

DATA ACCESS OBJECT (DAO)

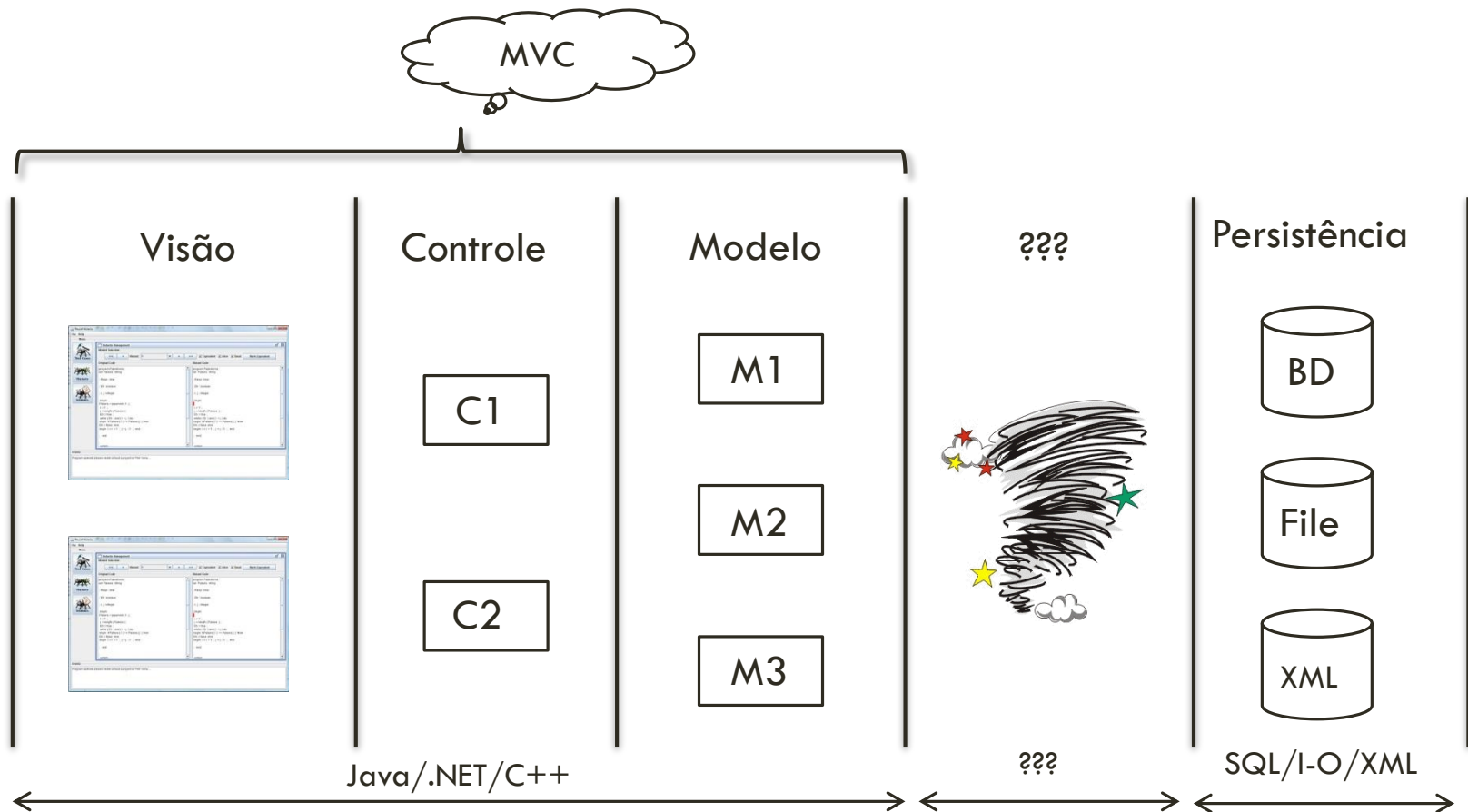
SSC 621: Análise e Projeto Orientados a Objetos
Prof. Dr. Lucas Bueno R. Oliveira

2º Semestre
2015

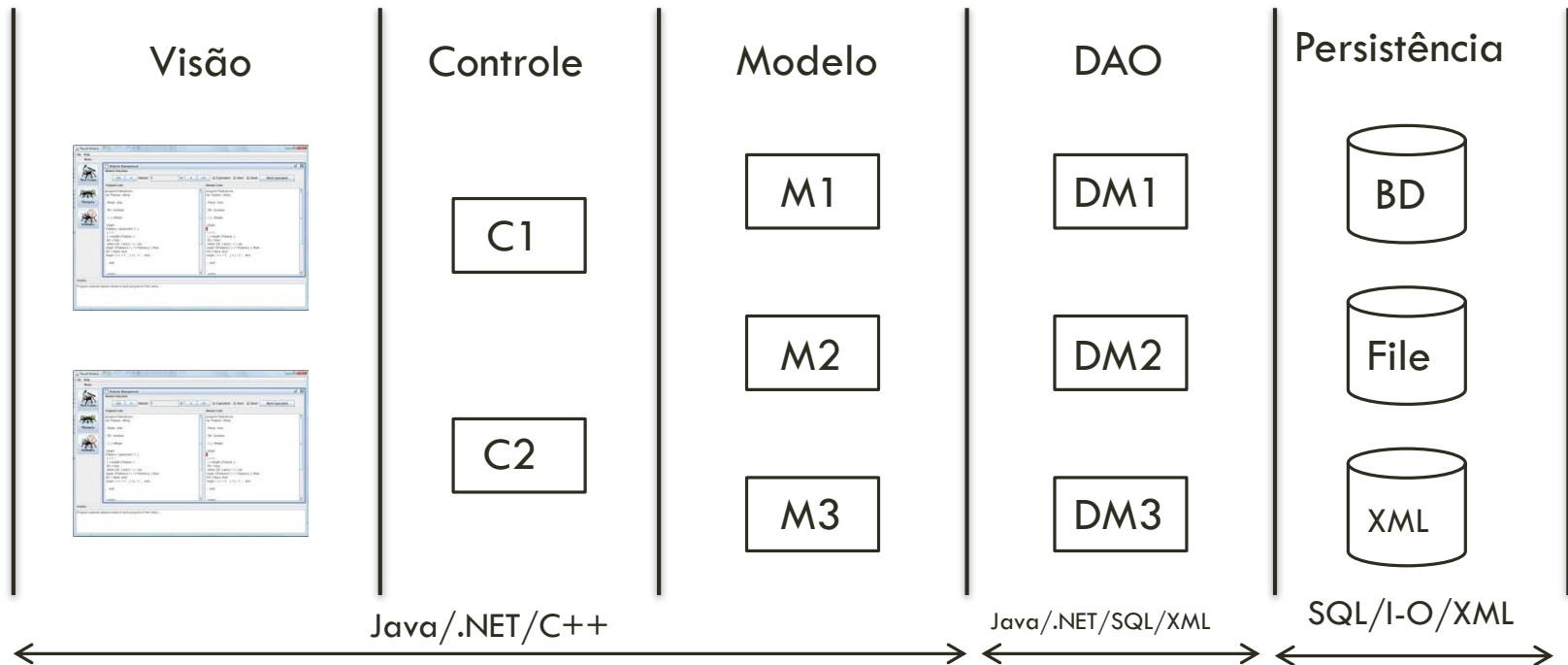
DATA ACCESS OBJECT (DAO) – CONTEXTO

- A maioria dos sistemas precisa persistir dados em algum ponto de seu funcionamento
- Dados de um sistema podem ser armazenados de diferentes formas
- O acesso ao banco de dados depende fortemente do tipo de armazenamento e da tecnologia utilizada
- A interação entre as regras de negócio e as regras de armazenamento de dados influencia na qualidade do sistema

