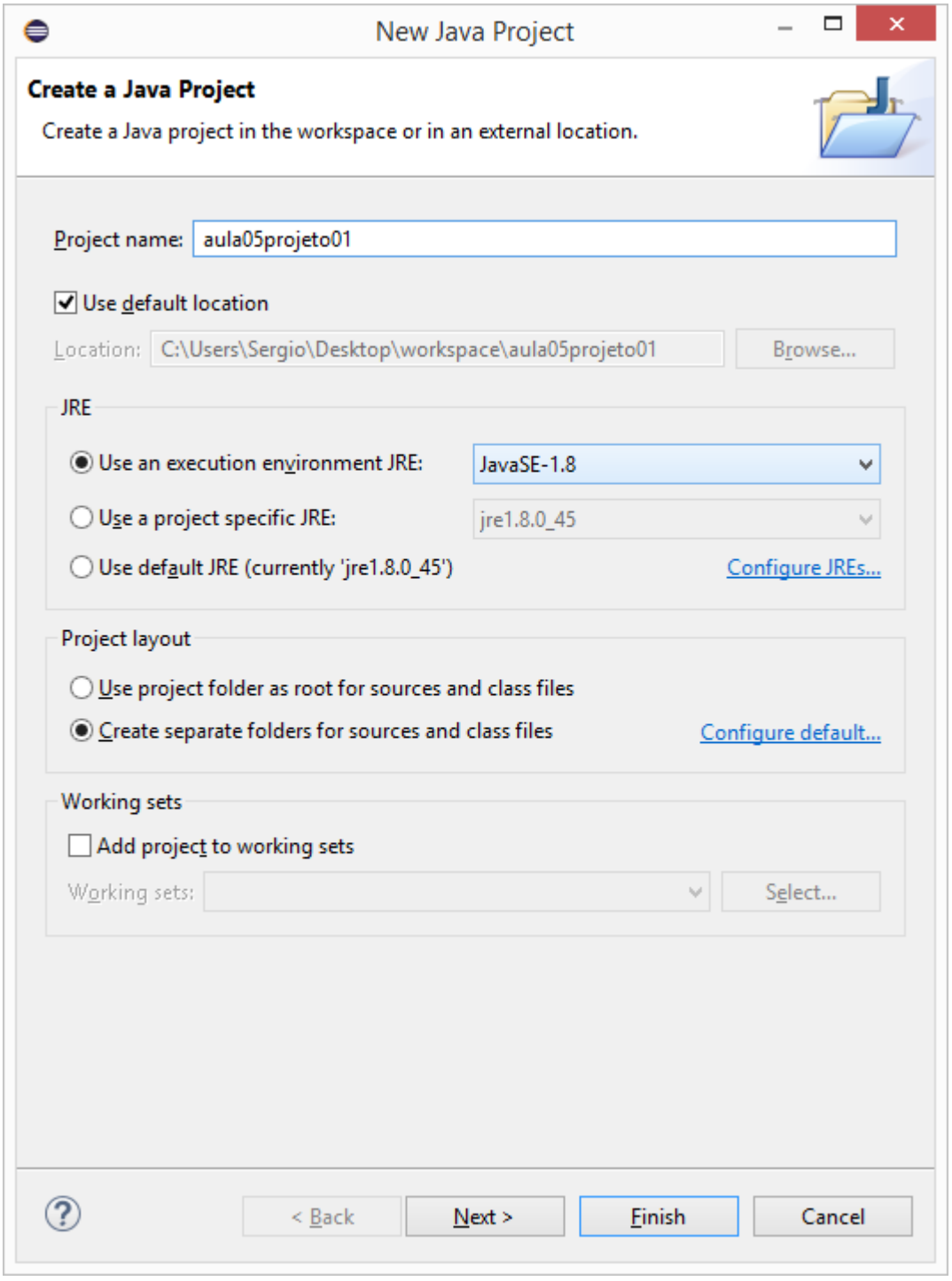
Criando um novo projeto

File > New > Java Project



Criando um JavaBean...

```
package entities;

public class Produto implements Comparable<Produto>{

	private Integer idProduto;
	private String nome;
	private Double preco;
	private Integer quantidade;

	public Produto() {
		// TODO Auto-generated constructor stub
	}

	public Produto(Integer idProduto, String nome,
		Double preco, Integer quantidade) {
		this.idProduto = idProduto;
		this.nome = nome;
		this.preco = preco;
		this.quantidade = quantidade;
	}

	public Integer getIdProduto() {
		return idProduto;
	}

	public void setIdProduto(Integer idProduto) {
		this.idProduto = idProduto;
	}

	public String getNome() {
		return nome;
	}

	public void setNome(String nome) {
		this.nome = nome;
	}

	public Double getPreco() {
		return preco;
	}

	public void setPreco(Double preco) {
		this.preco = preco;
	}
```

```
	public Integer getQuantidade() {
		return quantidade;
	}

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

	@Override
	public String toString() {
		return idProduto + ", " + nome + ", "
				+ preco + ", " + quantidade;
	}

	@Override
	public boolean equals(Object obj) {

		if(obj instanceof Produto){
			Produto p = (Produto) obj;
			if(p.getIdProduto() != null){
				return p.getIdProduto().equals(idProduto);
			}
		}
		return false;
	}

	@Override
	public int hashCode() {
		return idProduto.hashCode();
	}

	@Override
	public int compareTo(Produto p) {
		return idProduto.compareTo(p.getIdProduto());
	}
}
```

