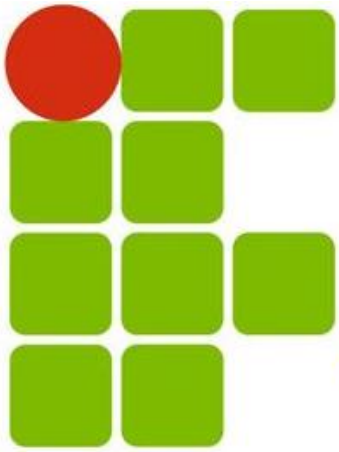


POO2: DAO

Generic Types

```
public class Box {
    private Object object;

    public void set(Object object) { this.object = object; }
    public Object get() { return object; }
}
```

Como os métodos dessa classe retornam ou aceitam o tipo Object, você é livre para passar qualquer tipo de objeto

```java
public class Box<T> {
   private T t;

   public void add(T t) {
      this.t = t;
   }

   public T get() {
      return t;
   }

   public static void main(String[] args) {
      Box<Integer> integerBox = new Box<Integer>();
      Box<String> stringBox = new Box<String>();

      integerBox.add(new Integer(10));
      stringBox.add(new String("Hello World"));

      System.out.printf("Integer Value :%d\n\n", integerBox.get());
      System.out.printf("String Value :%s\n", stringBox.get());
   }
}
```

Usando o Generic Types

Data Access Object

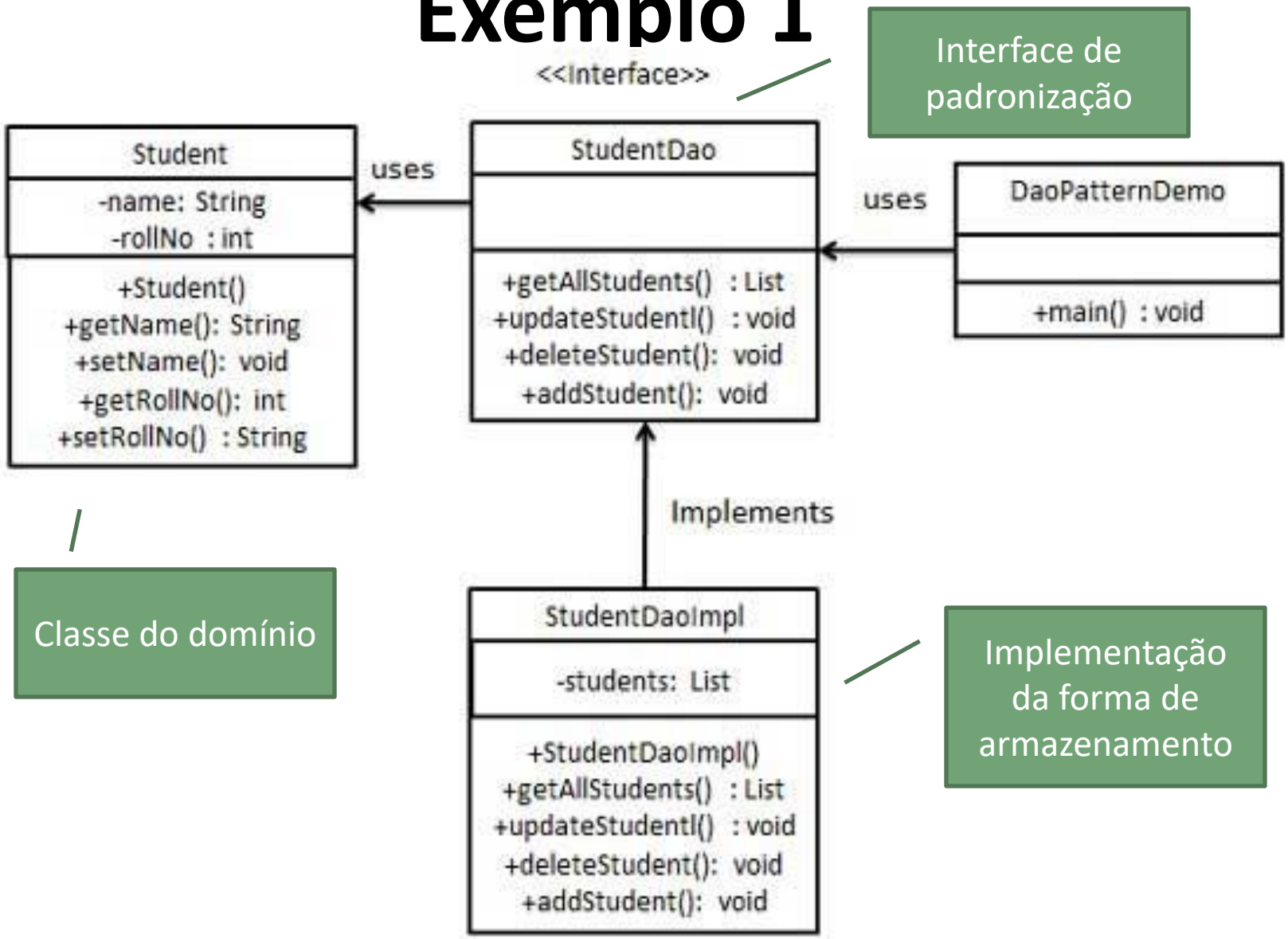
Padrão separa operações de acesso aos dados da aplicação