DATA ACCESS OBJECT (DAO) – PROBLEMA



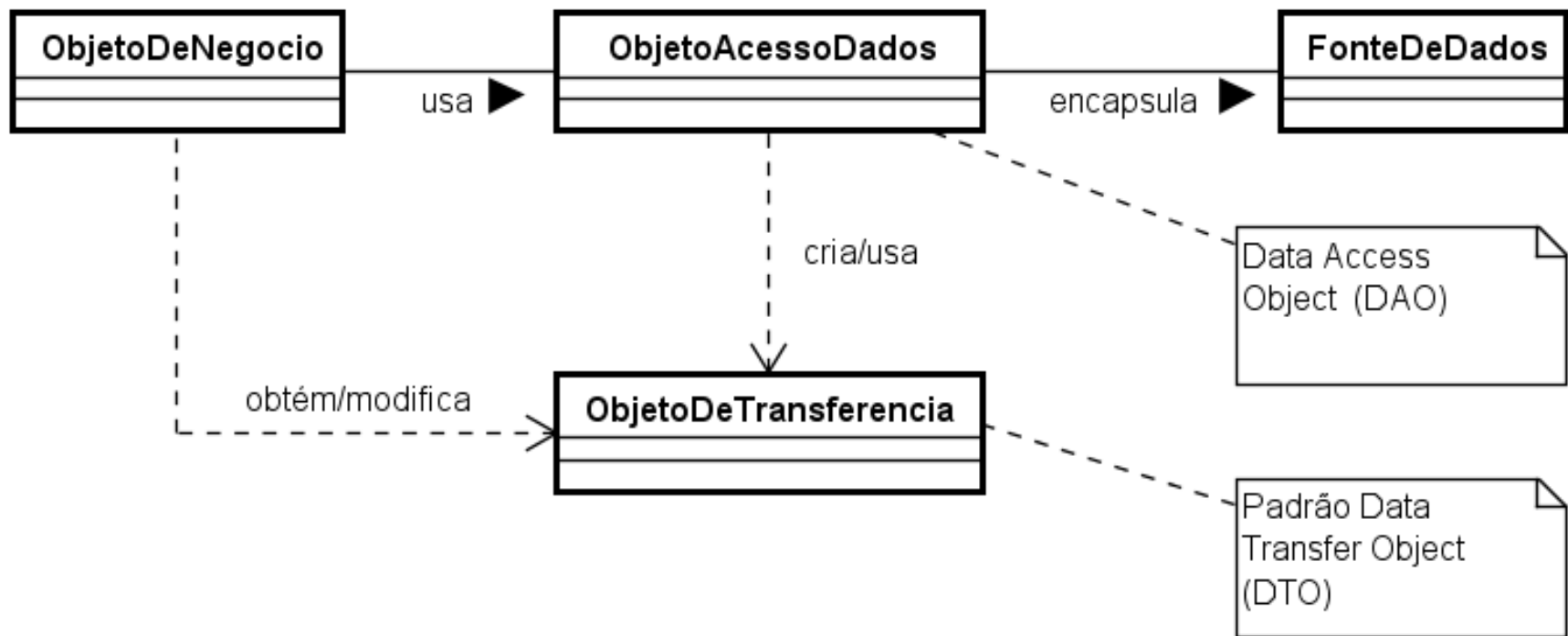
DATA ACCESS OBJECT (DAO) – SOLUÇÃO



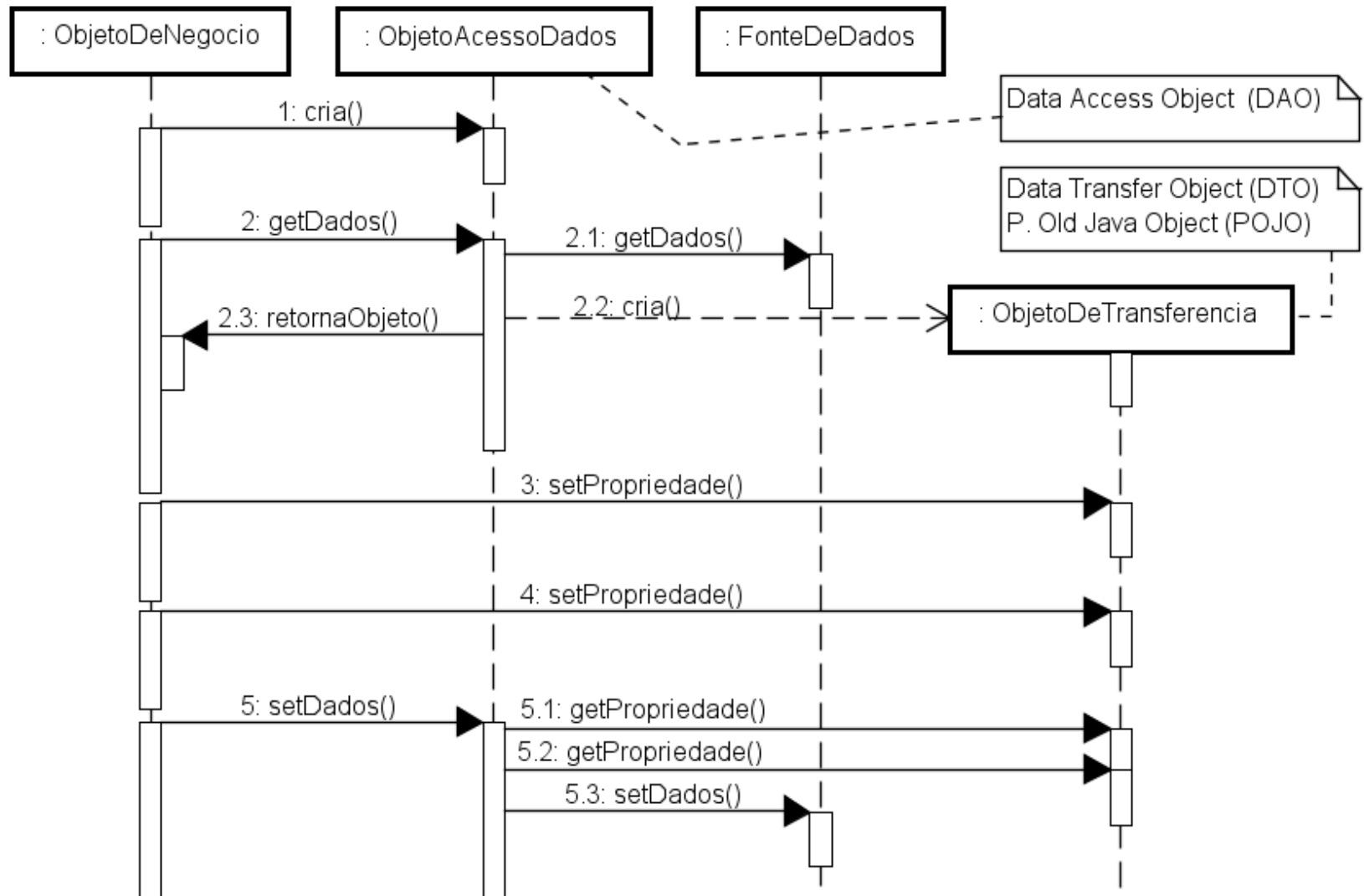
DATA ACCESS OBJECT (DAO) – SOLUÇÃO

- Criar um objeto para gerenciar a obtenção e o armazenamento de dados
- Abstrair e encapsular todo o acesso às fontes de dados
- Ocultar do cliente a forma de acesso por meio de uma interface
- Fornecer os dados do banco usando um objeto de transferência