Definindo uma interface para controle de produtos:

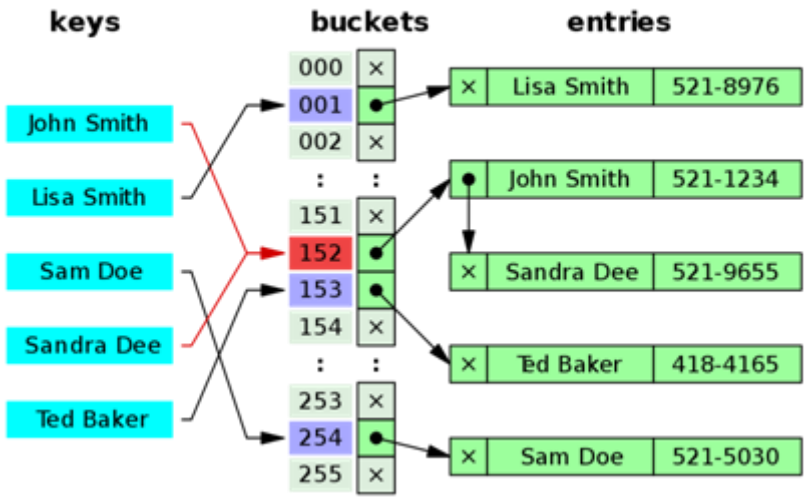
```
package contracts;

import entities.Produto;
```

```
public interface IControleProduto {

	//método para adicionar um produto em um mapa..
	void adicionarProduto(Produto p) throws Exception;

	//método para obter 1 produto do mapa..
	Produto obterProduto(Integer idProduto) throws Exception;

}
```

Mapas

Os objetos "Map" confiam seus dados em um algoritmo hash (hash code). Esse algoritmo transforma uma grande quantidade de dados em uma pequena quantidade de informações, sendo que o mecanismo de busca se baseia na construção de índices.

Um exemplo prático pode ser usado como uma lista telefônica onde a letra seria o índice a ser procurado, para conseguir achar mais fácil o nome desejado.



Essa interface é um objeto que mapeia valores para chaves, ou seja, através da chave consegue ser acessado o valor configurado, sendo que a chave não pode ser repetida ao contrário do valor, mas se caso tiver uma chave repetida é sobrescrito pela última chamada. Também faz parte do pacote "java.util" e não possui métodos da interface Collection.

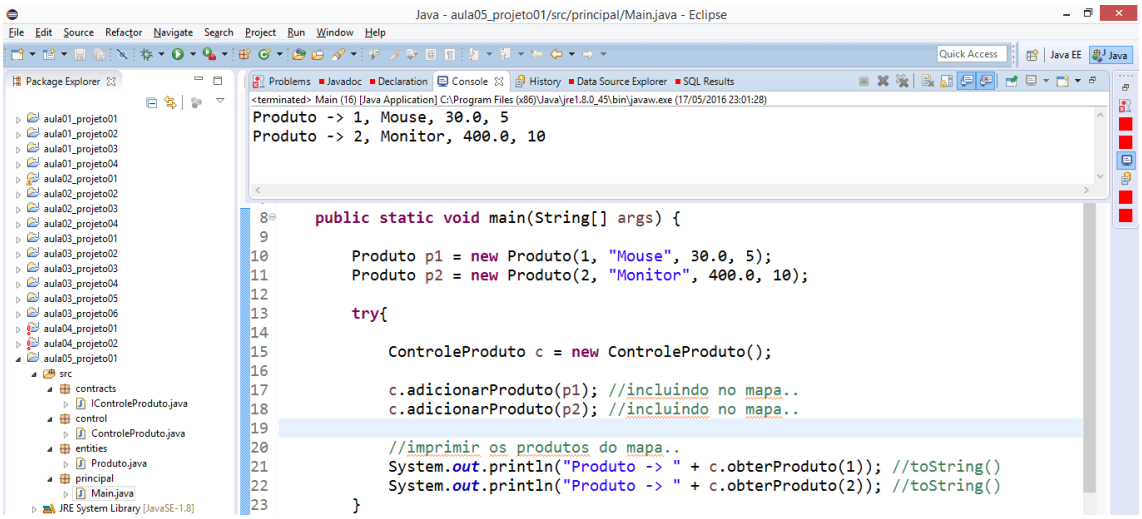
Manipulando um mapa de Produtos:

```
package control;

import java.util.LinkedHashMap;
import java.util.Map;

import contracts.IControleProduto;
import entities.Produto;

public class ControleProduto implements IControleProduto{

	//declarando um mapa de produtos..
	private Map<Integer, Produto> mapa;

	//construtor..
	public ControleProduto() {
		//inicializando o mapa..
		mapa = new LinkedHashMap<Integer, Produto>();
	}

	@Override
	public void adicionarProduto(Produto p) throws Exception  {

		//verificar se o mapa não possui um produto com a chave informada..
		if( ! mapa.containsKey(p.getIdProduto())){
			//adicionar o produto dentro do mapa..
			mapa.put(p.getIdProduto(), p);
		}
		else{
			throw new Exception("Erro. Este produto ja foi adicionado ao mapa.");
		}
	}

	@Override
	public Produto obterProduto(Integer idProduto) throws Exception {

		//verificando se o produto ja existe dentro do mapa..
		if(mapa.containsKey(idProduto)){
			//retornando o produto..
			return mapa.get(idProduto);
		}
		else{
			throw new Exception("Erro. Produto não encontrado no mapa.");
		}
	}

}
```