- **Data Access Object Interface** – Define operações padrões a ser implementada em uma classe de acesso a dados concreta

- **Data Access Object concrete class** – Responsável por acesso o mecanismo de armazenamento de dados

- **Model Object or Value Object** – Objetos do modelo que contém informações que serão armazenadas

Exemplo 1



ESPIRITO SANTO

Model Object or Value Object

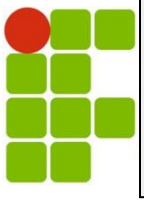
```java
public class Student {
   private String name;
   private int rollNo;

   Student(String name, int rollNo){
      this.name = name;
      this.rollNo = rollNo;
   }

   public String getName() {
      return name;
   }

   public void setName(String name) {
      this.name = name;
   }

   public int getRollNo() {
      return rollNo;
   }

   public void setRollNo(int rollNo) {
      this.rollNo = rollNo;
   }
}
```

Data Access Object Interface

```
import java.util.List;

public interface StudentDao {
   public List<Student> getAllStudents();
   public Student getStudent(int rollNo);
   public void updateStudent(Student student);
   public void deleteStudent(Student student);
}
```

Data Access Object concrete class

```
public class StudentDaoImpl implements StudentDao {

   //list is working as a database
   List<Student> students;

   public StudentDaoImpl(){
      students = new ArrayList<Student>();
      Student student1 = new Student("Robert",0);
      Student student2 = new Student("John",1);
      students.add(student1);
      students.add(student2);
   }
   @Override
   public void deleteStudent(Student student) {
      students.remove(student.getRollNo());
      System.out.println("Student: Roll No " + student.getRollNo() + ", deleted from 
   }

   //retrive list of students from the database
   @Override
   public List<Student> getAllStudents() {
      return students;
   }

   @Override
   public Student getStudent(int rollNo) {
      return students.get(rollNo);
   }
```



