|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **□ 수행평가 - 빅데이터를 활용한 IoT 시스템 개발(feat.커넥티드카)** | | | | | | |
|  |  |  | |  | |  |
| **과정명** | | 빅데이터를 활용한 IoT 시스템 개발(feat.커넥티드카) | | | | |
| **교과목명** | | IoT 운영시스템 구축 기반기술 | | **훈련교사** | | 이진만 |
| **과정명** | | SW기초기술이해  응용SW기술이해  Database 구문이해 | | | | |
| **수행날짜** | | 2019.06.10 | 훈련생명 | | 조민경 | |
| **과제개요** | | | | | | |
| 1. Workshop 교재 124Page의 테이블을 구축 한다. 2. DML, DDL을 작성하고 Table의 구조를 작성한다. 3. UML을 이용하여 프로그램을 설계 한다. 4. JDBC API를 이용하여 상품에 대한 CRUD 프로그램을 작성 한다.   작성 내용   1. ERD 2. DDL, DML 3. UML 4. 작성 코드 | | | | | | |
| **1. ERD**  \\M506ins\공유\수행평가\2.IoT 운영시스템구축기반기술\3조\ERD.PNG | | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **2-1. DDL**   |  |  |  | | --- | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | CREATE TABLE T\_FACTORY(  FACTNO VARCHAR2(5),  FACNAME VARCHAR2(14),  FACLOC VARCHAR2(13)  );  ALTER TABLE T\_FACTORY ADD PRIMARY KEY(FACTNO);    CREATE TABLE T\_PRODUCTS(  PDNO NUMBER,  PDNAME VARCHAR2(10),  PDSUBNAME VARCHAR2(10),  FACTNO VARCHAR2(5),  PDDATE DATE,  PDCOST NUMBER,  PDPRICE NUMBER,  PDAMOUNT NUMBER  );  ALTER TABLE T\_PRODUCTS ADD PRIMARY KEY(PDNO);  ALTER TABLE T\_PRODUCTS ADD FOREIGN KEY(FACTNO)  REFERENCES T\_FACTORY(FACTNO); | [cs](http://colorscripter.com/info#e) | |
| **2-2. DML**   |  |  |  | | --- | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68 | INSERT INTO T\_FACTORY (FACTNO, FACNAME, FACLOC)  VALUES ('1', 'EDIYA', 'SEOUL');    INSERT INTO T\_FACTORY (FACTNO, FACNAME, FACLOC)  VALUES ('2', 'STARBUCKS', 'NEWYORK');    INSERT INTO T\_FACTORY (FACTNO, FACNAME, FACLOC)  VALUES ('3', 'CAFEBENE', 'JEJU');    INSERT INTO T\_PRODUCTS VALUES(  001, 'COFFEE', 'AMERICANO', 1, SYSDATE, 500, 1500, 100  )    INSERT INTO T\_PRODUCTS VALUES(  002, 'COFFEE', 'LATTE', 2, SYSDATE, 1000, 2500, 100  )    INSERT INTO T\_PRODUCTS VALUES(  003, 'COFFEE', 'GREENLATTE', 2, SYSDATE, 1500, 3000, 50  )    INSERT INTO T\_PRODUCTS VALUES(  004, 'COFFEE', 'MOCHA', 2, SYSDATE, 1500, 3000, 55  )    INSERT INTO T\_PRODUCTS VALUES(  005, 'COFFEE', 'VANLATTE', 2, SYSDATE, 2000, 3500, 30  )    INSERT INTO T\_PRODUCTS VALUES(  006, 'COFFEE', 'VANLATTE', 1, SYSDATE, 1800, 3200, 30  )    INSERT INTO T\_PRODUCTS VALUES(  007, 'COFFEE', 'MOCHA', 1, SYSDATE, 1200, 2700, 70  )    INSERT INTO T\_PRODUCTS VALUES(  008, 'TEA', 'GREENTEA', 1, SYSDATE, 600, 1800, 170  )    INSERT INTO T\_PRODUCTS VALUES(  009, 'TEA', 'REDTEA', 1, SYSDATE, 600, 1900, 150  )    INSERT INTO T\_PRODUCTS VALUES(  010, 'TEA', 'BLACKTEA', 2, SYSDATE, 800, 2400, 130  )    INSERT INTO T\_PRODUCTS VALUES(  011, 'TEA', 'ICEDTEA', 2, SYSDATE, 400, 1600, 110  )    INSERT INTO T\_PRODUCTS VALUES(  012, 'DESERT', 'BREAD', 2, SYSDATE, 800, 3200, 45  )    INSERT INTO T\_PRODUCTS VALUES(  013, 'DESERT', 'BAGLE', 2, SYSDATE, 900, 3500, 30  )    INSERT INTO T\_PRODUCTS VALUES(  014, 'DESERT', 'CAKE', 1, SYSDATE, 1200, 4000, 25  )    INSERT INTO T\_PRODUCTS VALUES(  015, 'DESERT', 'MACARON', 2, SYSDATE, 800, 3500, 20  )  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) | |
| **3. UML**  **\\M506ins\공유\수행평가\1.SW기초기술\3조\jDBC04_UML.png** |