Executando:

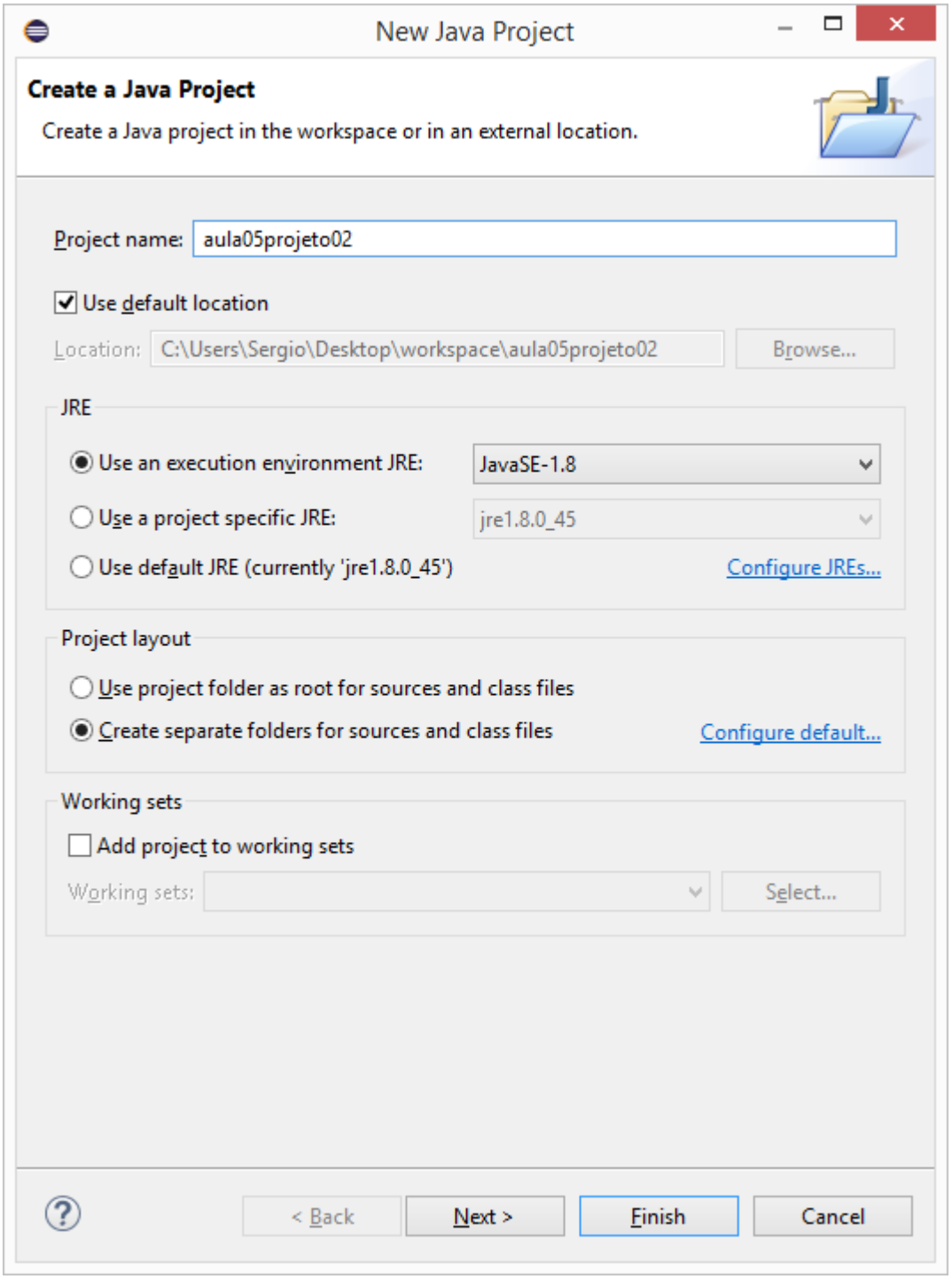
```
package principal;

import control.ControleProduto;
import entities.Produto;

public class Main {

	public static void main(String[] args) {

		Produto p1 = new Produto(1, "Mouse", 30.0, 5);
		Produto p2 = new Produto(2, "Monitor", 400.0, 10);

		try{

			ControleProduto c = new ControleProduto();

			c.adicionarProduto(p1); //incluindo no mapa..
			c.adicionarProduto(p2); //incluindo no mapa..

			//imprimir os produtos do mapa..
			System.out.println("Produto -> " + c.obterProduto(1)); //toString()
			System.out.println("Produto -> " + c.obterProduto(2)); //toString()
		}
		catch(Exception e){
			System.out.println(e.getMessage());
		}
	}
}
```