Main

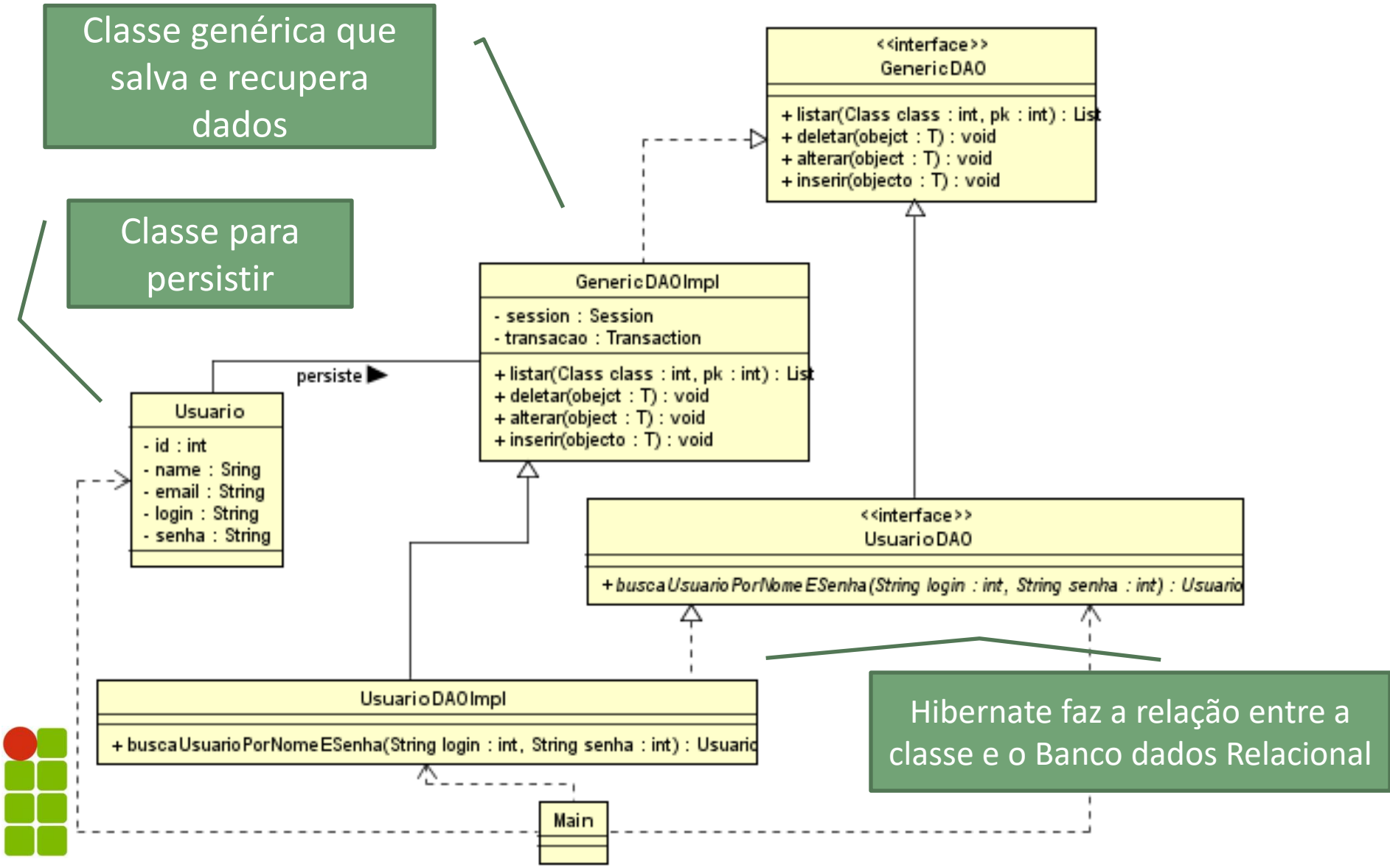
```
public class DaoPatternDemo {
   public static void main(String[] args) {
      StudentDao studentDao = new StudentDaoImpl();

      //print all students
      for (Student student : studentDao.getAllStudents()) {
         System.out.println("Student: [RollNo : " + student.getRollNo() + ", Name : "
      }


      //update student
      Student student =studentDao.getAllStudents().get(0);
      student.setName("Michael");
      studentDao.updateStudent(student);

      //get the student
      studentDao.getStudent(0);
      System.out.println("Student: [RollNo : " + student.getRollNo() + ", Name : " +
   }
}
```

Vantagens do Hibernate

Utilizando o Hibernate não preciso de:

- Um sql para inserir cada classe do meu domínio/tabela;
- Um sql para atualizar cada classe do meu domínio/tabela;
- Um sql para listar cada classe do meu domínio/tabela;
- Uma sql para criar cada tabela do meu domínio;