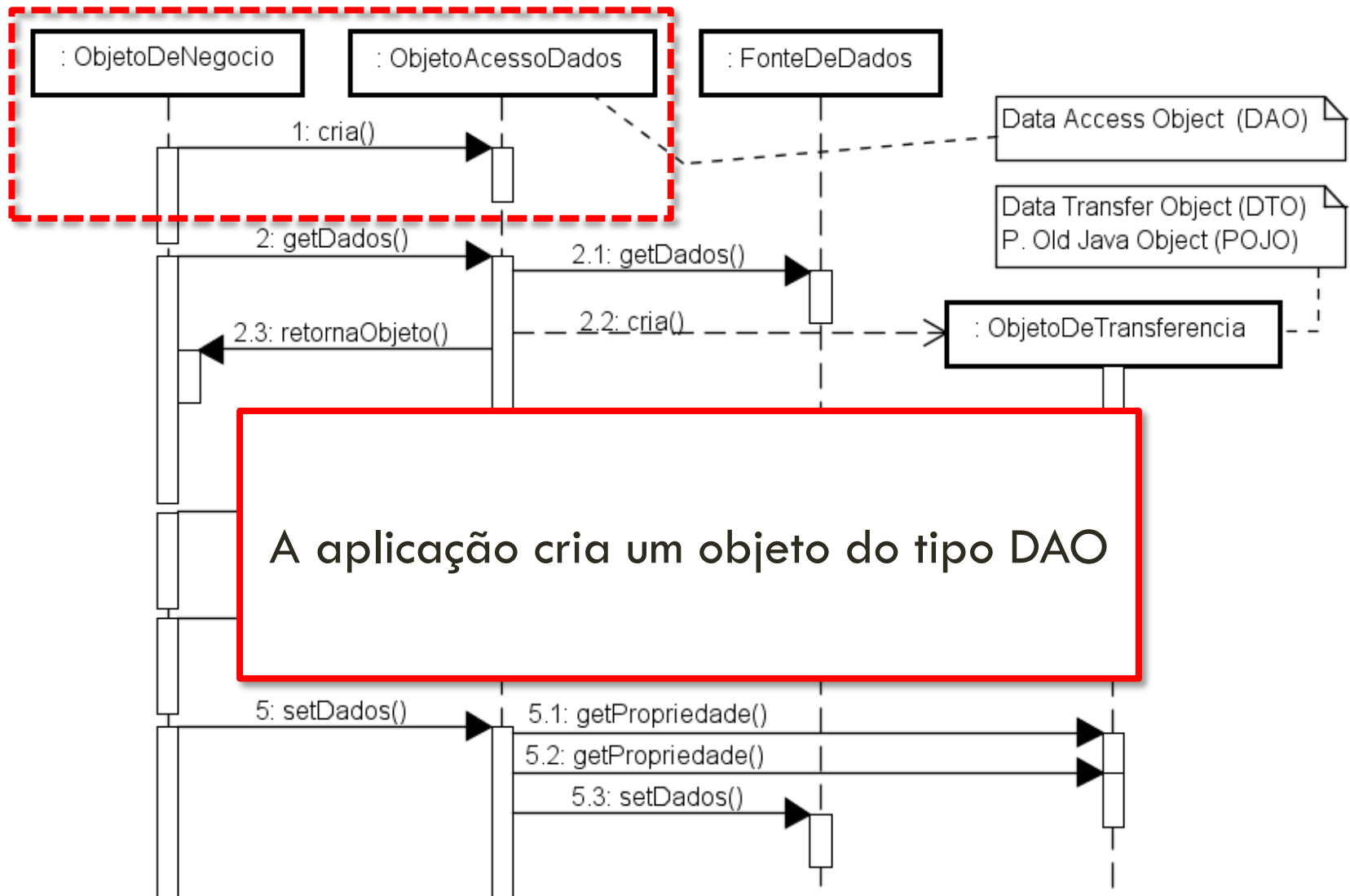
DATA ACCESS OBJECT (DAO) – SOLUÇÃO



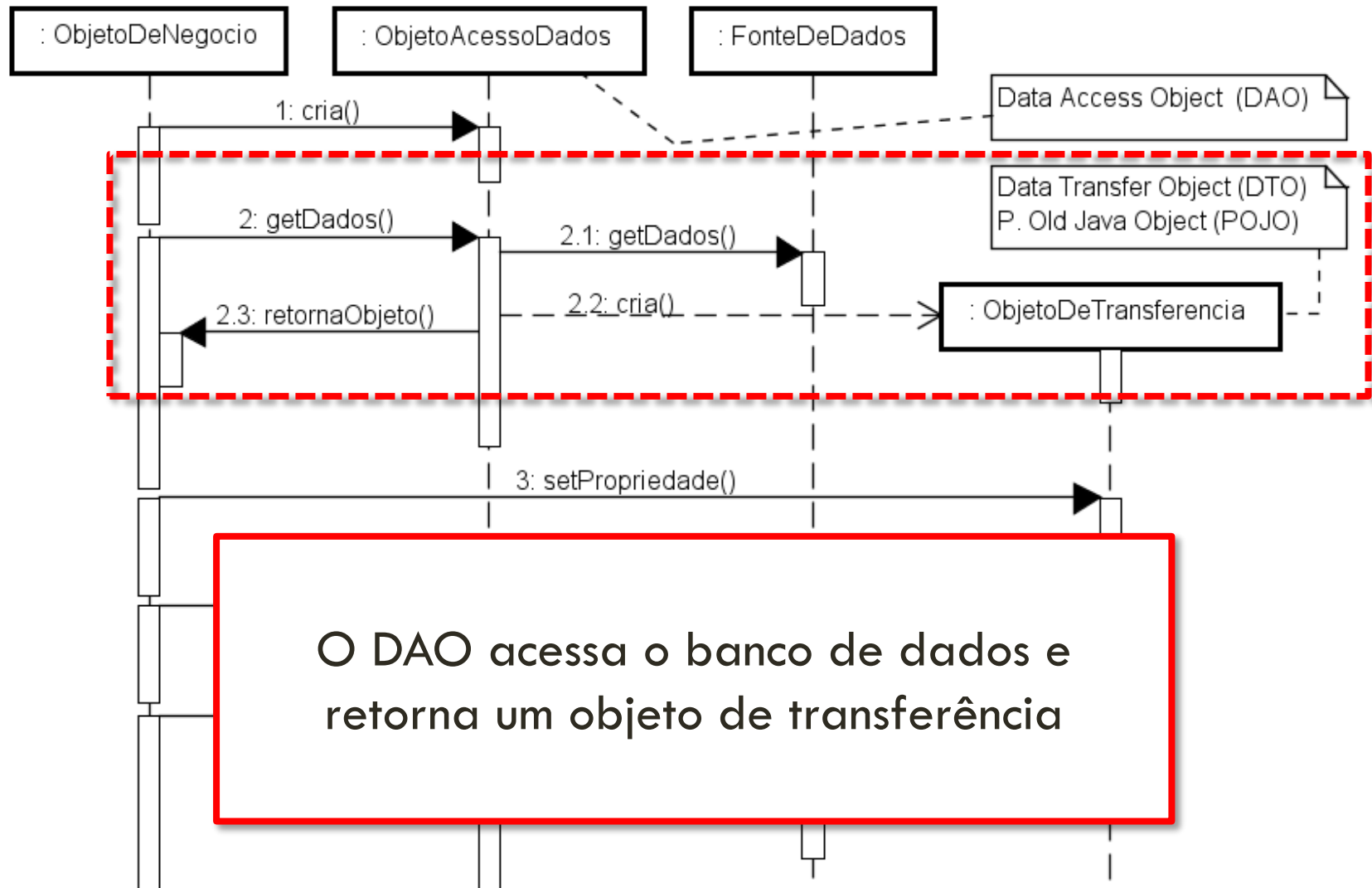
DATA ACCESS OBJECT (DAO) – SOLUÇÃO



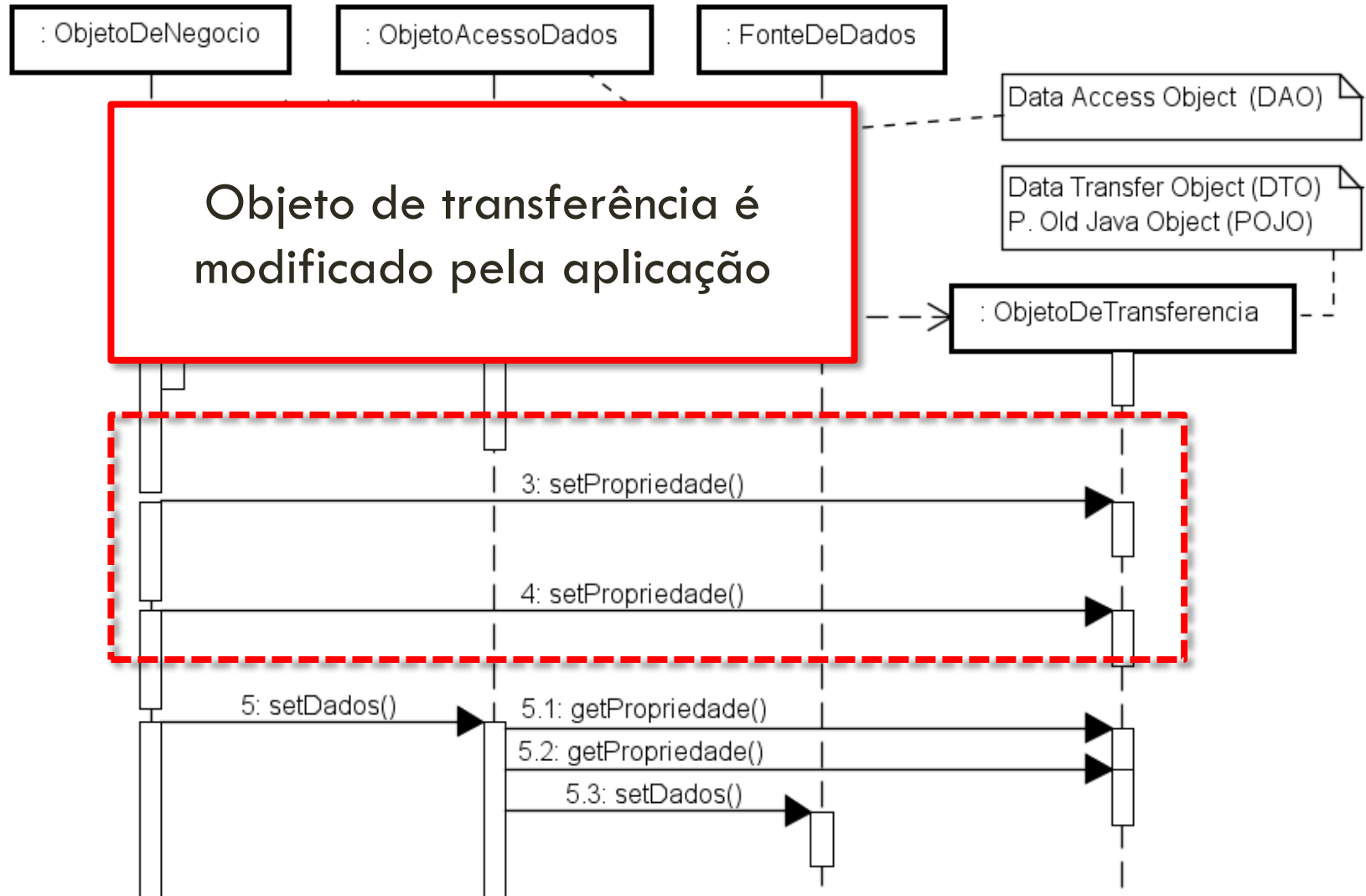
DATA ACCESS OBJECT (DAO) – SOLUÇÃO



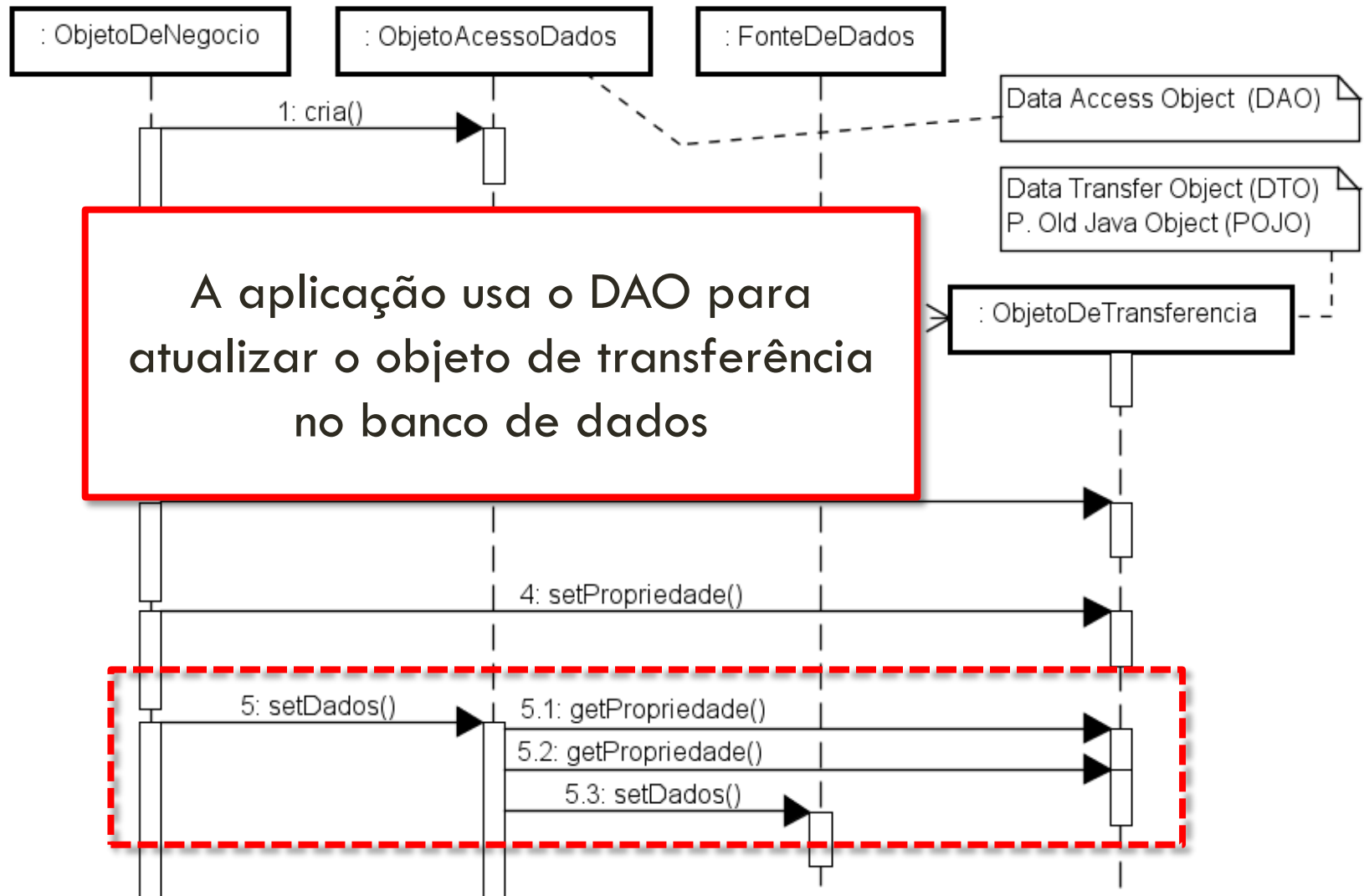
DATA ACCESS OBJECT (DAO) – SOLUÇÃO



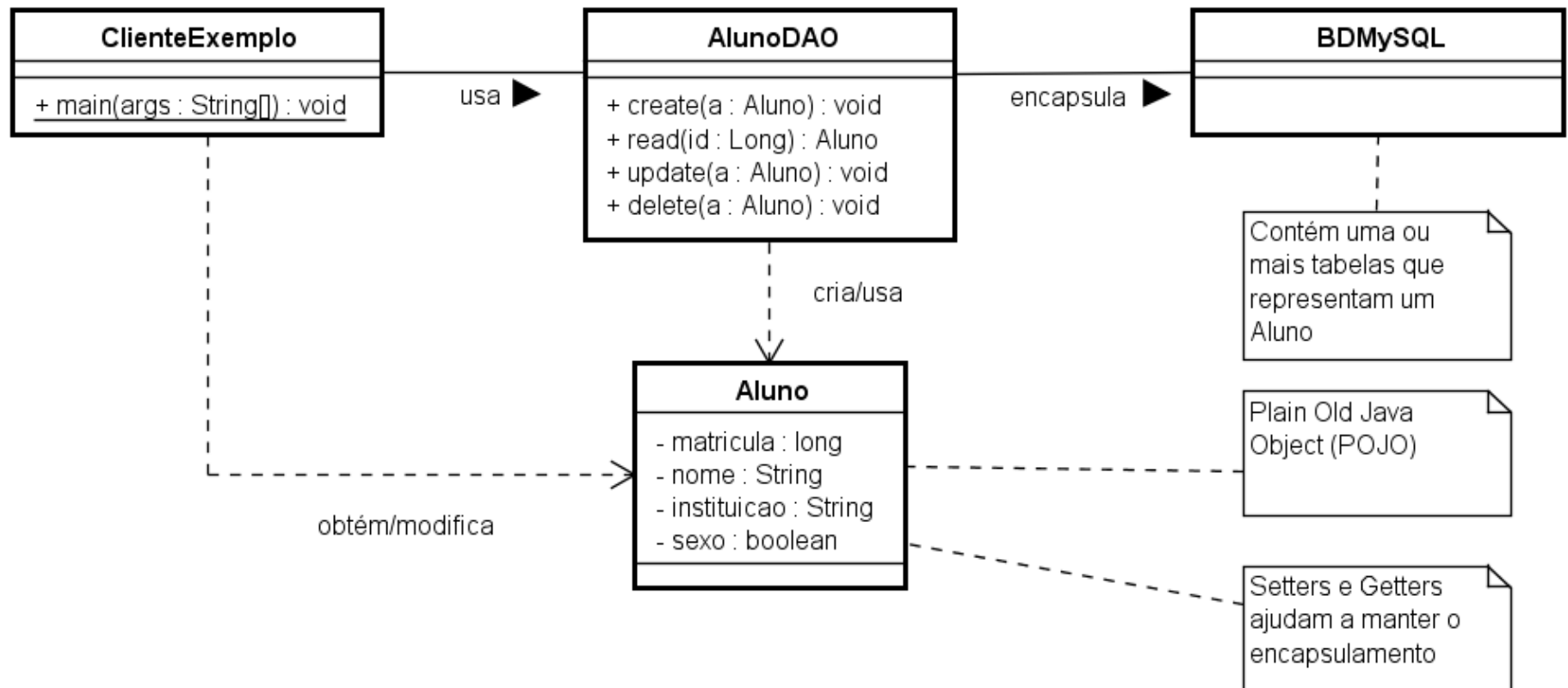
DATA ACCESS OBJECT (DAO) – SOLUÇÃO



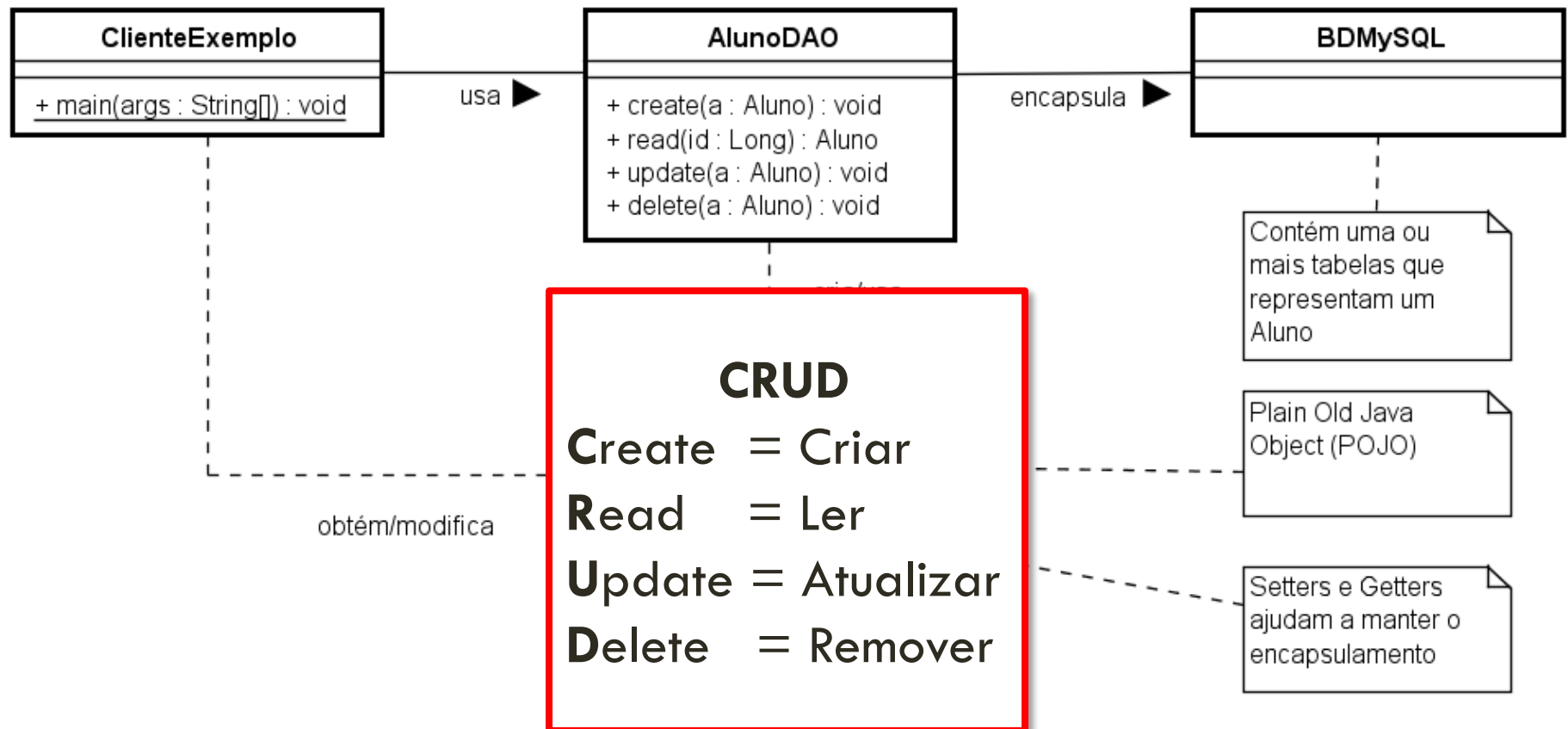