Executando:



Novo projeto:

File > New > Java Project



Criando o script da base de dados:
script.sql

```
drop table if exists produto;
drop table if exists estoque;

create table estoque(
	idestoque		integer		auto_increment,
	nome			varchar(50)	not null,
	descricao		varchar(100)	not null,
	primary key(idestoque));

create table produto(
	idproduto		integer		auto_increment,
	nome			varchar(50)	not null,
	preco			double		not null,
	quantidade		integer		not null,
	idestoque		integer		not null,
	primary key(idproduto),
	foreign key(idestoque) references estoque(idestoque));

insert into estoque(nome, descricao)
values('Informatica', 'Produtos de Informatica');

insert into estoque(nome, descricao)
values('Eletronicos', 'Produtos Eletronicos');

select * from estoque;

--------------------

insert into produto(nome, preco, quantidade, idestoque) values
('Mouse', 30.0, 10, 1),
('Monitor', 200, 15, 1),
('Computador', 1500.0, 10, 1),
('TV', 2000.0, 5, 2),
('Celular', 400.0, 6, 2),
('DVD', 200.0, 18, 2);

select * from produto;

-- consulta que retorne todos os produtos com preco de 10 a 1000
select * from produto
where preco >= 10 and preco <= 1000;

select * from produto
where preco between 10 and 1000;
```

```
-- consulta que retorne produtos pelo nome (like)
select * from produto
where nome like 'M%';

--ordenação..
select * from produto order by nome asc;
select * from produto order by nome desc;

--funções de calculo em banco de dados..
select sum(preco) as 'Somatorio de Preco' from produto;
select avg(preco) as 'Media de Preco' from produto;

select max(preco) as 'Maior Preco' from produto;
select min(preco) as 'Menor Preco' from produto;

select count(*) as 'Quantidade de Registros' from produto;

----------------------

--quais produtos tem o preco maior ou igual a media..
select * from produto
where preco >= (select avg(preco) from produto);

--mostrar os dados dos produtos com o seu estoque..
select
	p.idproduto,
	p.nome as nomeproduto,
	p.preco,
	p.quantidade,
	e.idestoque,
	e.nome as nomeestoque,
	e.descricao
from
produto p
inner join
estoque e
on
e.idestoque = p.idestoque;

-- somatorio de quantidade de produtos por estoque..
select
	e.idestoque,
	e.nome,
	e.descricao,
	sum(p.quantidade) as 'Qtd total de Produtos'
from produto p
inner join estoque e
on e.idestoque = p.idestoque
```

```
group by
	idestoque, nome, descricao;


-- somatorio dos precos de produtos por estoque..
select
	e.idestoque,
	e.nome,
	e.descricao,
	sum(p.preco) as 'Somatorio de Preco'
from produto p
inner join estoque e
on e.idestoque = p.idestoque
group by
	idestoque, nome, descricao;


-- total de cada produto (preco * quantidade)
select
	p.idproduto,
	p.nome,
	p.preco,
	p.quantidade,
	(p.preco * p.quantidade) as total
from produto p;

--somatorio do total dos produtos por estoque..
select
	e.idestoque,
	e.nome,
	e.descricao,
	sum(p.preco * p.quantidade) as 'Somatorio Total'
from produto p
inner join estoque e
on e.idestoque = p.idestoque
group by
	idestoque, nome, descricao;
```

