```java
package webapplication.datalayer;

import java.util.List;

/**
 * @author Gustav
 *
 */
public class MySQLProduktBatchKomponentDAO implements ProduktBatchKompDAO {
    SQLMapper map = new SQLMapper();

    public MySQLProduktBatchKomponentDAO(){
        try { new Connector(); }
        catch (InstantiationException e) { e.printStackTrace(); }
        catch (IllegalAccessException e) { e.printStackTrace(); }
        catch (ClassNotFoundException e) { e.printStackTrace(); }
        catch (SQLException e) { e.printStackTrace(); }
    }

    @Override
    public ProduktBatchKompDTO getProduktBatchKomp(int pbId, int rbId) throws DALException
{

        String statement = map.getStatement("pb_komponent_SELECT");
        String[] values = new String[]{Integer.toString(pbId), Integer.toString(rbId)};
        statement = map.insertValuesIntoString(statement, values);
        System.out.println("Query: "+statement);
        ResultSet rs = Connector.doQuery(statement);

        try {
            if (!rs.first()) throw new DALException("Produkt batch komponent med pbId = " +
pbId +" Eller rbId = " + rbId + " findes ikke");

            return new ProduktBatchKompDTO (rs.getInt("pb_id"), rs.getInt("rb_id"),
rs.getDouble("tara"), rs.getDouble("netto"), rs.getInt("opr_id"));

        }
        catch (SQLException e) {throw new DALException(e); }
    }

    @Override
    public List<ProduktBatchKompDTO> getProduktBatchKompList(int pbId) throws DALException
{

        List<ProduktBatchKompDTO> list = new ArrayList<ProduktBatchKompDTO>();
        String statement = map.getStatement("pb_komponent_SELECT_ALL_rec_id");
        statement = map.insertValuesIntoString(statement, new
String[]{Integer.toString(pbId)});
        ResultSet rs = Connector.doQuery(statement);

        try
        {
            while (rs.next())
            {
                list.add(new ProduktBatchKompDTO (rs.getInt("pb_id"), rs.getInt("rb_id"),
rs.getDouble("tara"), rs.getDouble("netto"), rs.getInt("opr_id")));
            }
        }
        catch (SQLException e) { throw new DALException(e); }
        return list;
    }

```

```
66      @Override
67      public List<ProduktBatchKompDTO> getProduktBatchKompList() throws DALException {
68
69          List<ProduktBatchKompDTO> list = new ArrayList<ProduktBatchKompDTO>();
70          String statement = map.getStatement("pb_komponent_SELECT_ALL");
71          ResultSet rs = Connector.doQuery(statement);
72
73          try
74          {
75              while (rs.next())
76              {
77                  list.add(new ProduktBatchKompDTO (rs.getInt("pb_id"), rs.getInt("rb_id"),
    rs.getDouble("tara"), rs.getDouble("netto"), rs.getInt("opr_id")));
78              }
79          }
80          catch (SQLException e) { throw new DALException(e); }
81          return list;
82      }
83
84
85      @Override
86      public void createProduktBatchKomp(ProduktBatchKompDTO produktbatchkomponent) throws
    DALException {
87
88          String statement = map.getStatement("pb_komponent_INSERT");
89          String[] values = new String[]{
90                                      Integer.toString(produktbatchkomponent.getPbId()),
91                                      Integer.toString(produktbatchkomponent.getRbId()),
92                                      Double.toString(produktbatchkomponent.getTara()),
93                                      Double.toString(produktbatchkomponent.getNetto()),
94                                      Integer.toString(produktbatchkomponent.getOprId())
95                          };
96          statement = map.insertValuesIntoString(statement, values);
97          System.out.println(statement);
98          Connector.doUpdate(statement);
99
100     }
101
102     @Override
103     public void updateProduktBatchKomp(ProduktBatchKompDTO produktbatchkomponent) throws
    DALException {
104
105         String statement = map.getStatement("pb_komponent_UPDATE");
106         String[] values = new String[]{
107                                     Double.toString(produktbatchkomponent.getTara()),
108                                     Double.toString(produktbatchkomponent.getNetto()),
109                                     Integer.toString(produktbatchkomponent.getOprId()),
110                                     Integer.toString(produktbatchkomponent.getPbId()),
111                                     Integer.toString(produktbatchkomponent.getRbId())
112                         };
113         statement = map.insertValuesIntoString(statement, values);
114         System.out.println(statement);
115         Connector.doUpdate(statement);
116
117     }
118
119 }
120
```