DATA ACCESS OBJECT (DAO) – SOLUÇÃO



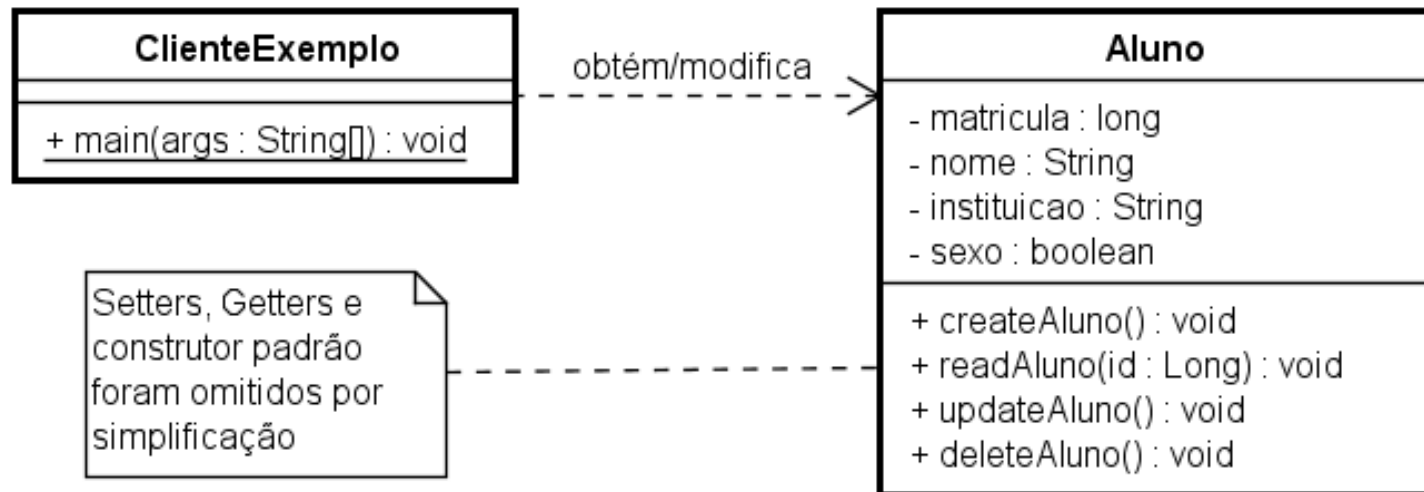
DATA ACCESS OBJECT (DAO) – EXEMPLO



DATA ACCESS OBJECT (DAO) – EXEMPLO



SEM DAO — EXEMPLO DE IMPLEMENTAÇÃO



SEM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
public class Aluno {

    private long matricula;
    private String nome, instituicao;
    private boolean sexo;
    public static final boolean MASCULINO = false;
    public static final boolean FEMININO = true;