Criando as entidades:

```
package br.com.brq.entities;

import java.util.Collection;
```

```java
/**
 * Classe de entidade: Estoque
 *
 * @author Sergio Mendes
 * @version 1.0
 * @since Projeto Aula 05 - Treinamento BRQ/SP
 *
 */
public class Estoque {

	private Integer idEstoque;
	private String nome;
	private String descricao;
	private Collection<Produto> produtos;

	public Estoque() {
	}

	public Estoque(Integer idEstoque, String nome,
			String descricao) {
		this.idEstoque = idEstoque;
		this.nome = nome;
		this.descricao = descricao;
	}

	public Estoque(Integer idEstoque, String nome,
		String descricao, Collection<Produto> produtos) {
		this(idEstoque, nome, descricao);
		this.produtos = produtos;
	}

	public Integer getIdEstoque() {
		return idEstoque;
	}

	public void setIdEstoque(Integer idEstoque) {
		this.idEstoque = idEstoque;
	}

	public String getNome() {
		return nome;
	}

	public void setNome(String nome) {
		this.nome = nome;
	}
```

```
	public String getDescricao() {
		return descricao;
	}

	public void setDescricao(String descricao) {
		this.descricao = descricao;
	}

	public Collection<Produto> getProdutos() {
		return produtos;
	}

	public void setProdutos(Collection<Produto> produtos) {
		this.produtos = produtos;
	}

	@Override
	public String toString() {
		return "Estoque [idEstoque=" + idEstoque + ", nome="
				+ nome + ", descricao=" + descricao + "]";
	}
}


package br.com.brq.entities;

/**
 * Classe de entidade: Produto
 *
 * @author Sergio Mendes
 * @version 1.0
 * @since Projeto Aula 05 - Treinamento BRQ/SP
 *
 */
public class Produto {

	private Integer idProduto;
	private String nome;
	private Double preco;
	private Integer quantidade;
	private Estoque estoque;

	public Produto() {
	}

	public Produto(Integer idProduto, String nome,
```

```
            Double preco, Integer quantidade) {
        this.idProduto = idProduto;
        this.nome = nome;
        this.preco = preco;
        this.quantidade = quantidade;
    }

    public Produto(Integer idProduto, String nome,
            Double preco, Integer quantidade, Estoque estoque) {
        this(idProduto, nome, preco, quantidade);
        this.estoque = estoque;
    }

    public Integer getIdProduto() {
        return idProduto;
    }

    public void setIdProduto(Integer idProduto) {
        this.idProduto = idProduto;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public Double getPreco() {
        return preco;
    }

    public void setPreco(Double preco) {
        this.preco = preco;
    }

    public Integer getQuantidade() {
        return quantidade;
    }

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

    public Estoque getEstoque() {
```

```
		return estoque;
	}

	public void setEstoque(Estoque estoque) {
		this.estoque = estoque;
	}

	@Override
	public String toString() {
		return "Produto [idProduto=" + idProduto + ", nome="
				+ nome + ", preco=" + preco
				+ ", quantidade=" + quantidade
				+ "]";
	}

}
```

Criando a Classe DAO:
Data Access Object

```
package br.com.brq.persistence;

import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

/**
 * Classe para conexão no banco de dados,
 * padrão DAO (Data Access Object)
 * @author Sergio Mendes
 * @version 1.0
 * @since Projeto Aula 05 - Treinamento BRQ/SP
 *
 */
public class Dao {

	/**
	 * Driver de conexão do MySql
	 */
	private static final String DRIVER = "com.mysql.jdbc.Driver";
```

```
/**
 * URL de acesso ao banco de dados MySql
 */
private static final String URL = "jdbc:mysql://localhost:3306/aula05";

/**
 * Nome do usuario administrador do MySql
 */
private static final String USER = "root";

/**
 * Senha do usuario administrador do MySql
 */
private static final String PASSWORD = "brqbrq";

/**
 * API JDBC para conexão com o banco de dados MySql
 */
protected Connection con;

/**
 * API JDBC para execução de rotinas SQL no banco de dados
 */
protected PreparedStatement stmt;

/**
 * API JDBC para execução de storedprocedures no banco de dados
 */
protected CallableStatement call;

/**
 * API JDBC para leitura de consultas no banco de dados
 */
protected ResultSet rs;

/**
 * Método para abrir conexão com a base de dados MySql
 * @autor Sergio Mendes
 * @throws Exception
 */
protected void openConnection() throws Exception{
```

```
                Class.forName(DRIVER);
                con = DriverManager.getConnection(URL, USER, PASSWORD);
        }

        /**
         * Método para fechar conexão com a base de dados MySql
         * @autor Sergio Mendes
         * @throws Exception
         */
        protected void closeConnection() throws Exception{
                if(con != null){
                        con.close();
                }
        }
}
```