Hibernate e Classes



Classe Main

```
UsuarioDAO usuarioDAO = new UsuarioDAOImpl();
Usuario usuario = new Usuario();
usuario.setEmail("felipefo@gmail.com");
usuario.setLogin("felipefo");
usuario.setSenha("123456");
usuarioDAO.inserir(usuario);
List<Usuario> lista = usuarioDAO.listar(Usuario.class);
for (Usuario usuarioRecuparado : lista) {
   System.out.println("Id: " + usuarioRecuparado.getId() + " | Login:"
           + usuarioRecuparado.getLogin()+
   " | Senha " + usuarioRecuparado.getSenha());
   usuarioDAO.deletar(usuario);
}
lista = usuarioDAO.listar(Usuario.class);
for (Usuario usuarioRecuparado : lista) {
    System.out.println("Id: " + usuarioRecuparado.getId() + " | Login:"
            + usuarioRecuparado.getLogin()+
    " | Senha " + usuarioRecuparado.getSenha());
}
```

Objeto que vai salvar as informações

Objeto que quero salvar

Inserindo o objeto na tabela Usuario!

Listando usuários do banco

Removendo usuário do banco

Interface GenericDAO

```
public interface GenericDAO<T> {

    public void inserir( T obj ) throws Exception;
    public void alterar( T obj ) throws Exception;
    public void deletar( T obj ) throws Exception;
    public List<T> listar(Class clazz) throws Exception;
    public T listar(Class clazz, String pk) throws Exception;
    public void rollBack();
}
```

Inserir com Hibernate

```
public abstract class GenericDAOImpl<T> implements Gene
{
    protected static Session sessao;
    protected Transaction transacao;
    /** Creates a new instance of GenericDAO */

    public void inserir( T obj ) throws Exception
    {
            sessao = HibernateUtil.getSession();
            transacao = sessao.beginTransaction();
            sessao.save(obj);
            sessao.flush();
            transacao.commit();
            sessao.close();
    }

    public void alterar( T obj ) t
    {
            sessao = HibernateUtil.getSession();
```

Classes do hibernate que fazem a relação entre o banco e a classe JAVA

Objeto que quero salvar

Criando a instancia do hibernate

Pegando uma transação para comitar

Salvando as informações

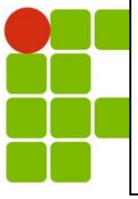
Consulta no banco com hibernate

```
public List<T> listar(Class clazz) throws Exception
{
    sessao = HibernateUtil.getSession();
    transacao = sessao.beginTransaction();
    List objts;
    objts = null;
    Criteria selectAll = sessao.createCriteria(clazz);
    transacao.commit();
    objts = selectAll.list();
    sessao.close();
    return objts;
}
```

Remoção no banco de dados

```
public void deletar( T obj ) throws Exception
{
        sessao = HibernateUtil.getSession();
        transacao = sessao.beginTransaction();
        sessao.delete(obj);
        sessao.flush();
        transacao.commit();
        sessao.close();
}
```

```java
@Entity
@Table(name = "usuario")
public class Usuario {

    private Integer id;
    private String name;
    private String email;
    private String login;
    private String senha;
    private boolean admin=false;


    public Usuario() {

    }

    public Usuario(Integer id, String name, String email, int idade) {
        this.id = id;
        this.name = name;
        this.email = email;

    }
```



UsuarioDAOImpl

```
public class UsuarioDAOImpl extends GenericDAOImpl<Usuario> implements UsuarioDAO {

    public Usuario buscaUsuarioPorNomeESenha(String login, String senha){
        sessao = HibernateUtil.getSession();
        transacao = sessao.beginTransaction();
        List lista = sessao.createQuery("from Usuario where login = '"+ login +
                "' and senha ='" + senha +"'").list();
        sessao.flush();
        transacao.commit();
        sessao.close();
        if(lista.isEmpty())
            return null;
        return (Usuario)lista.get(0);
    }
    public Usuario buscaUsuarioExiste(String login){
         sessao = HibernateUtil.getSession();
```

Como ele faz isso?

<Reflection API>
Java

Como ele faz isso?