    public Aluno() {}

    // Setters e Getters para manter o encapsulamento
    public long getMatricula() {return matricula;}
    public void setMatricula(long matr) {this.matricula = matr;}

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

    public String getInstituicao() {return instituicao;}
    public void setInstituicao(String inst) {this.instituicao = inst;}

    public boolean getSexo() {return sexo;}
    public void setSexo(boolean sexo) {this.sexo = sexo;}
```

Continua ...

SEM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
public class Aluno {  
  
    private long matricula;  
    private String nome, instituicao;  
    private boolean sexo;  
    public static final boolean MASCULINO = false;  
    public static final boolean FEMININO = true;  
  
    public Aluno() {}  
  
    // Setters e Getters para manter o encapsulamento  
    public long getMatricula() {return matricula;}  
    public void setMatricula(long matr) {this.matricula = matr;}  
  
    public String getNome() {return nome;}  
    public void setNome(String nome) {this.nome = nome;}  
  
    public String getInstituicao() {return instituicao;}  
    public void setInstituicao(String inst) {this.instituicao = inst;}  
  
    public boolean getSexo() {return sexo;}  
    public void setSexo(boolean sexo) {this.sexo = sexo;}  
}
```

Continua ...

SEM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
public class Aluno {  
  
    private long matricula;  
    private String nome, instituicao;  
    private boolean sexo;  
    public static final boolean MASCULINO = false;  
    public static final boolean FEMININO = true;  
  
    public Aluno() {}  
  
    // Setters e Getters para manter o encapsulamento  
    public long getMatricula() {return matricula;}  
    public void setMatricula(long matr) {this.matricula = matr;}  
  
    public String getNome() {return nome;}  
    public void setNome(String nome) {this.nome = nome;}  
  
    public String getInstituicao() {return instituicao;}  
    public void setInstituicao(String inst) {this.instituicao = inst;}  
  
    public boolean getSexo() {return sexo;}  
    public void setSexo(boolean sexo) {this.sexo = sexo;}  
}
```

Continua ...

SEM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
public class Aluno {  
  
    private long matricula;  
    private String nome, instituicao;  
    private boolean sexo;  
    public static final boolean MASCULINO = false;  
    public static final boolean FEMININO = true;  
  
    public Aluno() {}  
  
    // Setters e Getters para manter o encapsulamento  
    public long getMatricula() {return matricula;}  
    public void setMatricula(long matr) {this.matricula = matr;}  
  
    public String getNome() {return nome;}  
    public void setNome(String nome) {this.nome = nome;}  
  
    public String getInstituicao() {return instituicao;}  
    public void setInstituicao(String inst) {this.instituicao = inst;}  
  
    public boolean getSexo() {return sexo;}  
    public void setSexo(boolean sexo) {this.sexo = sexo;}  
}
```

Continua ...

SEM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
// Os métodos a seguir misturam as regras de negócio (Java)
// com as regras de bancos de dados (SQL), o que diminui a coesão
public void createAluno(){
    try {
        Class.forName("com.mysql.jdbc.Driver");
        Connection conexao =
            DriverManager.getConnection("jdbc:mysql://localhost/IFSP_DAO",
            "IFSP_DAO", "123123");

        // cria um preparedStatement baseado em uma string SQL
        String sql = "INSERT INTO aluno (id, nom, inst, gen) values (?, ?, ?, ?)";
        PreparedStatement stmt = conexao.prepareStatement(sql);

        // preenche os valores para (?, ?, ..., ?)
        stmt.setLong(1, matricula);
        stmt.setString(2, nome);
        stmt.setString(3, instituicao);
        stmt.setBoolean(4, sexo);

        // executa o comando SQL
        stmt.execute();
        stmt.close();
    }
```

Continua ...

SEM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
// Os métodos a seguir misturam as regras de negócio (Java)
// com as regras de bancos de dados (SQL), o que diminui a coesão
public void createAluno(){
    try {
        Class.forName("com.mysql.jdbc.Driver");
        Connection conexao =
            DriverManager.getConnection("jdbc:mysql://localhost/IFSP_DAO",
            "IFSP_DAO", "123123");

        // cria um preparedStatement baseado em uma string SQL
        String sql = "INSERT INTO aluno (id, nom, inst, gen) values (?, ?, ?, ?)";
        PreparedStatement stmt = conexao.prepareStatement(sql);

        // preenche os valores para (?, ?, ..., ?)
        stmt.setLong(1, matricula);
        stmt.setString(2, nome);
        stmt.setString(3, instituicao);
        stmt.setBoolean(4, sexo);

        // executa o comando SQL
        stmt.execute();
        stmt.close();
    }
}
```

Continua ...

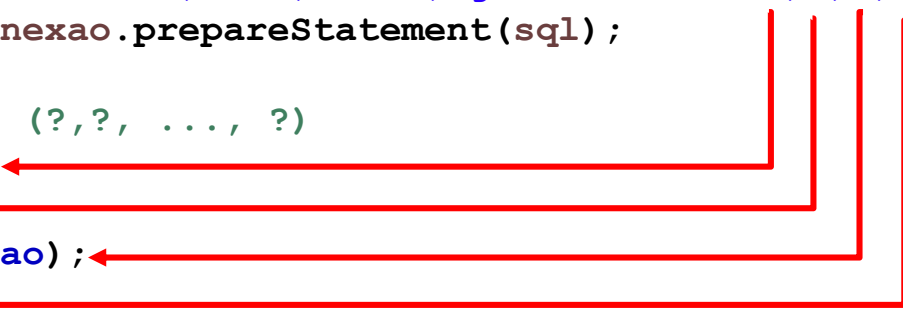
SEM DAO – EXEMPLO DE IMPLEMENTAÇÃO

```
// Os métodos a seguir misturam as regras de negócio (Java)
// com as regras de bancos de dados (SQL), o que diminui a coesão
public void createAluno(){
    try {
        Class.forName("com.mysql.jdbc.Driver");
        Connection conexao =
            DriverManager.getConnection("jdbc:mysql://localhost/IFSP_DAO",
                "IFSP_DAO", "123123");

        // cria um preparedStatement baseado em uma string SQL
        String sql = "INSERT INTO aluno (id, nom, inst, gen) values (?, ?, ?, ?)";
        PreparedStatement stmt = conexao.prepareStatement(sql);

        // preenche os valores para (?, ?, ..., ?)
        stmt.setLong(1, matricula);
        stmt.setString(2, nome);
        stmt.setString(3, instituicao);
        stmt.setBoolean(4, sexo);

        // executa o comando SQL
        stmt.execute();
        stmt.close();
    }
}
```



Continua ...

SEM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
// Os métodos a seguir misturam as regras de negócio (Java)
// com as regras de bancos de dados (SQL), o que diminui a coesão
public void createAluno(){
    try {
        Class.forName("com.mysql.jdbc.Driver");
        Connection conexao =
            DriverManager.getConnection("jdbc:mysql://localhost/IFSP_DAO",
            "IFSP_DAO", "123123");

        // cria um preparedStatement baseado em uma string SQL
        String sql = "INSERT INTO aluno (id, nom, inst, gen) values (?, ?, ?, ?)";
        PreparedStatement stmt = conexao.prepareStatement(sql);

        // preenche os valores para (?, ?, ..., ?)
        stmt.setLong(1, matricula);
        stmt.setString(2, nome);
        stmt.setString(3, instituicao);
        stmt.setBoolean(4, sexo);

        // executa o comando SQL
        stmt.execute();
        stmt.close();
```

Continua ...

SEM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
        System.out.println("O aluno " + nome + " foi gravado no banco de  
dados.");  
        conexao.close();  
        return true;  
  
    } catch (ClassNotFoundException e) {  
        System.err.println("Não foi possível encontrar a classe de conexão!");  
        e.printStackTrace();  
    } catch (SQLException e) {  
        System.err.println("Erro na comunicação com o banco de dados!");  
        e.printStackTrace();  
    }  
}
```

SEM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
        System.out.println("O aluno " + nome + " foi gravado no banco de  
dados.");  
        conexao.close();  
        return true;  
  
    } catch (ClassNotFoundException e) {  
        System.err.println("Não foi possível encontrar a classe de conexão!");  
        e.printStackTrace();  
    } catch (SQLException e) {  
        System.err.println("Erro na comunicação com o banco de dados!");  
        e.printStackTrace();  
    }  
}
```


SEM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
System.out.println("O aluno " + nome + " foi gravado no banco de  
dados.");  
conexao.close();  
return true;  
  
} catch (ClassNotFoundException e) {  
    System.err.println("Não foi possível encontrar a classe de conexão!");  
    e.printStackTrace();  
} catch (SQLException e) {  
    System.err.println("Erro na comunicação com o banco de dados!");  
    e.printStackTrace();  
}  
}
```