| **4. 작성코드**  **vo 패키지**  package vo;  public class Factory {    String fact\_no;  String fac\_name;  String fac\_loc;        public Factory() {  }  public Factory(String fact\_no, String fac\_name, String fac\_loc) {  this.fact\_no = fact\_no;  this.fac\_name = fac\_name;  this.fac\_loc = fac\_loc;  }  public String getFact\_no() {  return fact\_no;  }  public void setFact\_no(String fact\_no) {  this.fact\_no = fact\_no;  }  public String getFac\_name() {  return fac\_name;  }  public void setFac\_name(String fac\_name) {  this.fac\_name = fac\_name;  }  public String getFac\_loc() {  return fac\_loc;  }  public void setFac\_loc(String fac\_loc) {  this.fac\_loc = fac\_loc;  }  @Override  public String toString() {  return "Factory [fact\_no=" + fact\_no + ", fac\_name=" + fac\_name + ", fac\_loc=" + fac\_loc + "]";  }          }  package vo;  import java.sql.Date;  public class Products {    int pd\_no;  String pd\_name;  String pd\_sub\_name;  String fact\_no;  Date pd\_date;  int pd\_cost;  int pd\_price;  int pd\_amount;        public Products() {  }  public Products(int pd\_no, int pd\_amount) {  this.pd\_no = pd\_no;  this.pd\_amount = pd\_amount;  }  public Products(int pd\_no, String pd\_name, String pd\_sub\_name, String fact\_no, Date pd\_date, int pd\_cost,  int pd\_price, int pd\_amount) {  this.pd\_no = pd\_no;  this.pd\_name = pd\_name;  this.pd\_sub\_name = pd\_sub\_name;  this.fact\_no = fact\_no;  this.pd\_date = pd\_date;  this.pd\_cost = pd\_cost;  this.pd\_price = pd\_price;  this.pd\_amount = pd\_amount;  }  public Products(int pd\_no, String pd\_name, String pd\_sub\_name, String fact\_no, int pd\_cost,  int pd\_price, int pd\_amount) {  this.pd\_no = pd\_no;  this.pd\_name = pd\_name;  this.pd\_sub\_name = pd\_sub\_name;  this.fact\_no = fact\_no;  this.pd\_cost = pd\_cost;  this.pd\_price = pd\_price;  this.pd\_amount = pd\_amount;  }  public int getPd\_no() {  return pd\_no;  }  public void setPd\_no(int pd\_no) {  this.pd\_no = pd\_no;  }  public String getPd\_name() {  return pd\_name;  }  public void setPd\_name(String pd\_name) {  this.pd\_name = pd\_name;  }  public String getPd\_sub\_name() {  return pd\_sub\_name;  }  public void setPd\_sub\_name(String pd\_sub\_name) {  this.pd\_sub\_name = pd\_sub\_name;  }  public String getFact\_no() {  return fact\_no;  }  public void setFact\_no(String fact\_no) {  this.fact\_no = fact\_no;  }  public Date getPd\_date() {  return pd\_date;  }  public void setPd\_date(Date pd\_date) {  this.pd\_date = pd\_date;  }  public int getPd\_cost() {  return pd\_cost;  }  public void setPd\_cost(int pd\_cost) {  this.pd\_cost = pd\_cost;  }  public int getPd\_price() {  return pd\_price;  }  public void setPd\_price(int pd\_price) {  this.pd\_price = pd\_price;  }  public int getPd\_amount() {  return pd\_amount;  }  public void setPd\_amount(int pd\_amount) {  this.pd\_amount = pd\_amount;  }  @Override  public String toString() {  return "Products [pd\_no=" + pd\_no + ", pd\_name=" + pd\_name + ", pd\_sub\_name=" + pd\_sub\_name + ", fact\_no="  + fact\_no + ", pd\_date=" + pd\_date + ", pd\_cost=" + pd\_cost + ", pd\_price=" + pd\_price + ", pd\_amount="  + pd\_amount + "]";  }      }  **Frame 패키지**  package frame;  import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.SQLException;  import java.util.ArrayList;  public abstract class Biz<K,V> {    String id;  String pwd;  String url;    public Biz() {  // 1. JDBC Driver Loading..  try {  Class.forName("oracle.jdbc.driver.OracleDriver"); //자동으로 오라클에 접속하는 프로그램으로 바뀐다  } catch (ClassNotFoundException e) {  System.out.println("Driver Loading Error");  }  id = "db";  pwd = "db";  url = "jdbc:oracle:thin:@70.12.50.234:1521:xe";    }    public Connection getCon() { // connection  Connection con = null;  try {  con =  DriverManager.getConnection(url,id,pwd);  con.setAutoCommit(false);  } catch (SQLException e) {  e.printStackTrace();  }  return con;  }    public void close(Connection con) { //connection close  if(con != null) {  try {  con.close();  } catch (SQLException e) {  e.printStackTrace();  }  }  }    public abstract void insert(V v) throws Exception;  public abstract void delete(K k) throws Exception;  public abstract void update(V v) throws Exception;  public abstract V select(K k) throws Exception;  public abstract ArrayList<V> selectAll() throws Exception;    // public abstract V max(K k) throws Exception;      }  package frame;  import java.sql.Connection;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.sql.SQLException;  import java.util.ArrayList;  public abstract class Dao<K, V> {    public void close(PreparedStatement con) {  if(con != null) {  try {  con.close();  } catch (SQLException e) {  e.printStackTrace();  }  }  }    public void close(ResultSet con) {  if(con != null) {  try {  con.close();  } catch (SQLException e) {  e.printStackTrace();  }  }  }    public abstract void insert(V v,Connection con) throws Exception;  public abstract void delete(K k,Connection con) throws Exception;  public abstract void update(V v,Connection con) throws Exception;  public abstract V select(K k,Connection con) throws Exception;  public abstract ArrayList<V> selectAll(Connection con) throws Exception;    // public abstract V max(K k, Connection con) throws Exception;  }  package frame;  public class Sql {  public static String insertProducts = "INSERT INTO T\_PRODUCTS VALUES (?,?,?,?,SYSDATE,?,?,?)";  public static String deleteProducts = "DELETE FROM T\_PRODUCTS WHERE PDNO =?";  public static String updateProducts = "UPDATE T\_PRODUCTS SET PDAMOUNT=? WHERE PDNO=?";  public static String selectProducts = "SELECT \* FROM T\_PRODUCTS WHERE PDNO = ?";  public static String selectAllProducts = "SELECT \* FROM T\_PRODUCTS ORDER BY PDNO";  public static String insertFactory = "INSERT INTO T\_FACTORY VALUES (?,?,?)";  public static String deleteFactory = "DELETE FROM T\_FACTORY WHERE FACTNO =?";  public static String updateFactory = "UPDATE T\_FACTORY SET FACLOC=? WHERE FACTNO=?";  public static String selectFactory = "SELECT \* FROM T\_FACTORY WHERE FACTNO =?";  public static String selectAllFactory = "SELECT \* FROM T\_FACTORY ORDER BY FACTNO";  public static String maxPriceProductsInTheFactory = "SELECT \* FROM T\_FACTORY f, T\_PRODUCTS p WHERE f.FACTNO = p.FACTNO AND p.PDPRICE IN (SELECT MAX(p2.PDPRICE) FROM T\_PRODUCTS p2 GROUP BY FACTNO) AND f.FACTNO = ?";  }  **com 패키지**  package com;  import java.sql.Connection;  import java.util.ArrayList;  import frame.Biz;  import frame.Dao;  import vo.Factory;  public class FactoryBiz extends Biz<String, Factory> {  Dao<String, Factory> dao;  public FactoryBiz() {  dao = new FactoryDao();  }  @Override  public void insert(Factory v) throws Exception {  Connection con = null;  try {  con = getCon();  dao.insert(v, con);  con.commit();  }catch(Exception e) {  con.rollback();  throw e;  }finally {  close(con);  }  }  @Override  public void delete(String k) throws Exception {  Connection con = null;  try {  con = getCon();  dao.delete(k, con);  con.commit();  }catch(Exception e) {  con.rollback();  throw e;  }finally {  close(con);  }  }  @Override  public void update(Factory v) throws Exception {  Connection con = null;  try {  con = getCon();  dao.update(v, con);  con.commit();  }catch(Exception e) {  con.rollback();  throw e;  }finally {  close(con);  }  }  @Override  public Factory select(String k) throws Exception {  Connection con = getCon();  Factory f = null;    try {  f = dao.select(k, con);  } catch (Exception e) {    } finally {  close(con);  }    return f;  }  @Override  public ArrayList<Factory> selectAll() throws Exception {  Connection con = getCon();  ArrayList<Factory> fs = null;    try {  fs = dao.selectAll(con);  } catch (Exception e) {    } finally {  close(con);  }    return fs;  }    }  package com;  import