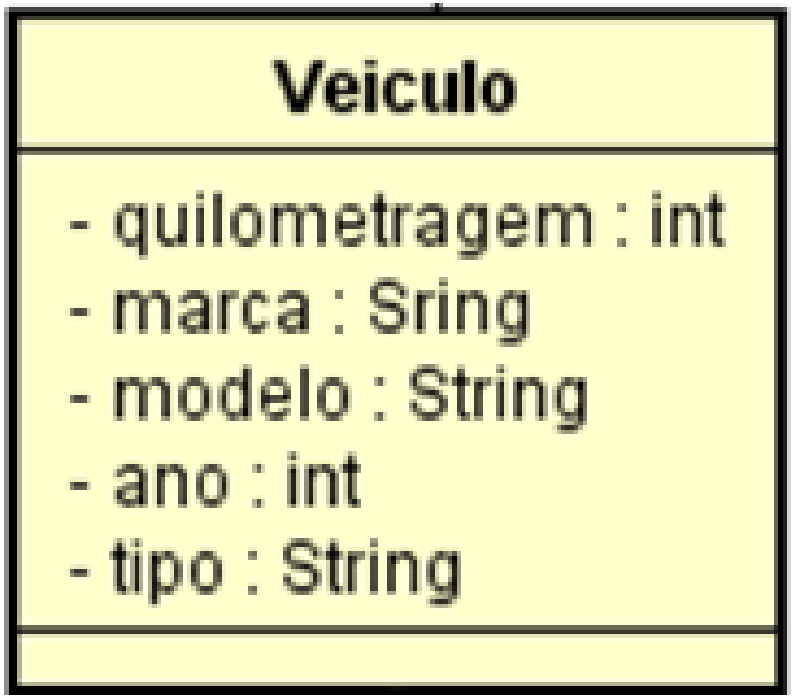
```
public void inserir( T obj ) {

      sessao = HibernateUtil.getSession();
      sessao.save(obj);
       ……….
}
```

Método Inserir

Como declarei como Template posso passar qualquer tipo de objeto

```
public static void main(String[] args) {
    VeiculoDAO veiculoDAO = new VeiculoDAOImpl();
    Veiculo novoVeiculo = new Veiculo();
    veiculoDAO.inserir(veiculo);

}
```

Classe Main


Veiculo
- quilometragem : int
- marca : Sring
- modelo : String
- ano : int
- tipo : String

Java Reflection

```java
/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
 Veiculo n = new Veiculo();
 // chamamos o getClass de Object
 Class<Veiculo> classe = (Class<Veiculo>) n.getClass();
 Field field[] = classe.getDeclaredFields();
 for(int i=0;i< field.length;i++){
     System.out.println("Field :" + field[i]);
 }

 Method metodos[] = classe.getMethods();
 for(int i=0;i< metodos.length;i++){
     System.out.println("método:" + metodos[i]);
 }
```

Class é a forma genêrica de pegar os metadados de uma classe

Pegando atributos

Pegando métodos

Resultado do código anterior Refleciton

Tipo do Campo

Nome do Campo

```
Field :int example.hibernate.persistencia.Veiculo.id
Field :java.lang.String example.hibernate.persistencia.Veiculo.marca
Field :java.lang.String example.hibernate.persistencia.Veiculo.modelo
Field :java.lang.String example.hibernate.persistencia.Veiculo.tipoVeiculo
Field :int example.hibernate.persistencia.Veiculo.ano
Field :int example.hibernate.persistencia.Veiculo.quilometragem
método:public int example.hibernate.persistencia.Veiculo.getId()
método:public void example.hibernate.persistencia.Veiculo.setModelo(java.lang.String)
método:public int example.hibernate.persistencia.Veiculo.getQuilometragem()
método:public java.lang.String example.hibernate.persistencia.Veiculo.getModelo()
método:public void example.hibernate.persistencia.Veiculo.setTipoVeiculo(java.lang.String)
método:public void example.hibernate.persistencia.Veiculo.setMarca(java.lang.String)
método:public java.lang.String example.hibernate.persistencia.Veiculo.getTipoVeiculo()
método:public void example.hibernate.persistencia.Veiculo.setQuilometragem(int)
método:public void example.hibernate.persistencia.Veiculo.setAno(int)
método:public java.lang.String example.hibernate.persistencia.Veiculo.getMarca()
método:public void example.hibernate.persistencia.Veiculo.setId(int)
método:public int example.hibernate.persistencia.Veiculo.getAno()
```