SEM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
public Aluno readAluno(long matr){
    try {
        Class.forName("com.mysql.jdbc.Driver");
        Connection conexao =
            DriverManager.getConnection("jdbc:mysql://localhost/IFSP_DAO",
            "IFSP_DAO", "123123");

        // Busca o aluno pela matrícula
        String sql = "SELECT * FROM aluno WHERE id = ?" ;
        PreparedStatement stmt = conexao.prepareStatement(sql);
        stmt.setLong(1, matr);

        ResultSet resultado = stmt.executeQuery();
        if(resultado.next()){
            Aluno a = new Aluno(); // Cria o objeto de transferência
            a.setNome(resultado.getString("nom"));
            a.setMatricula(resultado.getInt("id"));
            a.setInstituicao(resultado.getString("inst"));
            a.setSexo(resultado.getBoolean("gen"));
            System.out.println("O aluno " + a.getNome() + ", matrícula n. " +
            a.getMatricula() + ", estuda no " + a.getInstituicao() + ".");
            return a; // retorna o objeto de transferência
        }
    }
}
```

Continua ...

SEM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
public Aluno readAluno(long matr){
    try {
        Class.forName("com.mysql.jdbc.Driver");
        Connection conexao =
            DriverManager.getConnection("jdbc:mysql://localhost/IFSP_DAO",
            "IFSP_DAO", "123123");

        // Busca o aluno pela matrícula
        String sql = "SELECT * FROM aluno WHERE id = ?" ;
        PreparedStatement stmt = conexao.prepareStatement(sql);
        stmt.setLong(1, matr);

        ResultSet resultado = stmt.executeQuery();
        if(resultado.next()){
            Aluno a = new Aluno(); // Cria o objeto de transferência
            a.setNome(resultado.getString("nom"));
            a.setMatricula(resultado.getInt("id"));
            a.setInstituicao(resultado.getString("inst"));
            a.setSexo(resultado.getBoolean("gen"));
            System.out.println("O aluno " + a.getNome() + ", matrícula n. " +
            a.getMatricula() + ", estuda no " + a.getInstituicao() + ".");
            return a; // retorna o objeto de transferência
        }
    }
}
```

Continua ...

SEM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
public Aluno readAluno(long matr){
    try {
        Class.forName("com.mysql.jdbc.Driver");
        Connection conexao =
            DriverManager.getConnection("jdbc:mysql://localhost/IFSP_DAO",
            "IFSP_DAO", "123123");

        // Busca o aluno pela matrícula
        String sql = "SELECT * FROM aluno WHERE id = ?" ;
        PreparedStatement stmt = conexao.prepareStatement(sql);
        stmt.setLong(1, matr);

        ResultSet resultado = stmt.executeQuery();
        if(resultado.next()){
            Aluno a = new Aluno(); // Cria o objeto de transferência
            a.setNome(resultado.getString("nom"));
            a.setMatricula(resultado.getInt("id"));
            a.setInstituicao(resultado.getString("inst"));
            a.setSexo(resultado.getBoolean("gen"));
            System.out.println("O aluno " + a.getNome() + ", matrícula n. " +
            a.getMatricula() + ", estuda no " + a.getInstituicao() + ".");
            return a; // retorna o objeto de transferência
        }
    }
}
```

Continua ...

SEM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
    }else{
        System.out.println("Matrícula não encontrada!");
    }
    conexao.close();
    return null;
} catch (ClassNotFoundException e) {
    ...
} catch (SQLException e) {
    ...
}
}
```

SEM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
    }else{  
        System.out.println("Matrícula não encontrada!");  
    }  
    conexao.close();  
    return null;  
} catch (ClassNotFoundException e) {  
    ...  
} catch (SQLException e) {  
    ...  
}  
}
```

SEM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
public boolean updateAluno() {
    try {
        Class.forName("com.mysql.jdbc.Driver");
        Connection conexao =
            DriverManager.getConnection("jdbc:mysql://localhost/IFSP_DAO",
            "IFSP_DAO", "123123");

        // cria um preparedStatement baseado em uma string SQL
        String sql = "UPDATE aluno SET nom = ?, inst = ?, gen = ? WHERE id = ?";
        PreparedStatement stmt = conexao.prepareStatement(sql);

        // preenche os valores para (?, ?, ..., ?)
        stmt.setString(1, nome);
        stmt.setString(2, instituicao);
        stmt.setBoolean(3, sexo);
        stmt.setLong(4, matricula);
        stmt.executeUpdate();
        stmt.close();

        System.out.println("O aluno " + nome + " foi atualizado no BD.");
        conexao.close();
    } // adiciona os métodos catch
}
```

Continua ...

SEM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
public boolean deleteAluno() {
    try {
        Class.forName("com.mysql.jdbc.Driver");
        Connection conexao =
            DriverManager.getConnection("jdbc:mysql://localhost/IFSP_DAO",
            "IFSP_DAO", "123123");

        // Remove o aluno
        String sql = "DELETE FROM aluno WHERE id = ?" ;
        PreparedStatement stmt = conexao.prepareStatement(sql);
        stmt = conexao.prepareStatement(sql);
        stmt.setLong(1, matricula);
        stmt.execute();

        stmt.close();
        conexao.close();

        System.out.println("O aluno " + nome + " foi removido do BD.");

    } // adiciona os métodos catch
    } // fim do método deleteAluno
} // fim da classe Aluno
```


SEM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
public boolean deleteAluno() {  
    try {  
        Class.forName("com.mysql.jdbc.Driver");  
        Connection conexao =  
            DriverManager.getConnection("jdbc:mysql://localhost/IFSP_DAO",  
"IFSP_DAO", "123123");  
  
        // Remove o aluno  
        String sql = "DELETE FROM aluno";  
        PreparedStatement stmt = conexao.prepareStatement(sql);  
        stmt.setLong(1, matricula);  
        stmt.execute();  
  
        stmt.close();  
        conexao.close();  
  
        System.out.println("O aluno " + matricula + " foi  
removido do BD.");  
  
    } // adiciona os métodos catch  
    } // fim do método deleteAluno  
} // fim da classe Aluno
```

Arrg! Deve ter um
jeito mais fácil!



SEM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
public class ClienteExemplo {  
  
    public static void main(String[] args) {  
  
        //Cria um aluno  
        Aluno jorge = new Aluno();  
        jorge.setMatricula(101010);  
        jorge.setInstituicao("IFSP São Carlos");  
        jorge.setSexo(Aluno.MASCULINO);  
        jorge.setNome("Jorge");  
  
        //Salva o aluno no banco de dados  
        jorge.createAluno();  
  
        //Atualiza as informações do aluno  
        jorge.setNome("Jorge Fernandes");  
        jorge.updateAluno();  
  
        //Busca as informações cadastradas no banco de dados  
        jorge.readAluno(jorge.getMatricula());  
    }  
}
```

SEM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
public class ClienteExemplo {  
  
    public static void main(String[] args) {  
  
        //Cria um aluno  
        Aluno jorge = new Aluno();  
        jorge.setMatricula(101010);  
        jorge.setInstituicao("IFSP São Carlos");  
        jorge.setSexo(Aluno.MASCULINO);  
        jorge.setNome("Jorge");  
  
        //Salva o aluno no banco de dados  
        jorge.createAluno();  
  
        //Atualiza as informações do aluno  
        jorge.setNome("Jorge Fernandes");  
        jorge.updateAluno();  
  
        //Busca as informações cadastradas no banco de dados  
        jorge.readAluno(jorge.getMatricula());  
    }  
}
```



Continua ...

SEM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
//Remove o aluno
jorge.deleteAluno();

//Verifica se as informações foram mesmo removidas
jorge.readAluno(jorge.getMatricula());
}
}
```

SEM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
//Remove o aluno  
jorge.deleteAluno();
```

```
//Verifica se as informações foram mesmo removidas  
jorge.readAluno(jorge.getMatricula());
```

```
}
```

```
}
```

Ops! Um aluno
responsável por
ler outro aluno?



SEM DAO — EXEMPLO DE IMPLEMENTAÇÃO

| Aluno |
|---|
| <ul style="list-style-type: none">- matricula : long- nome : String- instituicao : String- sexo : boolean |
| <ul style="list-style-type: none">+ createAlunoMySql() : void+ readAlunoMySql() : void+ updateAlunoMySql() : void+ deleteAlunoMySql() : void |



SEM DAO — EXEMPLO DE IMPLEMENTAÇÃO

| Aluno |
|--|
| - matricula : long - nome : String - instituicao : String - sexo : boolean |
| + createAlunoMySQL() : void + readAlunoMySQL() : void + updateAlunoMySQL() : void + deleteAlunoMySQL() : void + createAlunoPostgre() : void + readAlunoPostgre() : void + updateAlunoPostgre() : void + deleteAlunoPostgre() : void |

- **coesão**
- + **complexidade**
- **Modularidade**



SEM DAO — EXEMPLO DE IMPLEMENTAÇÃO

| Aluno |
|---|
| <ul style="list-style-type: none">- matricula : long- nome : String- instituicao : String- sexo : boolean |
| <ul style="list-style-type: none">+ createAlunoMySQL() : void+ readAlunoMySQL() : void+ updateAlunoMySQL() : void+ deleteAlunoMySQL() : void+ createAlunoPostgre() : void+ readAlunoPostgre() : void+ updateAlunoPostgre() : void+ deleteAlunoPostgre() : void+ createAlunoDB2() : void+ readAlunoDB2() : void+ updateAlunoDB2() : void+ deleteAlunoDB2() : void+ createAlunoFirebird() : void+ readAlunoFirebird() : void+ updateAlunoFirebird() : void+ deleteAlunoFirebird() : void |

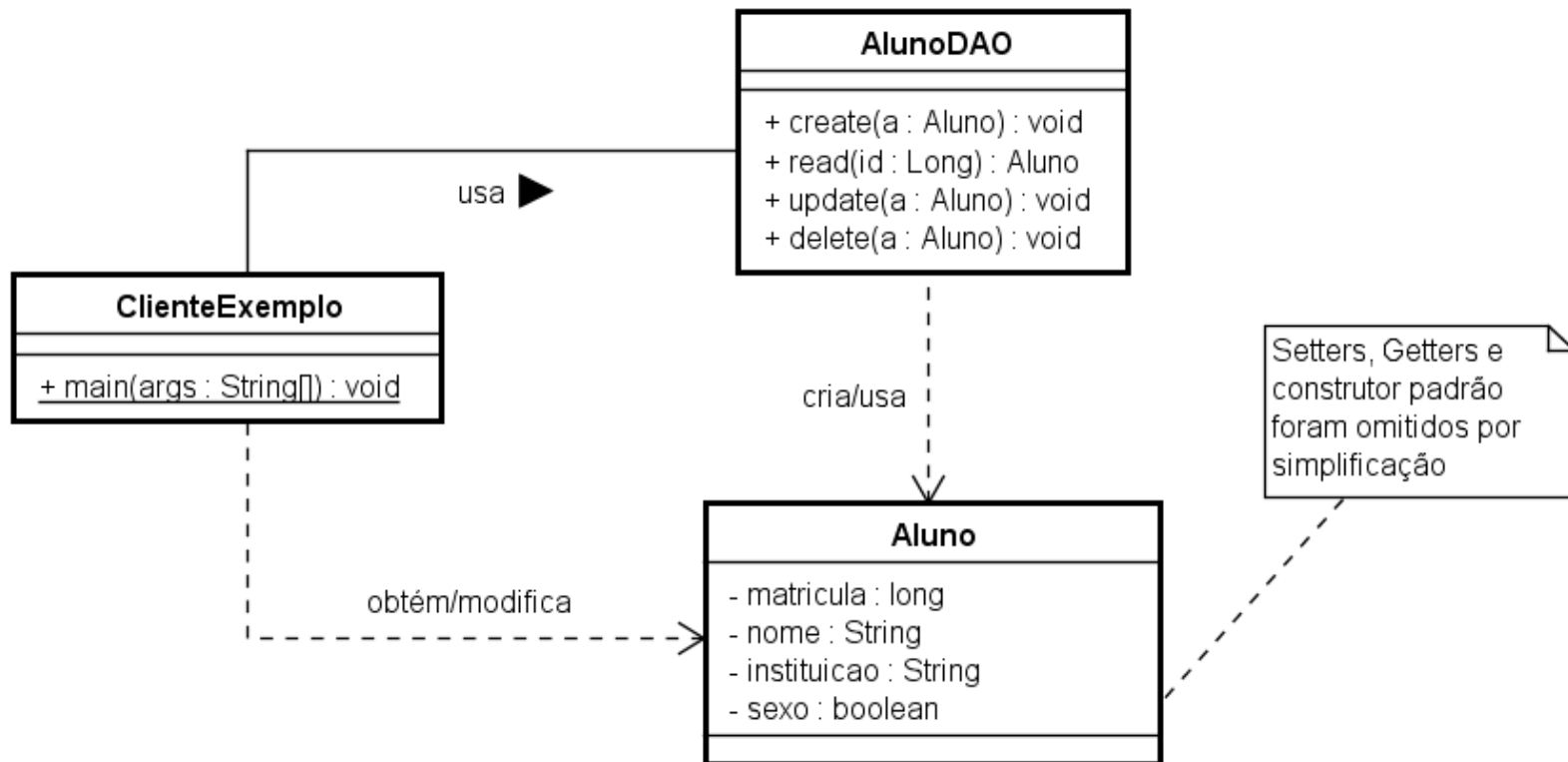
--- coesão

+++ complexidade

--- Modularidade



COM DAO – EXEMPLO DE IMPLEMENTAÇÃO



COM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
public class Aluno {  
  
    private long matricula;  
    private String nome;  
    private String instituicao;  
    private boolean sexo;  
  
    public static final boolean MASCULINO = false;  
    public static final boolean FEMININO = true;  
  
    public Aluno() {}  
  
    // Setters e Getters para manter o encapsulamento  
    public long getMatricula() {return matricula;}  
    public void setMatricula(long matr) {this.matricula = matr;}  
  
    public String getNome() {return nome;}  
    public void setNome(String nome) {this.nome = nome;}  
  
    // Continua a definição dos Setters e Getters ...  
}  
}
```

COM DAO – EXEMPLO DE IMPLEMENTAÇÃO

```
public class ConnectionFactory {  
  
    public ConnectionFactory() {  
    }  
  
    public Connection createConnection() throws SQLException {  
        try {  
            Class.forName("com.mysql.jdbc.Driver");  
            return DriverManager.getConnection(  
                "jdbc:mysql://localhost/IFSP_DAO", "IFSP_DAO", "123123");  
        } catch (ClassNotFoundException e) {  
            System.err.println("Não foi possível encontrar a classe de  
conexão!");  
            e.printStackTrace();  
            return null;  
        }  
    }  
}
```



Com o padrão Método Fábrica
facilitamos a criação de conexões
com o banco de dados!

COM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
public class AlunoDAO{

    public void create(Aluno a){
        try {
            // utiliza a fábrica de conexões para criar uma Connection Sql
            ConnectionFactory fabricaCon = new ConnectionFactory();
            Connection conexao = fabricaCon.createConnection();

            // cria um preparedStatement baseado em uma string SQL
            String sql = "INSERT INTO aluno (id, nom, inst, gen) values (?, ?, ?, ?)";
            PreparedStatement stmt = conexao.prepareStatement(sql);

            // preenche os valores para (?, ?, ..., ?)
            stmt.setLong(1, a.getMatricula());
            stmt.setString(2, a.getNome());
            stmt.setString(3, a.getInstituicao());
            stmt.setBoolean(4, a.getSexo());

            // executa o comando SQL
            stmt.execute();
            stmt.close();
        }
    }
}
```



Continua ...

COM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
System.out.println("O aluno " + a.getNome() + " foi gravado BD.");  
conexao.close();
```

```
} catch (SQLException e) {  
    System.err.println("Erro na comunicação com o banco de dados!");  
    e.printStackTrace();  
}
```

```
}
```



Continua ...

COM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
public Aluno read(long matr){
    try {
        // Utiliza a fábrica de conexões para criar uma Connection Sql
        ConnectionFactory fabricaCon = new ConnectionFactory();
        Connection conexao = fabricaCon.createConnection();

        // Busca o aluno pela matrícula
        String sql = "SELECT * FROM aluno WHERE id = ?" ;
        PreparedStatement stmt = conexao.prepareStatement(sql);
        stmt.setLong(1, matr);

        ResultSet resultado = stmt.executeQuery();
        if(resultado.next()){
            Aluno a = new Aluno();
            a.setNome(resultado.getString("nom"));
            a.setMatricula(resultado.getInt("id"));
            a.setInstituicao(resultado.getString("inst"));
            a.setSexo(resultado.getBoolean("gen"));
            System.out.println("O aluno " + a.getNome() + ", matrícula n. " +
a.getMatricula() + ", estuda no " + a.getInstituicao() + ".");
            return a;
        }
    }
}
```

Continua ...

COM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
else{
    System.out.println("Matrícula não encontrada!");
}
conexao.close();
return null;
} catch (SQLException e) {
    System.err.println("Erro na comunicação com o banco de dados!");
    e.printStackTrace();
    return null;
}
}
```

COM DAO – EXEMPLO DE IMPLEMENTAÇÃO

```
public void update(Aluno a) {
    try {
        // utiliza a fábrica de conexões para criar uma Connection Sql
        ConnectionFactory fabricaCon = new ConnectionFactory();
        Connection conexao = fabricaCon.createConnection();

        // cria um preparedStatement baseado em uma string SQL
        String sql = "UPDATE aluno SET nom = ?, inst = ?, gen = ? WHERE id = ?";
        PreparedStatement stmt = conexao.prepareStatement(sql);

        // preenche os valores para (?, ?, ..., ?)
        stmt.setString(1, a.getNome());
        stmt.setString(2, a.getInstituicao());
        stmt.setBoolean(3, a.getSexo());
        stmt.setLong(4, a.getMatricula());

        stmt.executeUpdate();
        stmt.close();
        conexao.close();

        System.out.println("O aluno " + a.getNome() + " foi atualizado no BD");
    } catch (SQLException e) {...} } // fim do método update
```

Continua ...

COM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
public void delete(Aluno a) {
    try {
        // utiliza a fábrica de conexões para criar uma Connection Sql
        ConnectionFactory fabricaCon = new ConnectionFactory();
        Connection conexao = fabricaCon.createConnection();

        // Remove o aluno
        String sql = "DELETE FROM aluno WHERE id = ?" ;
        PreparedStatement stmt = conexao.prepareStatement(sql);
        stmt = conexao.prepareStatement(sql);
        stmt.setLong(1, a.getMatricula());
        stmt.execute();

        stmt.close();
        System.out.println("O aluno " + a.getNome() + " foi removido do BD.");
        conexao.close();

    } catch (SQLException e) {...}
} // fim do método delete
} // fim da classe AlunoDAO
```

COM DAO – EXEMPLO DE IMPLEMENTAÇÃO

```
public class ClienteExemplo {  
  
    public static void main(String[] args) {  
  
        //Cria um aluno  
        Aluno jorge = new Aluno();  
        jorge.setMatricula(101010);  
        jorge.setInstituicao("IFSP São Carlos");  
        jorge.setSexo(Aluno.MASCULINO);  
        jorge.setNome("Jorge");  
  
        //Cria o DAO para conexão com o banco de dados  
        AlunoDAO alunoDAO = new AlunoDAO();  
  
        //Salva o aluno no banco de dados;  
        alunoDAO.create(jorge);  
  
        //Atualiza as informações do aluno  
        jorge.setNome("Jorge Fernandes"); // gerencia a aplicação  
        alunoDAO.update(jorge);           //gerencia o banco de dados  
    }  
}
```

COM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
public class ClienteExemplo {  
  
    public static void main(String[] args) {  
  
        //Cria um aluno  
        Aluno jorge = new Aluno();  
        jorge.setMatricula(101010);  
        jorge.setInstituicao("IFSP São Carlos");  
        jorge.setSexo(Aluno.MASCULINO);  
        jorge.setNome("Jorge");  
  
        //Cria o DAO para conexão com o banco de dados  
        AlunoDAO alunoDAO = new AlunoDAO();  
  
        //Salva o aluno no banco de dados;  
        alunoDAO.create(jorge);  
  
        //Atualiza as informações do aluno  
        jorge.setNome("Jorge Fernandes"); // gerencia a aplicação  
        alunoDAO.update(jorge);           //gerencia o banco de dados  
    }  
}
```



COM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
//Busca as informações cadastradas no banco de dados
alunoDAO.read(jorge.getMatricula());

//Remove o aluno
alunoDAO.delete(jorge);

//Verifica se as informações foram mesmo removidas
alunoDAO.read(jorge.getMatricula());
}
}
```

COM DAO — EXEMPLO DE IMPLEMENTAÇÃO

```
//Busca as informações cadastradas no banco de dados  
alunoDAO.read(jorge.getMatricula());
```

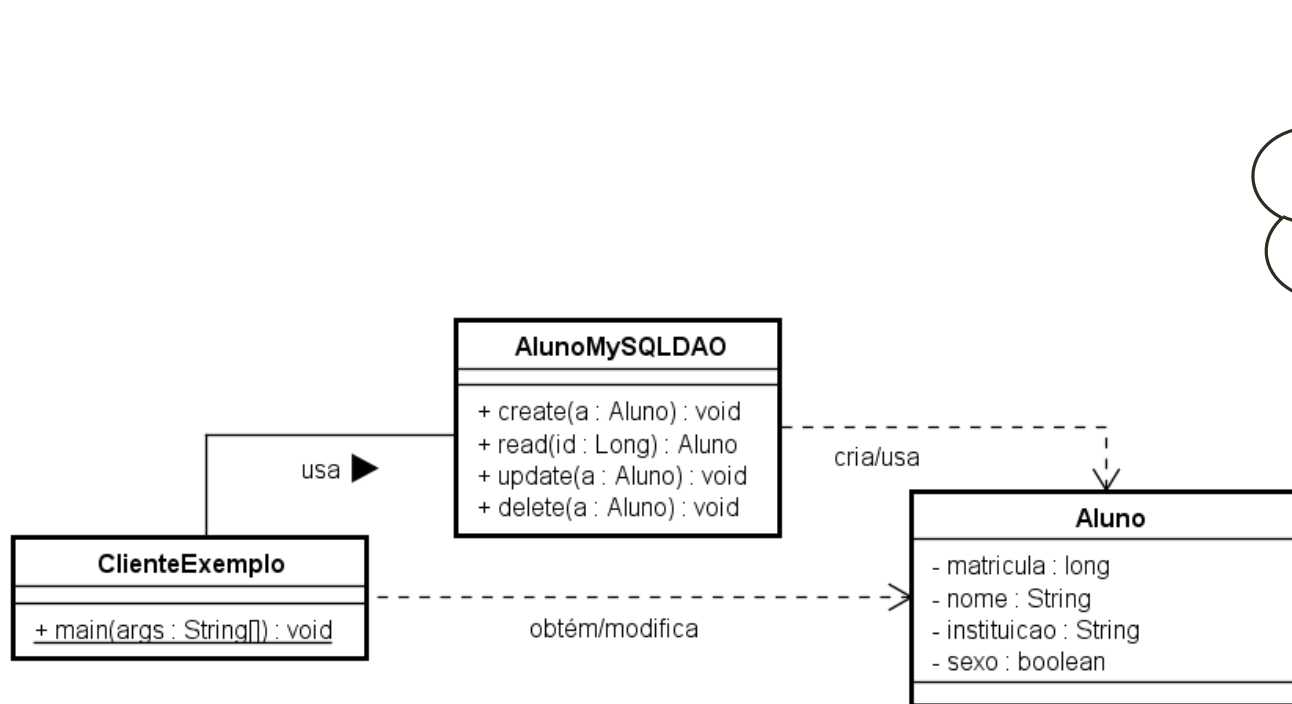
```
//Remove o aluno  
alunoDAO.delete(jorge);
```

```
//Verifica se as informações foram mesmo removidas  
alunoDAO.read(jorge.getMatricula());
```

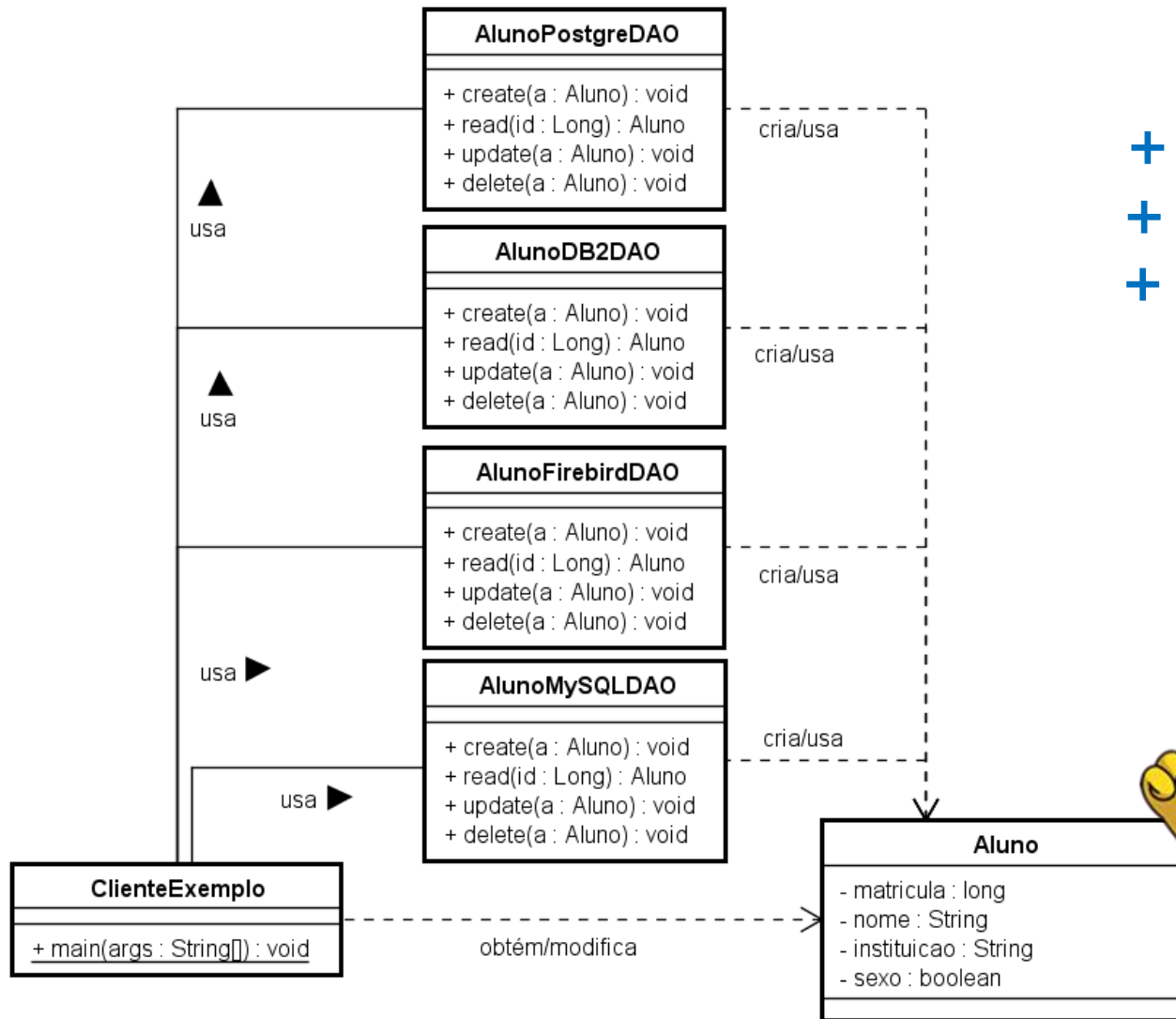
```
}
```



COM DAO — EXEMPLO DE IMPLEMENTAÇÃO



COM DAO — EXEMPLO DE IMPLEMENTAÇÃO



+++ coesão

+++ modularidade

+++ reusabilidade



RESUMO

- **MVC:** Organizar o sistema de forma mais coesa, separando a representação gráfica, o controle e as regras de negócio
- **DAO:** Encapsular o acesso ao banco de dados separando as regras de persistência das regras de negócio
 - O acesso ao banco pode variar sem que a aplicação seja alterada
 - A aplicação pode ter múltiplas formas de acesso aos dados de forma mais organizada
 - Maior flexibilidade, coesão, modularidade e facilidade de manutenção

EXERCÍCIOS

1. Criar uma classe *Instituicao* com uma relação “um para muitos” com *Aluno*. Desenvolver uma classe *InstituicaoDAO* para persistir objetos da classe *Instituicao*.
2. Estender a estrutura do Exercício 1 utilizando o padrão Fábrica Abstrata para fabricar objetos do tipo DAO.
3. Implementar classes DAO para gravar objetos do tipo *Aluno* e *Instituicao* em arquivos de texto. Criar também a fábrica para construir os objetos do tipo DAO.