Criando a Classe de persistência para Estoque
EstoqueDao.java

```
package br.com.brq.persistence;

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

import br.com.brq.entities.Estoque;

/**
 * Classe de persistencia para a entidade Estoque
 * @author Sergio Mendes
 * @version 1.0
 * @since Projeto Aula 05 - Treinamento BRQ/SP
 */
public class EstoqueDao extends Dao{

        /**
         * Método para cadastrar um registro na tabela de estoque
         * @param e: objeto da Classe Estoque
         * @autor Sergio Mendes
         * @throws Exception
         */
        public void insert(Estoque e) throws Exception{


                String query = "insert into estoque(nome, descricao) "
                                        + "values(?, ?)";
```

```
            openConnection();

            stmt = con.prepareStatement(query);
            stmt.setString(1, e.getNome());
            stmt.setString(2, e.getDescricao());
            stmt.execute();
            stmt.close();

            closeConnection();
    }

    /**
     * Método para atualizar um registro na tabela de estoque
     * @param e: Objeto da Classe Estoque
     * @autor Sergio Mendes
     * @throws Exception
     */
    public void update(Estoque e) throws Exception{

            String query = "update estoque set nome = ?, descricao = ? "
                                        + "where idestoque = ?";

            openConnection();

            stmt = con.prepareStatement(query);
            stmt.setString(1, e.getNome());
            stmt.setString(2, e.getDescricao());
            stmt.setInt(3, e.getIdEstoque());
            stmt.execute();
            stmt.close();

            closeConnection();
    }

    /**
     * Método para excluir um registro na tabela de estoque
     * @param idEstoque: valor inteiro correspondente a chave primaria
     * @autor Sergio Mendes
     * @throws Exception
     */
    public void delete(Integer idEstoque) throws Exception{

            String query = "delete from estoque where idestoque = ?";

            openConnection();

            stmt = con.prepareStatement(query);
```

```
            stmt.setInt(1, idEstoque);
            stmt.execute();
            stmt.close();

            closeConnection();
}

/**
 * Método para consultar todos os registros da tabela estoque
 * @return Lista contendo os objetos de Estoque obtidos do banco de dados
 * @autor Sergio Mendes
 * @throws Exception
 */
public List<Estoque> findAll() throws Exception{

            String query = "select * from estoque order by idestoque asc";

            openConnection();

            stmt = con.prepareStatement(query);
            rs = stmt.executeQuery();

            List<Estoque> lista = new ArrayList<Estoque>();

            while(rs.next()){
                        Estoque e = new Estoque();
                        e.setIdEstoque(rs.getInt("idestoque"));
                        e.setNome(rs.getString("nome"));
                        e.setDescricao(rs.getString("descricao"));

                        lista.add(e);
            }

            closeConnection();
            return lista;
}

/**
 * Método para obter 1 Estoque baseado no id
 * @param idEstoque: valor inteiro correspondente a chave primaria
 * @return Objeto da Classe Estoque ou null (não encontrado)
 * @autor Sergio Mendes
 * @throws Exception
 */


public Estoque findById(Integer idEstoque) throws Exception{
```

```
		String query = "select * from estoque where idestoque = ?";

		openConnection();

		stmt = con.prepareStatement(query);
		stmt.setInt(1, idEstoque);
		rs = stmt.executeQuery();

		Estoque e = null;

		if(rs.next()){
			e = new Estoque();
			e.setIdEstoque(rs.getInt("idestoque"));
			e.setNome(rs.getString("nome"));
			e.setDescricao(rs.getString("descricao"));
		}

		stmt.close();
		closeConnection();

		return e;
	}

}
```