Mapeamento Objeto Relacional

```
Class.forName("org.sqlite.JDBC");
c = DriverManager.getConnection("jdbc:sqlite:v
System.out.println("Base de dados aberta");
stmt = c.createStatement();
String sql = "CREATE TABLE VEICULO "
        + "(ID INT PRIMARY KEY  NOT NULL," /
           + " QUILOMETRAGEM   INT  NULL, "
        + " MARCA            CHAR(40)     NOT NULL, "
        + " MODELO         CHAR(50) NULL, "
        + " ANO          INT NOT NULL, "
        + " TIPO_VEICULO NOT NULL )";
```

Nome do Campo

Tipo do Campo

INSTITUTO FEDERAL
ESPIRITO SANTO

Configuração

Diz ao hibernate que essa classe é ser uma tabela no banco

```
@Entity
@Table(name = "Veiculo")
public class Veiculo {
    int id;
    String marca;
    String modelo;
    String tipoVeiculo;
    int ano;
    int quilometragem;
    @Id
    @GeneratedValue(generator = "increment")
    @GenericGenerator(name = "increment", strategy = "increment")
    public int getId() {
        return id;
    }
```

Classe auxiliar para facilitar o uso do Hibernate

```
public final class HibernateUtil {

    private static SessionFactory sessionFactory;
    private static ServiceRegistry serviceRegistry = null;

    private static SessionFactory getSessionFactory() throws HibernateException {
        Configuration configuration = new Configuration();
        configuration.configure();
        Properties properties = configuration.getProperties();
        if(serviceRegistry == null){
            serviceRegistry = new ServiceRegistryBuilder().applySettings(properties).buildServiceRegistry();
        }
        if(sessionFactory == null){
            sessionFactory = configuration.buildSessionFactory(serviceRegistry);
        }
        return sessionFactory;
    }

    public static Session getSession() {
        return getSessionFactory().openSession();
    }
}
```

Configuração do hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
        <session-factory>
                <property name="show_sql">true</property>
                <property name="format_sql">true</property>
                <property name="dialect">org.hibernate.dialect.SQLiteDialect</property>
                <property name="connection.driver_class">org.sqlite.JDBC</property>
                <property name="connection.url">jdbc:sqlite:mydb.db</property>
                <property name="connection.username"></property>
                <property name="connection.password"></property>
                <property name="hibernate.hbm2ddl.auto">update</property>
                <mapping class="example.hibernate.persistencia.Usuario"/>
        </session-factory>
</hibernate-configuration>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
        <session-factory>
                <property name="show_sql">true</property>
                <property name="format_sql">true</property>
                <property name="dialect">org.hibernate.dialect.SQLiteDialect</property>
                <property name="connection.driver_class">org.sqlite.JDBC</property>
                <property name="connection.url">jdbc:sqlite:mydb.db</property>
                <property name="connection.username"></property>
                <property name="connection.password"></property>
                <property name="hibernate.hbm2ddl.auto">update</property>
                <mapping class="example.hibernate.persistencia.Usuario"/>
                <mapping class="example.hibernate.persistencia.Veiculo"/>
        </session-factory>
</hibernate-configuration>
```

Preciso adicionar a classe Veiculo para que o hibernate saiba quais classes devem virar tabelas

Referencias

- http://www.tutorialspoint.com/design_pattern/mvc_pattern.htm
- http://www.tutorialspoint.com/design_pattern/data_access_object_pattern.htm
- http://www.wrox.com/WileyCDA/WroxTitle/Professional-Java-EE-Design-Patterns.productCd-111884341X,descCd-DOWNLOAD.html
- Professional JAVA EE Design Patterns, Murat Yener, Alex Threedom