|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **□ 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** | | 2019.06.11 | 훈련생명 | |  | |
| **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      1. DDL, DML   **[PRODUCTS 테이블 생성 및 PRIMARY KEY 지정]**    **[PRODUCTS 테이블 생성]**    **[FACTORY 테이블 생성 및 PRIMARY KEY, FOREIGN KEY 지정]**    **[FACTORY 테이블 생성]**    **[PRODUCTS 테이블 DML]**      **[FACTORY 테이블 DML]**       1. UML      1. 작성코드   **[vo.Products.java]**   |  |  |  |  | | --- | --- | --- | --- | | package vo;    import java.sql.Date;    public class Products {        int pdno;      String pdname;      String pdsubname;      String factno;      Date pddate;      int pdcost;      int pdprice;      int pdamount;        public Products()      {      }        public Products(int pdno, String pdname, String pdsubname, String factno, Date d, int pdcost, int pdprice,int pdamount) {          this.pdno = pdno;          this.pdname = pdname;          this.pdsubname = pdsubname;          this.factno = factno;          this.pddate =  d;          this.pdcost = pdcost;          this.pdprice = pdprice;          this.pdamount = pdamount;      }        @Override      public String toString() {          return "Products [pdno=" + pdno + ", pdname=" + pdname + ", pdsubname=" + pdsubname + ", factno=" + factno + ", pddate=" + pddate + ", pdcost=" + pdcost + ", pdprice=" + pdprice + ", pdamount=" + pdamount + "]";      }      public int getPdno() {          return pdno;      }        public void setPdno(int pdno) {          this.pdno = pdno;      }        public String getPdname() {          return pdname;      }        public void setPdname(String pdname) {          this.pdname = pdname;      }        public String getPdsubname() {          return pdsubname;      }        public void setPdsubname(String pdsubname) {          this.pdsubname = pdsubname;      }        public String getFactno() {          return factno;      }        public void setFactno(String factno) {          this.factno = factno;      }        public Date getPddate() {          return pddate;      }        public void setPddate(Date pddate) {          this.pddate = pddate;      }        public int getPdcost() {          return pdcost;      }        public void setPdcost(int pdcost) {          this.pdcost = pdcost;      }        public int getPdprice() {          return pdprice;      }        public void setPdprice(int pdprice) {          this.pdprice = pdprice;      }        public int getPdamount() {          return pdamount;      }        public void setPdamount(int pdamount) {          this.pdamount = pdamount;      }  }  **[vo.Factory.java]**   |  |  | | --- | --- | | package vo;  public class Factory  {      String factno;      String facname;      String facloc;      public Factory() {      }      public Factory(String factno, String facname, String facloc) {          this.factno = factno;          this.facname = facname;          this.facloc = facloc;      }      @Override      public String toString() {          return "Factory [factno=" + factno + ", facname=" + facname + ", facloc=" +  facloc + "]";      }      public String getFactno() {          return factno;      }      public void setFactno(String factno) {          this.factno = factno;      }      public String getFacname() {          return facname;      }      public void setFacname(String facname) {          this.facname = facname;      }      public String getFacloc() {          return facloc;      }      public void setFacloc(String facloc) {          this.facloc = facloc;      }  } | [cs](http://colorscripter.com/info#e) | | [cs](http://colorscripter.com/info#e) |   **[vo.Join.java]**   |  |  | | --- | --- | | package vo;  public class Join  {      String pname;      String psubname;      String fname;      String floc;        public Join()      {      }        public Join(String pname, String psubname, String fname, String floc)      {          this.pname = pname;          this.psubname = psubname;          this.fname = fname;          this.floc = floc;      }        public String getPname() {          return pname;      }      public void setPname(String pname) {          this.pname = pname;      }      public String getPsubname() {          return psubname;      }      public void setPsubname(String psubname) {          this.psubname = psubname;      }      public String getFname() {          return fname;      }      public void setFname(String fname) {          this.fname = fname;      }      public String getFloc() {          return floc;      }      public void setFloc(String floc) {          this.floc = floc;      }      @Override      public String toString() {          return "Join [pname=" + pname + ", psubname=" + psubname + ", fname=" +  fname + ", floc=" + floc + "]";      }  }    [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |   **[frame.Biz.java]**   |  |  | | --- | --- | | package frame;    import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.sql.SQLException;  import java.util.ArrayList;  import vo.Join;    public abstract class Biz<K,V> {        String id;      String pwd;      String url;        public Biz()      {          try          {              Class.forName("oracle.jdbc.driver.OracleDriver");          }          catch (ClassNotFoundException e)          {              System.out.println("Driver Loading Error");          }          id = "db";          pwd = "db";          url = "jdbc:oracle:thin:@70.12.50.237:1521:xe";      }      public Connection getCon() {          Connection con =null;          try {              con =              DriverManager.getConnection(url, id, pwd);              con.setAutoCommit(false);//transaction 처리를 수동으로 하겠다.          } catch (SQLException e) {              e.printStackTrace();          }          return con;      }      public void close(Connection con) {          if(con!= null) {              try {                  con.close();              } catch (SQLException e) {                  e.printStackTrace();              }          }        }      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 register(V v)              throws Exception;      public abstract void remove(K k) //id 선택              throws Exception;      public abstract void modify(V v)              throws Exception;      public abstract V get(K k)              throws Exception;      public abstract ArrayList<V> get()              throws Exception;      public abstract ArrayList<Join> getall()              throws Exception;  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |   **[frame.Dao.java]**   |  |  | | --- | --- | | package frame;    import java.sql.Connection;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.sql.SQLException;  import java.util.ArrayList;    import vo.Join;    public abstract class Dao<K,V>{//data access object        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) //Biz가 커넥션 준다.              throws Exception;      public abstract void delete(K k, Connection con) //id 선택              throws Exception;      public abstract void update(V v, Connection con)              throws Exception;      public abstract V select(K k, Connection con) // id 선택              throws Exception;      public abstract ArrayList<V> select(Connection con)              throws Exception;      public abstract ArrayList<Join> selectAll(Connection con)              throws Exception;  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |   **[frame.SqlProducts.java]**   |  |  | | --- | --- | | package frame;    public class SqlProducts {      public static String insertProduct = "INSERT INTO PRODUCTS VALUES(?,?,?,?,?,?,?,?)";      public static String DeleteProduct = "DELETE FROM PRODUCTS WHERE PDNO = ?";      public static String UpdateProduct = "Update PRODUCTS SET PDNAME=?, PDSUBNAME=?, FACTNO=?, PDDATE=?, PDCOST=?, PDPRICE=?, PDAMOUNT = ?  WHERE PDNO=?";      public static String SelectProduct = "SELECT \* FROM PRODUCTS WHERE PDNO = ?";      public static String SelectAllProduct = "SELECT \* FROM PRODUCTS";      public static String SelectAll = "SELECT p.PDNAME,p.PDSUBNAME,f.FACNAME,f.FACLOC FROM PRODUCTS p, FACTORY f WHERE p.FACTNO = f.FACTNO";  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |   **[frame.SqlFactory.java]**   |  |  | | --- | --- | | package frame;    public class SqlFactory {      public static String insertFactory = "INSERT INTO FACTORY VALUES(?,?,?)";      public static String DeleteFactory = "DELETE FROM FACTORY WHERE FACTNO = ?";      public static String UpdateFactory = "Update PRODUCTS SET FACNAME =?, FACLOC=?, WHERE FACTNO=?";      public static String SelectFactory = "SELECT \* FROM PRODUCTS WHERE PDNO = ?";      public static String SelectAllFactory = "SELECT \* FROM PRODUCTS";  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |   **[com.ProductsBiz.java]**   |  |  | | --- | --- | | package com;    import java.sql.Connection;  import java.util.ArrayList;  import frame.Biz;  import frame.Dao;  import vo.Products;  import vo.Join;    public class ProductsBiz extends Biz<Integer, Products>  {      Dao<Integer,Products> dao;      public ProductsBiz()      {          dao = new ProductsDao();      }      @Override      public void register(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 remove(Integer 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 modify(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 get(Integer k) throws Exception {          Connection con = null;          Products u = null;          try {              con = getCon();              u = dao.select(k, con);          }catch(Exception e) {              throw e;          }finally {          close(con);          }          return u;      }        @Override      public ArrayList<Products> get() throws Exception {          Connection con = null;          ArrayList<Products> a = null;          try {              con = getCon();              a = dao.select(con);          }catch(Exception e) {              throw e;          }finally {          close(con);          }          return a;      }        public ArrayList<Join> getall() throws Exception {          Connection con = null;          ArrayList<Join> a = null;          try {              con = getCon();              a = dao.selectAll(con);          }catch(Exception e) {              throw e;          }finally {          close(con);          }          return a;      }  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |   **[com.ProductsDao.java]**   |  |  | | --- | --- | | package com;    import java.sql.Connection;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.sql.SQLException;  import java.util.ArrayList;  import java.sql.Date;  import frame.Dao;  import frame.SqlProducts;  import vo.Join;  import vo.Products;    public class ProductsDao extends Dao<Integer, Products> {      @Override      public void insert(Products v, Connection con) throws Exception {          PreparedStatement pstmt = null;          try{          pstmt = con.prepareStatement(SqlProducts.insertProduct);          pstmt.setInt(1,v.getPdno());          pstmt.setString(2,v.getPdname());          pstmt.setString(3,v.getPdsubname());          pstmt.setString(4,v.getFactno());          pstmt.setDate(5,v.getPddate());          pstmt.setInt(6,v.getPdcost());          pstmt.setInt(7,v.getPdprice());          pstmt.setInt(8,v.getPdamount());          pstmt.executeUpdate();          }catch(Exception e) {              throw e;          }finally {          close(pstmt);//close 안 될 수 있으니 주의!          }      }        @Override      public void delete(Integer k, Connection con) throws Exception {          PreparedStatement pstmt = null;          try {              pstmt = con.prepareStatement(SqlProducts.DeleteProduct);              pstmt.setInt(1, k);              int result = pstmt.executeUpdate();              System.out.println(result);          } catch (SQLException e) {              throw e;          }finally {              close(pstmt);              }      }        @Override      public void update(Products v, Connection con) throws Exception {          PreparedStatement pstmt = null;          try {              pstmt = con.prepareStatement(SqlProducts.UpdateProduct);              pstmt.setString(1, v.getPdname());              pstmt.setString(2, v.getPdsubname());              pstmt.setString(3, v.getFactno());              pstmt.setDate(4, v.getPddate());              pstmt.setInt(5, v.getPdcost());              pstmt.setInt(6, v.getPdprice());              pstmt.setInt(7, v.getPdamount());              pstmt.setInt(8, v.getPdno());              pstmt.executeUpdate();          } catch (SQLException e) {              throw e;          } finally {              close(pstmt);              }      }        @Override      public Products select(Integer k, Connection con) throws Exception {          PreparedStatement pstmt = null;          ResultSet rset = null;          Products p;          try {              pstmt = con.prepareStatement(SqlProducts.SelectProduct);              pstmt.setInt(1, k);              rset = pstmt.executeQuery();              rset.next();              int pno = rset.getInt("PDNO");              String pname = rset.getString("PDNAME");              String psubname = rset.getString("PDSUBNAME");              String fno = rset.getString("FACTNO");              Date pdate = rset.getDate("PDDATE");              int pcost = rset.getInt("PDCOST");              int pprice = rset.getInt("PDPRICE");              int pamount = rset.getInt("PDAMOUNT");              p = new Products(pno,pname,psubname,fno,pdate,pcost,pprice,pamount);          } catch (SQLException e) {              throw e;          } finally {              close(pstmt);              }          return p;        }        @Override      public ArrayList<Products> select(Connection con) throws Exception {          ArrayList<Products> a = new ArrayList<>();          PreparedStatement pstmt = null;          ResultSet rset = null;          try {              pstmt = con.prepareStatement(SqlProducts.SelectAllProduct);              rset = pstmt.executeQuery();              while(rset.next()) {                  int pno = rset.getInt("PDNO");                  String pname = rset.getString("PDNAME");                  String psubname = rset.getString("PDSUBNAME");                  String fno = rset.getString("FACTNO");                  