Criando a Classe de persistencia para Produto
ProdutoDao.java

```
package br.com.brq.persistence;

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

import br.com.brq.entities.Estoque;
import br.com.brq.entities.Produto;

/**
 * Classe de persistencia para a entidade Estoque
 * @author Sergio Mendes
 * @version 1.0
 * @since Projeto Aula 05 - Treinamento BRQ/SP
 */


public class ProdutoDao extends Dao{
```

```
/**
 * Método para cadastrar um registro na tabela de produto
 * @param p: objeto da Classe Produto
 * @autor Sergio Mendes
 * @throws Exception
 */
public void insert(Produto p) throws Exception{

        String query = "insert into produto(nome, preco, quantidade, idestoque) "
                                + "values(?, ?, ?, ?)";

        openConnection();

        stmt = con.prepareStatement(query);
        stmt.setString(1, p.getNome());
        stmt.setDouble(2, p.getPreco());
        stmt.setInt(3, p.getQuantidade());
        stmt.setInt(4, p.getEstoque().getIdEstoque());
        stmt.execute();
        stmt.close();

        closeConnection();
}

/**
 * Método para atualizar um registro na tabela de produto
 * @param p: Objeto da Classe Produto
 * @autor Sergio Mendes
 * @throws Exception
 */
public void update(Produto p) throws Exception{

        String query = "update produto set nome = ?, preco = ?,
                                quantidade = ?, idestoque = ? "
                                + "where idestoque = ?";

        openConnection();

        stmt = con.prepareStatement(query);
        stmt.setString(1, p.getNome());
        stmt.setDouble(2, p.getPreco());
        stmt.setInt(3, p.getQuantidade());
        stmt.setInt(4, p.getEstoque().getIdEstoque());
        stmt.setInt(5, p.getIdProduto());
        stmt.execute();
        stmt.close();

        closeConnection();
```

```
}

/**
 * Método para excluir um registro na tabela de produto
 * @param idProduto: valor inteiro correspondente a chave primaria
 * @autor Sergio Mendes
 * @throws Exception
 */
public void delete(Integer idProduto) throws Exception{

	String query = "delete from produto where idproduto = ?";

	openConnection();

	stmt = con.prepareStatement(query);
	stmt.setInt(1, idProduto);
	stmt.execute();
	stmt.close();

	closeConnection();
}

/**
 * Método para consultar todos os registros da tabela produto
 * @return Lista contendo os objetos de Produto obtidos do banco de dados
 * @autor Sergio Mendes
 * @throws Exception
 */
public List<Produto> findAll() throws Exception{

	String query = "select p.idproduto, p.nome as nomeproduto,
				p.preco, p.quantidade, "
				 + "e.idestoque, e.nome as nomeestoque, e.descricao "
				 + "from produto p inner join estoque e "
				 + "on e.idestoque = p.idestoque";

	openConnection();

	stmt = con.prepareStatement(query);
	rs = stmt.executeQuery();

	List<Produto> lista = new ArrayList<Produto>();

	while(rs.next()){
		Produto p = new Produto();
		p.setEstoque(new Estoque());

		p.setIdProduto(rs.getInt("idproduto"));
```

```
                p.setNome(rs.getString("nomeproduto"));
                p.setPreco(rs.getDouble("preco"));
                p.setQuantidade(rs.getInt("quantidade"));
                p.getEstoque().setIdEstoque(rs.getInt("idestoque"));
                p.getEstoque().setNome(rs.getString("nomeestoque"));
                p.getEstoque().setDescricao(rs.getString("descricao"));

                lista.add(p);
        }

        closeConnection();
        return lista;
}

/**
 * Método para obter 1 Produto baseado no id
 * @param idProduto: valor inteiro correspondente a chave primaria
 * @return Objeto da Classe Produto ou null (não encontrado)
 * @autor Sergio Mendes
 * @throws Exception
 */
public Produto findById(Integer idProduto) throws Exception{

        String query = "select p.idproduto, p.nome as nomeproduto,
                        p.preco, p.quantidade, "
                        + "e.idestoque, e.nome as nomeestoque, e.descricao "
                        + "from produto p inner join estoque e "
                        + "on e.idestoque = p.idestoque "
                        + "where idproduto = ?";

        openConnection();

        stmt = con.prepareStatement(query);
        stmt.setInt(1, idProduto);
        rs = stmt.executeQuery();

        Produto p = null;

        if(rs.next()){
                p = new Produto();
                p.setEstoque(new Estoque());

                p.setIdProduto(rs.getInt("idproduto"));
                p.setNome(rs.getString("nomeproduto"));
                p.setPreco(rs.getDouble("preco"));
                p.setQuantidade(rs.getInt("quantidade"));
                p.getEstoque().setIdEstoque(rs.getInt("idestoque"));
                p.getEstoque().setNome(rs.getString("nomeestoque"));
```

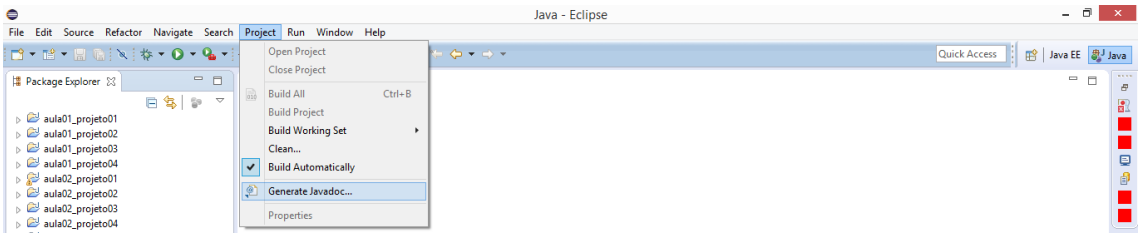
```
                p.getEstoque().setDescricao(rs.getString("descricao"));
            }

            stmt.close();
            closeConnection();

            return p;
        }
}
```

JavaDoc

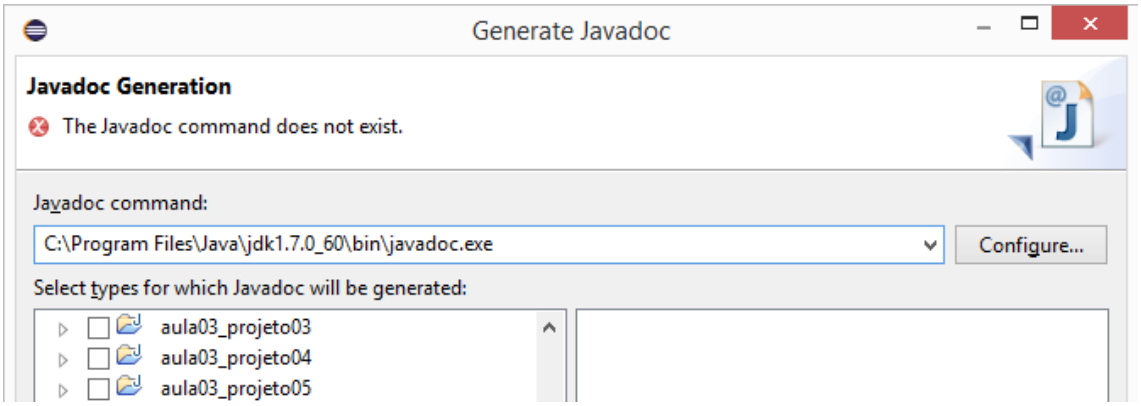
Javadoc é um gerador de documentação criado pela Sun Microsystems para documentar a API dos programas em Java, a partir do código-fonte. O resultado é expresso em HTML. É constituído, basicamente, por algumas marcações muitos simples inseridas nos comentários do programa.