java.sql.Connection;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.util.ArrayList;  import frame.Dao;  import frame.Sql;  import vo.Factory;  public class FactoryDao extends Dao<String,Factory> {  @Override  public void insert(Factory v, Connection con) throws Exception {  PreparedStatement pstmt = null;  try {  pstmt = con.prepareStatement(Sql.insertFactory);  pstmt.setString(1, v.getFact\_no());  pstmt.setString(2, v.getFac\_name());  pstmt.setString(3, v.getFac\_loc());  pstmt.executeUpdate();  }catch(Exception e) {  throw e;  }finally {  close(pstmt);  }  }  @Override  public void delete(String k, Connection con) throws Exception {  PreparedStatement pstmt = null;  try {  pstmt = con.prepareStatement(Sql.deleteFactory);  pstmt.setString(1, k);  pstmt.executeUpdate();  }catch(Exception e) {  throw e;  }finally {  close(pstmt);  }    }  @Override  public void update(Factory v, Connection con) throws Exception {  PreparedStatement pstmt = null;  try {  pstmt = con.prepareStatement(Sql.updateFactory);  pstmt.setString(1, v.getFac\_loc());  pstmt.setString(2, v.getFact\_no());  pstmt.executeUpdate();  }catch(Exception e) {  throw e;  }finally {  close(pstmt);  }    }  @Override  public Factory select(String k, Connection con) throws Exception {  PreparedStatement pstmt = null;  ResultSet rs = null;  Factory f = null;    try {  pstmt = con.prepareStatement(Sql.selectFactory);  pstmt.setString(1, k);  rs = pstmt.executeQuery();  rs.next();  f = new Factory(rs.getString("FACTNO"), rs.getString("FACNAME"), rs.getString("FACLOC"));  } catch (Exception e) {  throw e;  } finally {  close(rs);  close(pstmt);  }    return f;  }  @Override  public ArrayList<Factory> selectAll(Connection con) throws Exception {  PreparedStatement pstmt = null;  ResultSet rs = null;  ArrayList<Factory> fs = new ArrayList<Factory>();    try {  pstmt = con.prepareStatement(Sql.selectAllFactory);    rs = pstmt.executeQuery();    while(rs.next()) {  Factory f = new Factory(rs.getString("FACTNO"), rs.getString("FACNAME"), rs.getString("FACLOC"));    fs.add(f);  }  } catch (Exception e) {  throw e;  } finally {  close(rs);  close(pstmt);  }    return fs;  }  }  package com;  import java.sql.Connection;  import java.util.ArrayList;  import frame.Biz;  import frame.Dao;  import vo.Factory;  import vo.Products;  public class ProductsBiz extends Biz<String, Products> {  Dao<String, Products> dao;    public ProductsBiz() {  dao = new ProductsDao();  }  @Override  public void insert(Products v) throws Exception {  Connection con = null;  try {  con = getCon();  dao.insert(v, con);  con.commit();  }catch(Exception e) {  con.rollback();  throw e;  }finally {  close(con);  }  }  @Override  public void delete(String k) throws Exception {  Connection con = null;  try {  con = getCon();  dao.delete(k, con);  con.commit();  }catch(Exception e) {  con.rollback();  throw e;  }finally {  close(con);  }    }  @Override  public void update(Products v) throws Exception {  Connection con = null;  try {  con = getCon();  dao.update(v, con);  con.commit();  }catch(Exception e) {  con.rollback();  throw e;  }finally {  close(con);  }  }  @Override  public Products select(String k) throws Exception {  Connection con =getCon();  Products p=null;    try {  p=dao.select(k,con);    }catch(Exception e) {    }finally {  close(con);  }  return p;  }  @Override  public ArrayList<Products> selectAll() throws Exception {  Connection con = getCon();  ArrayList<Products> ps=null;    try {  ps=dao.selectAll(con);  }catch(Exception e) {    }finally {  close(con);  }  return ps;  }  public Products max(String k) throws Exception {  Connection con =getCon();  Products p=null;  ProductsDao pdao = (ProductsDao) dao;  try {  p=pdao.max(k,con);    }catch(Exception e) {  throw e;  }finally {  close(con);  }  return p;  }  }  package com;  import java.sql.Connection;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.util.ArrayList;  import frame.Dao;  import frame.Sql;  import vo.Factory;  import vo.Products;  public class ProductsDao extends Dao<String, Products> {  @Override  public void insert(Products v, Connection con) throws Exception {  PreparedStatement pstmt = null;  try {  pstmt = con.prepareStatement(Sql.insertProducts);  pstmt.setInt(1, v.getPd\_no());  pstmt.setString(2, v.getPd\_name());  pstmt.setString(3, v.getPd\_sub\_name());  pstmt.setString(4, v.getFact\_no());  pstmt.setInt(5, v.getPd\_cost());  pstmt.setInt(6, v.getPd\_price());  pstmt.setInt(7, v.getPd\_amount());  pstmt.executeUpdate();  }catch(Exception e) {  throw e;  }finally {  close(pstmt);  }  }  @Override  public void delete(String k, Connection con) throws Exception {  PreparedStatement pstmt = null;  try {  pstmt = con.prepareStatement(Sql.deleteProducts);  pstmt.setString(1, k);  pstmt.executeUpdate();  }catch(Exception e) {  throw e;  }finally {  close(pstmt);  }  }  @Override  public void update(Products v, Connection con) throws Exception {  PreparedStatement pstmt = null;  try {  pstmt = con.prepareStatement(Sql.updateProducts);  pstmt.setInt(1, v.getPd\_amount());  pstmt.setInt(2, v.getPd\_no());  pstmt.executeUpdate();  }catch(Exception e) {  throw e;  }finally {  close(pstmt);  }  }  @Override  public Products select(String k, Connection con) throws Exception {  PreparedStatement pstmt =null;  ResultSet rs=null;  Products p =null;    try {  pstmt=con.prepareStatement(Sql.selectProducts);  pstmt.setString(1,k);  rs=pstmt.executeQuery();  rs.next();  p=new Products(rs.getInt("PDNO"),rs.getString("PDNAME"),rs.getString("PDSUBNAME"),rs.getString("FACTNO"),  rs.getDate("PDDATE"),rs.getInt("PDCOST"),rs.getInt("PDPRICE"),  rs.getInt("PDAMOUNT"));  }catch(Exception e) {  throw e;  }finally {  close(rs);  close(pstmt);    }  return p;  }  @Override  public ArrayList<Products> selectAll(Connection con) throws Exception {  PreparedStatement pstmt=null;  ResultSet rs= null;  ArrayList<Products> ps=new ArrayList<Products>();  try {  pstmt=con.prepareStatement(Sql.selectAllProducts);    rs=pstmt.executeQuery();  while(rs.next()) {  Products p=new Products(rs.getInt("PDNO"),rs.getString("PDNAME"),rs.getString("PDSUBNAME"),rs.getString("FACTNO"),  rs.getDate("PDDATE"),rs.getInt("PDCOST"),rs.getInt("PDPRICE"),  rs.getInt("PDAMOUNT")); //INT형 EX) PD\_NO에 있는 값을 INT형으로 가져온다  ps.add(p);  } // "대문자" 형으로 맞춘다  }catch(Exception e) {  throw e;  }finally {  close(rs);  close(pstmt);  }  return ps;  }    public Products max(String k, Connection con) throws Exception {  PreparedStatement pstmt =null;  ResultSet rs=null;  Products p =null;    try {  pstmt=con.prepareStatement(Sql.maxPriceProductsInTheFactory);  pstmt.setString(1,k);  rs=pstmt.executeQuery();  rs.next();  p=new Products(rs.getInt("PDNO"),rs.getString("PDNAME"),rs.getString("PDSUBNAME"),rs.getString("FACTNO"),  rs.getDate("PDDATE"),rs.getInt("PDCOST"),rs.getInt("PDPRICE"),  rs.getInt("PDAMOUNT"));  }catch(Exception e) {  throw e;  }finally {  close(rs);  close(pstmt);    }  return p;  }  }  **test 패키지**  package test;  import com.FactoryBiz;  import frame.Biz;  import vo.Factory;  public class FactoryDelete {  public static void main(String[] args) {  String str = "3";  Biz<String, Factory> biz = new FactoryBiz();  try {  biz.delete(str);  System.out.println("OK");  } catch (Exception e) {  e.printStackTrace();  }  }  }  package test;  import com.FactoryBiz;  import frame.Biz;  import vo.Factory;  public class FactoryInsert {  public static void main(String[] args) {  Factory f = new Factory("4", "BLUEBOTTLE", "GANGNAM");  Biz<String, Factory> biz = new FactoryBiz();    try {  biz.insert(f);  System.out.println("OK");  } catch (Exception e) {  e.printStackTrace();  }  }  }  package test;  import com.FactoryBiz;  import frame.Biz;  import vo.Factory;  public class FactorySelect {  public static void main(String[] args) {  // TODO Auto-generated method stub  Biz<String,Factory> biz = new FactoryBiz();    try {  Factory f = biz.select("2");  System.out.println(f.toString());  } catch (Exception e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  }  }  package test;  import java.util.ArrayList;  import com.FactoryBiz;  import frame.Biz;  import vo.Factory;  public class FactorySelectAll {  public static void main(String[] args) {  Biz<String,Factory> biz = new FactoryBiz();    try {  ArrayList<Factory> fs = biz.selectAll();    for (Factory f : fs) {  System.out.println(f.toString());  }  } catch (Exception e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  }  }  package test;  import com.FactoryBiz;  import frame.Biz;  import vo.Factory;  public class FactoryUpdate {  public static void main(String[] args) {  Factory f = new Factory("4","","GANGNAM");  Biz<String, Factory> biz = new FactoryBiz();  try {  biz.update(f);  System.out.println("OK");  } catch (Exception e) {  e.printStackTrace();  }  }  }  package test;  import com.ProductsBiz;  import frame.Biz;  import vo.Products;  public class Max {  public static void main(String[] args) {  Biz<String,Products> biz= new ProductsBiz();  ProductsBiz pbiz = (ProductsBiz) biz;  try {  Products p=pbiz.max("2");  System.out.println(p.toString());  }catch(Exception e) {  e.printStackTrace();  }  }  }  package test;  import com.ProductsBiz;  import frame.Biz;  import vo.Products;  public class ProductDelete {  public static void main(String[] args) {  String str = "16";  Biz<String, Products> biz = new ProductsBiz();  try {  biz.delete(str);  System.out.println("OK");  } catch (Exception e) {  e.printStackTrace();  }  }  }  package test;  import com.ProductsBiz;  import frame.Biz;  import vo.Products;  public class ProductInsert {  public static void main(String[] args) {  Products p = new Products(16, "COFFEE", "LONGBLACK", "4", 1300, 3900, 32);  Biz<String, Products> biz = new ProductsBiz();    try {  biz.insert(p);  System.out.println("OK");  } catch (Exception e) {  e.printStackTrace();  }  }  }  package test;  import com.ProductsBiz;  import frame.Biz;  import vo.Products;  public class ProductSelect {  public static void main(String[] args) {  Biz<String,Products> biz= new ProductsBiz();    try {  Products p=biz.select("3");  System.out.println(p.toString());  }catch(Exception e) {  e.printStackTrace();  }  }  }  package test;  import java.util.ArrayList;  import com.ProductsBiz;  import frame.Biz;  import vo.Products;  public class ProductSelectAll {  public static void main(String[] args) {  Biz<String,Products> biz= new ProductsBiz();  try {  ArrayList<Products> ps=biz.selectAll();    for(Products p : ps) {  System.out.println(p.toString());  }  }catch(Exception e) {  e.printStackTrace();  }  }  }  package test;  import com.ProductsBiz;  import frame.Biz;  import vo.Products;  public class ProductUpdate {  public static void main(String[] args) {  Products p = new Products(16, 35);  Biz<String, Products> biz = new ProductsBiz();  try {  biz.update(p);  System.out.println("OK");  } catch (Exception e) {  e.printStackTrace();  }  }  } |