Date pdate = rset.getDate("PDDATE");                  int pcost = rset.getInt("PDCOST");                  int pprice = rset.getInt("PDPRICE");                  int pamount = rset.getInt("PDAMOUNT");                  a.add(new Products(pno,pname,psubname,fno,pdate,pcost,pprice,pamount));              }          }catch (SQLException e) {              throw e;          }finally {              close(pstmt);              }          return a;            }        @Override      public ArrayList<Join> selectAll(Connection con) throws Exception {          // TODO Auto-generated method stub          ArrayList<Join> b = new ArrayList<>();          PreparedStatement pstmt = null;          ResultSet rset = null;          try {              pstmt = con.prepareStatement(SqlProducts.SelectAll);              rset = pstmt.executeQuery();              while(rset.next()) {                  String pname = rset.getString("PDNAME");                  String psubname = rset.getString("PDSUBNAME");                  String fname = rset.getString("FACNAME");                  String floc = rset.getString("FACLOC");                  b.add(new Join(pname, psubname,fname,floc));              }          }catch (SQLException e) {              throw e;          }finally {              close(pstmt);              }          return b;      }  }    [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |   **[test.ProductsJoin.java]**   |  |  | | --- | --- | | package test;    import java.util.ArrayList;  import com.ProductsBiz;  import frame.Biz;  import vo.Join;  import vo.Products;    public class ProducstJoin {      public static void main(String[] args) {          Biz<Integer, Products> biz = new ProductsBiz();              ArrayList<Join> a;              try              {                  a = biz.getall();                  for(Join u: a)                  {                      System.out.println(u);                  }              }              catch(Exception e)              {                  e.printStackTrace();              }      }  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |   **[test.ProductsSelect.java]**   |  |  | | --- | --- | | package test;    import com.ProductsBiz;  import frame.Biz;  import vo.Products;    public class ProductsSelect {          public static void main(String[] args) {              Biz<Integer, Products> biz = new ProductsBiz();              int pdno = 99;              try {                  System.out.println(biz.get(pdno));              }catch(Exception e){                  e.printStackTrace();              }          }  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |   **[test.ProductsInsert.java]**   |  |  | | --- | --- | | package test;    import com.ProductsBiz;  import frame.Biz;  import vo.Products;    public class ProductsInsert {        public static void main(String[] args) {          long milis = System.currentTimeMillis();          java.sql.Date d = new java.sql.Date(milis);          Products p = new Products(99,"핸드폰","갤럭99","3333",d,900,190000,19);          Biz<Integer, Products> biz = new ProductsBiz();          try {              biz.register(p);              System.out.println("OK");          } catch (Exception e) {              e.printStackTrace();          }      }  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) | |  |  |   **[test.ProductsDelete.java]**   |  |  | | --- | --- | | package test;    import com.ProductsBiz;  import frame.Biz;  import vo.Products;    public class ProductsDelete {      public static void main(String[] args) {          Biz<Integer,Products> biz = new ProductsBiz();          int pdno = 3;          try          {              biz.remove(pdno);              System.out.println("Product Deleted");          }          catch(Exception e){              e.printStackTrace();          }      }  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |   **[test.ProductsUpdate.java]**   |  |  | | --- | --- | | package test;    import com.ProductsBiz;  import frame.Biz;  import vo.Products;    public class ProductsUpdate {          public static void main(String[] args) {              Biz<Integer, Products> biz = new ProductsBiz();              long milis = System.currentTimeMillis();              java.sql.Date d = new java.sql.Date(milis);              Products p = new Products(1,"핸드폰","갤럭9","1111",d,900,90000,9);              try              {                  biz.modify(p);                  System.out.println("Products Updated");              }              catch(Exception e)              {                  e.printStackTrace();              }          }  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |  1. CRUD 및 JOIN 결과   **[SELECT]**    **[SELECTALL]**    **[INSERT]**      **[UPDATE]**    **[DELETE]** | | | | | | |