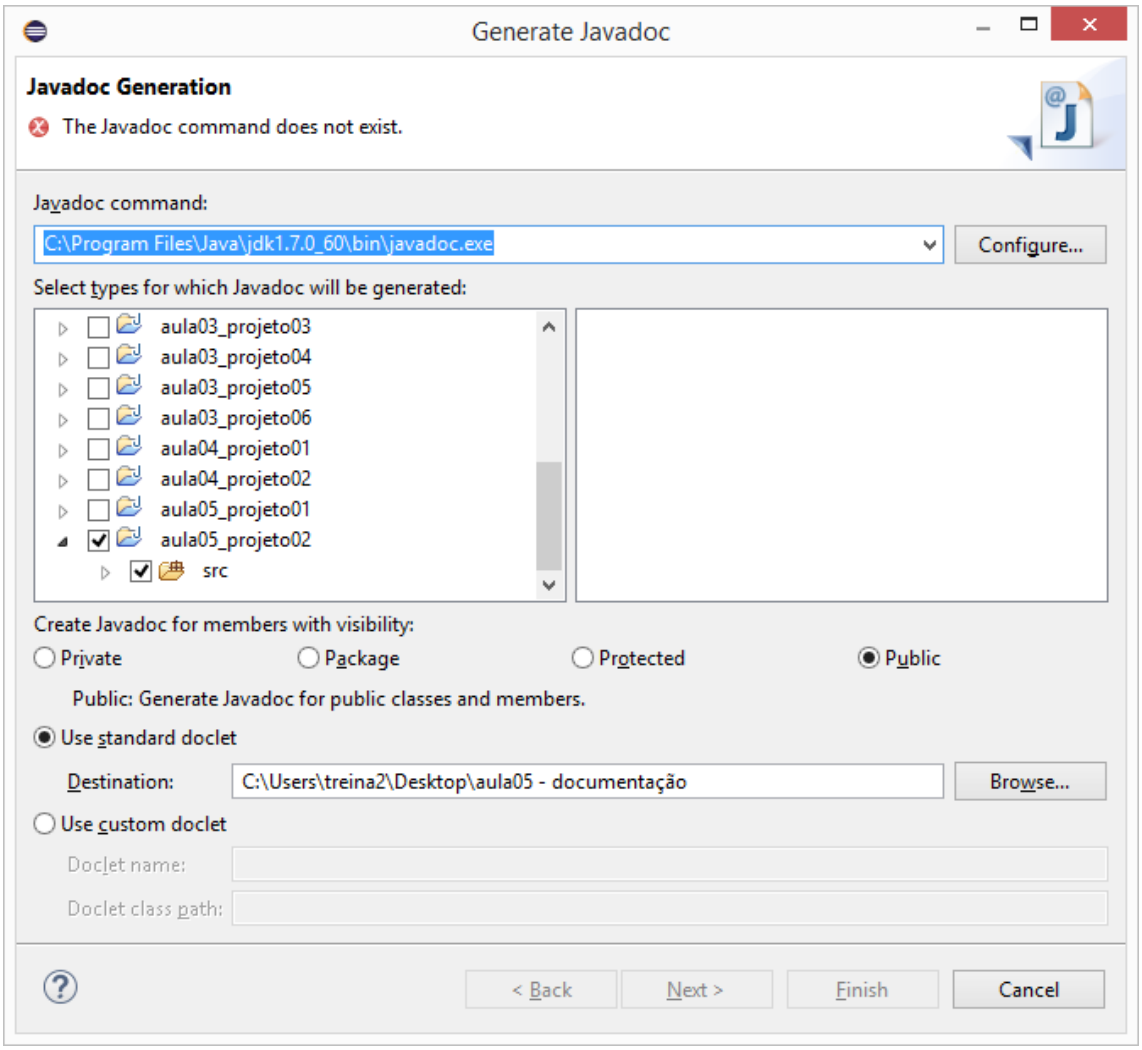
Este sistema é o padrão de documentação de classes em Java, e muitas dos IDEs desta linguagem irão automaticamente gerar um Javadoc em HTML.

Gerando JavaDoc no Eclipse



Selecione: **C:\Program Files\Java\jdk1.7.0_60\bin\javadoc.exe**





JavaDoc gerado: