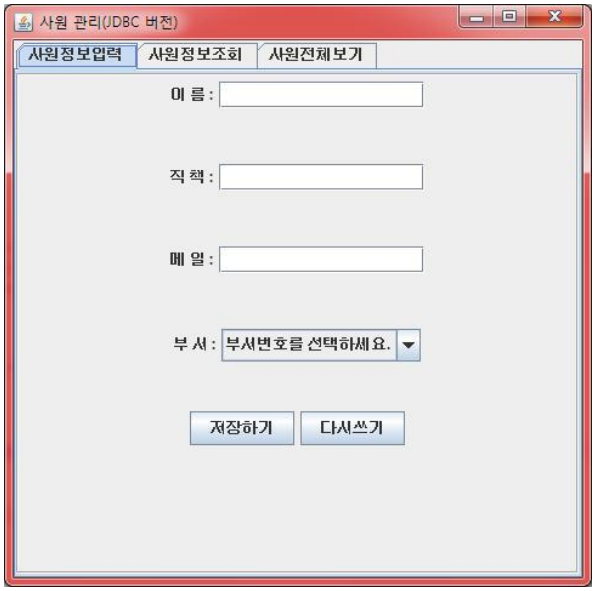
27. 미니 프로젝트 사원 정보 관리 프로그램

사원 관리는 크게 새로운 사원의 등록과 사원 정보를 조회하는 작업으로 나눈다.

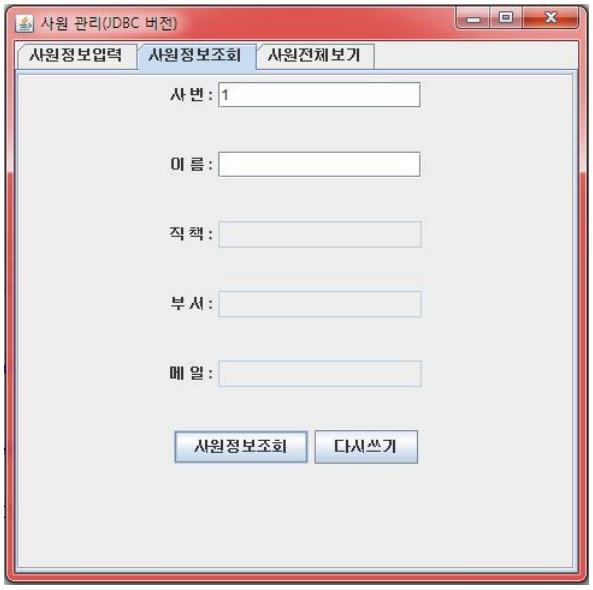
사원의 등록화면

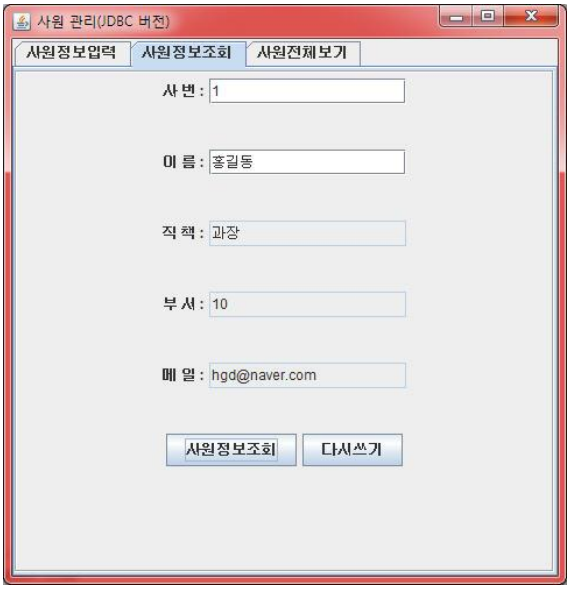
사원의 등록은 이름과 직책, 메일, 부서를 입력한 후 저장하기 버튼을 클릭하면 자동으로 사번의 일련번호가 생성되어 사원의 정보가 추가된다.



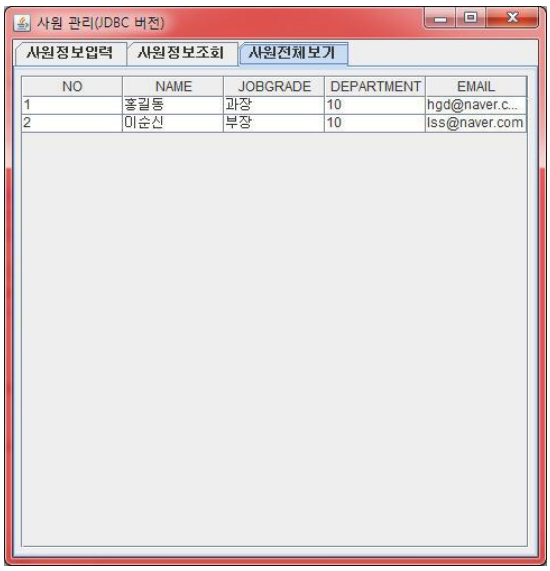
사원 정보 조회 화면

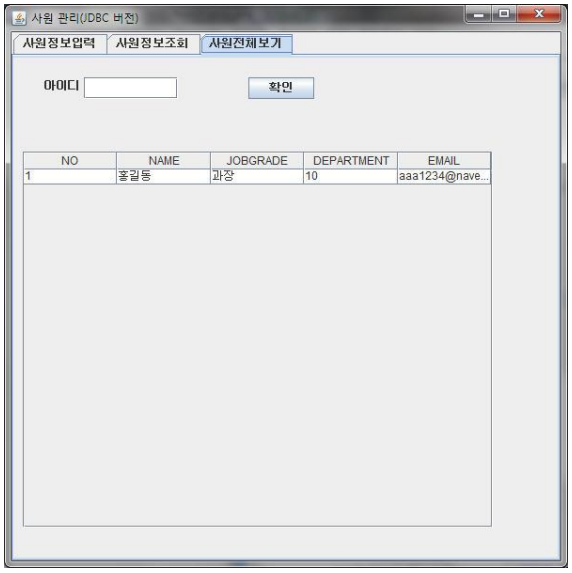
사번이나 이름으로 조회할 수 있으며, 사번과 이름을 동시에 입력할 경우에는 두 가지 모두 일치해야 한다.





사원 전체 보기 화면






```
create table employee (
no number primary key,
name varchar2(20) not null,
jobgrade varchar2(10),
department number,
email varchar2(20)
);

create sequence employee_seq
start with 1
increment by 1
nocycle;
```

1. VO 클래스

VO(Value Object)클래스는 데이터를 담는 컨테이너 역할을 하는 클래스로 데이터 전달을 목적으로 만들어진 클래스이다. VO클래스는 어플리케이션 구조에서 각 단계의 입출력 정보를 전달하는 역할을 수행하며, 속성(attribute), setter와 getter로 구성된다.

속성(attribute) : VO 클래스에 입력되는 정보를 저장한다.
setter 메소드 : 정보를 VO클래스에 저장할 때 사용한다.
getter 메소드 : VO클래스의 정보를 조회할 때 사용한다.

EmployeeVO.java

```
package exam01;


public class EmployeeVO {
```

```
	// 필드
	private int no;
	private String name;
	private String jobGrade;
	private int department;
	private String email;
	private String status;
	// 생성자
	public EmployeeVO(int no, String name, String jobGrade, int department, String email) {
		this.no = no;
		this.name = name;
		this.jobGrade = jobGrade;
		this.department = department;
		this.email = email;
	}
	// 이전 생성자 매개변수에 status v필드의 초기화를 추가한다.
	public EmployeeVO(int no, String name, String jobGrade, int department, String email, String
status) {
		this.no = no;
		this.name = name;
		this.jobGrade = jobGrade;
		this.department = department;
		this.email = email;
		this.status = status;
	}
	// 디폴트 생성자
	public EmployeeVO() {
	}
	public int getNo() {
		return no;
	}
	public void setNo(int no) {
		this.no = no;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getJobGrade() {
```

```
		return jobGrade;
	}
	public void setJobGrade(String jobGrade) {
		this.jobGrade = jobGrade;
	}
	public int getDepartment() {
		return department;
	}
	public void setDepartment(int department) {
		this.department = department;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
	public String getStatus() {
		return status;
	}
	public void setStatus(String status) {
		this.status = status;
	}
	public String toString() {
		return getNo()+","+getName();
	}
}
```

2. DAO 클래스

데이터베이스와 연동을 위한 DBUtil 클래스를 작성한다.

메소드	설 명
getConnection( )	Class 클래스의 forName() 메소드를 호출하여 Oracle의 JDBC 드라이버를 등록한다. JDBC 드라이버를 등록한 후에 DriverManager 클래스의 getConnection( ) 메소드를 호출하여 데이터베이스에 대한 커넥션을 얻는다.

DBUtil.java

```
package exam01;

import java.sql.Connection;
import java.sql.DriverManager;
```

```
public class DBUtil {
	static final String driver = "oracle.jdbc.driver.OracleDriver";
	static final String url ="jdbc:oracle:thin:@127.0.0.1:1521:orcl";
	public static Connection getConnection( ) throws Exception{
		Class.forName(driver);
		Connection con = DriverManager.getConnection(url, "scott", "tiger");
		return con;
	}
}
```

DAO(Data Access Object)클래스가 데이터 처리의 궁극적인 단계이다.
비즈니스 로직에서 처리된 자료를 받아 데이터베이스에 입력과 조회를 한다.

메소드	설 명
getEmployeeregiste()	신규 사용자 등록
getEmployeeCheck()	사원의 no, name을 입력 받아 사원 정보 조회
getEmployeeNo()	사원의 no를 입력 받아 정보를 조회
getEmployeeName()	사원의 name을 입력 받아 정보를 조회

위 메소드는 공통으로 커넥션 객체의 prepareStatement() 메소드를 호출하고, SQL 쿼리를 실행시켜 데이터베이스를 조작할 PreparedStatement 객체를 얻는다.
값이 들어갈 자리에 텍스트 필드의 getText() 메소드를 사용하여 얻은 문자열을 대입하여 쿼리문을 완성한다.

EmployeeDAO.java

```
package exam01;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.util.ArrayList;

public class EmployeeDAO {
    //① 신규 사용자 등록
    public EmployeeVO getEmployeeregiste(EmployeeVO evo) throws Exception {
        //② 데이터 처리를 위한 SQL 문
        String                  dml      = "insert into employee "
                                         + "   (no, name, jobGrade, department, email)"
                                         + "values "
```

```
                                    + "   (employee_seq.nextval, ?, ?, ?, ?)";
    Connection              con      =   null;
    PreparedStatement   pstmt    = null;
    EmployeeVO           retval    = null;

    try{
        //③ DBUtil이라는 클래스의 getConnection( )메서드로 데이터베이스와 연결
        con    = DBUtil.getConnection( );

        //④ 입력받은 사용자 정보를 처리하기 위하여 SQL문장을 생성
        pstmt = con.prepareStatement(dml);
        pstmt.setString(1, evo.getName( ));
        pstmt.setString(2, evo.getJobGrade( ));
        pstmt.setInt(3, evo.getDepartment( ));
        pstmt.setString(4, evo.getEmail( ));

        //⑤ SQL문을 수행후 처리 결과를 얻어옴
        int i   = pstmt.executeUpdate( );
        retval = new EmployeeVO( );

        retval.setStatus(i+"");    // SQL 문 수행 결과를 EmpioyeeVo 에 세팅

    }catch(SQLException e) {
        System.out.println("e=["+e+"]");
    }catch (Exception e){
        System.out.println("e=["+e+"]");
    } finally{
        try{
            //⑥ 데이터베이스와의 연결에 사용되었던 오브젝트를 해제
            if (pstmt != null) pstmt.close( );
            if (con != null) con.close( );
        }catch(SQLException e){
        }
    }
    return retval;
}
//⑦ 사원의 no, name을 입력받아 사원 정보 조회
public EmployeeVO getEmployeeCheck(int no, String name) throws Exception {
    String              dml       = "select * from employee where no = ? and name = ?";
    Connection          con     = null;
    PreparedStatement pstmt = null;
```

```
    ResultSet               rs        = null;
    EmployeeVO              retval = null;
    try{
        con     = DBUtil.getConnection( );
        pstmt = con.prepareStatement(dml);
        pstmt.setInt(1, no);
        pstmt.setString(2, name);
        rs       = pstmt.executeQuery( );
        if(rs.next( )) {
             retval = new EmployeeVO( rs.getInt(1),
                                      rs.getString(2),
                                      rs.getString(3),
                                      rs.getInt(4),
                                      rs.getString(5));
        }
    }catch(SQLException se) {
        System.out.println(se);
    }catch (Exception e){
        System.out.println(e);
    }finally{
        try{
            if (rs != null) rs.close( );
            if (pstmt != null) pstmt.close( );
            if (con != null) con.close( );
        }catch(SQLException se){
        }
    }
    return retval;
}

//⑧ 사원의 no를 입력받아 정보를 조회
public EmployeeVO getEmployeeNo(int no) throws Exception {
    String                  dml      = "select * from employee where no = ?";
    Connection              con      = null;
    PreparedStatement pstmt    = null;
    ResultSet               rs       = null;
    EmployeeVO          retval    = null;

    try{
        con     = DBUtil.getConnection( );
        pstmt = con.prepareStatement(dml);
```

```
            pstmt.setInt(1, no);
            rs      = pstmt.executeQuery( );
            if(rs.next( )) {
                retval = new EmployeeVO( rs.getInt(1),
                                         rs.getString(2),
                                         rs.getString(3),
                                         rs.getInt(4),
                                         rs.getString(5));
            }
        }catch(SQLException se) {
            System.out.println(se);
        }catch (Exception e){
            System.out.println(e);
        }finally{
            try{
                if (rs != null) rs.close( );
                if (pstmt != null) pstmt.close( );
                if (con != null) con.close( );
            }catch(SQLException se){
            }
        }
        return retval;
    }

    //⑨ 사원의 name을 입력받아 정보를 조회
    public EmployeeVO getEmployeeName(String name) throws Exception {
        String              dml     = "select * from employee where name = ?";
        Connection          con     = null;
        PreparedStatement pstmt = null;
        ResultSet           rs      = null;
        EmployeeVO          retval = null;
        try{
            con = DBUtil.getConnection( );
            pstmt = con.prepareStatement(dml);
            pstmt.setString(1, name);
            rs = pstmt.executeQuery( );
            if(rs.next( )) {
                retval = new EmployeeVO(   rs.getInt(1),
                                           rs.getString(2),
                                           rs.getString(3),
                                           rs.getInt(4),
```

```
                                        rs.getString(5));
            }
        }catch(SQLException se) {
            System.out.println(se);
        }catch (Exception e){
             System.out.println(e);
        }finally{
            try{
                if (rs != null) rs.close( );
                if (pstmt != null) pstmt.close( );
                if (con != null) con.close( );
            }catch(SQLException se){
            }
        }
        return retval;
    }

    public ArrayList<EmployeeVO> getEmployeetotal(){
        ArrayList<EmployeeVO> list=new ArrayList<EmployeeVO>();
        String                tml       = "select * from employee";

        Connection            con     = null;
        PreparedStatement pstmt = null;
        ResultSet             rs        = null;
        EmployeeVO            emVo = null;

        try{
            con     = DBUtil.getConnection();
            pstmt = con.prepareStatement(tml);
            rs      = pstmt.executeQuery();
            while(rs.next()) {
                emVo = new EmployeeVO(     rs.getInt(1),
                                           rs.getString(2),
                                           rs.getString(3),
                                           rs.getInt(4),
                                           rs.getString(5));
                list.add(emVo);
            }
        }catch(SQLException se) {
            System.out.println(se);
        }catch (Exception e){
```

```
            System.out.println(e);
        }finally{
            try{
                if (rs != null) rs.close();
                if (pstmt != null) pstmt.close();
                if (con != null) con.close();
            }catch(SQLException se){
            }
        }
        return list;
    }

    public ArrayList<String> getColumnName(){
        ArrayList<String> columnName=new ArrayList<String>();

        String    sql    = "select * from employee";
        Connection          con     = null;
        PreparedStatement pstmt = null;
        ResultSet           rs        = null;
        //ResultSetMetaData 객체 변수 선언
        ResultSetMetaData rsmd = null;
        try{
            con     = DBUtil.getConnection();
            pstmt = con.prepareStatement(sql);
            rs       = pstmt.executeQuery();
            rsmd    = rs.getMetaData();
            int cols = rsmd.getColumnCount();
            for( int i = 1 ; i <= cols ; i++){
                columnName.add(rsmd.getColumnName(i));
            }
        }catch(SQLException se) {
            System.out.println(se);
        }catch (Exception e){
            System.out.println(e);
        }finally{
            try{
                if (rs != null) rs.close();
                if (pstmt != null) pstmt.close();
                if (con != null) con.close();
            }catch(SQLException se){
            }
```

```
        }
        return columnName;
    }
}
```

3. 화면 레이아웃

사용자로부터 입력을 받거나 처리된 결과를 보여주기 위한 아플리케이션의 프리젠테이션 기능을 구현한다.

사용자가 쉽게 사용할 수 있도록 UI(User Interface) 컴포넌트를 배열하고 Navigation 정책을 세운다.

메소드	기 능
AddPane()	사원 정보 입력 탭 객체는 정보를 입력 받기 위한 텍스트 필드 배열, 저장하기 버튼, 다시 쓰기 버튼으로 구성된다.
actionPerformed()	저장하기 버튼을 누르면 텍스트 필드의 내용을 데이터베이스에 저장한다. 다시 쓰기 버튼을 누르면 텍스트 필드의 내용을 초기화 한다.

AddPane.java

```
package exam01;

import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;

import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.border.EtchedBorder;

// 패널로 사원 정보 입력을 위한 클래스 설계
public class AddPane extends JPanel implements ActionListener, ItemListener{
        //Swing 관련 멤버
        private JPanel jp[]=new JPanel[6];
        private JLabel jl[]=new JLabel[5];
        private JTextField tf[]=new JTextField[5];
        private JButton okb;
        private JButton rsb;
```

```
    private int department=10;
    // 입력 받을 항목의 제목
    String []caption={"이 름 :", "직 책 :", "메 일 :", "부 서 :"};

    public AddPane(){
      setLayout(new GridLayout(6,1));
      EtchedBorder eb = new EtchedBorder();
      setBorder(eb);

      int size=caption.length;  //입력 받을 항목의 개수를 구한다.
      // 텍스트 필드와 레이블을 패널에 묶어서 AddPane에 붙인다.
      int i;
      for(i=0; i<size-1; i++){
            jp[i] = new JPanel();
            jl[i] = new JLabel(caption[i]);
            tf[i]= new JTextField(15);
            jp[i].add(jl[i]);
            jp[i].add(tf[i]);
            add(jp[i]);
      }
      jp[i] = new JPanel();
      jl[i] = new JLabel(caption[i]);
      jp[i].add(jl[i]);  //화면의 공간을 확보한다.
      add(jp[i]);

      JComboBox combo = new JComboBox( );
      combo.addItem("부서번호를 선택하세요.");
      for(int c=1; c<=5; c++)
              combo.addItem(c*10);    // 부서 번호 추가
      jp[i].add(combo);
      combo.addItemListener(this);

      jp[size] = new JPanel();
      okb = new JButton("저장하기");
      okb.addActionListener(this);
      rsb = new JButton("다시쓰기");
      rsb.addActionListener(this);
      jp[size].add(okb);
      jp[size].add(rsb);
      add(jp[size]);
    }
```

```
        // 버튼에 대한 이벤트 처리
        public void actionPerformed(ActionEvent ae) {
                String ae_type = ae.getActionCommand();
                EmployeeVO evo=null;
                EmployeeDAO edvo=null;
                if(ae_type.equals(okb.getText())) {    //저장하기 버튼이 클릭되었을 경우
                        try {
                  evo=new EmployeeVO(0, tf[0].getText(), tf[1].getText(), department, tf[2].getText());
                            //입력받은 사원 정보를 데이터베이스에 추가하기 위한 DAO 객체생성
                            edvo=new EmployeeDAO();
                            edvo.getEmployeeregiste(evo);
                        }catch (Exception e){
                            System.out.println("e=["+e+"]");
                        }
                        if(edvo != null )
                 JOptionPane.showMessageDialog(this, tf[0].getText() + "님이 성공적으로 추가됨");
                 } else if(ae_type.equals(rsb.getText())) {
                           int size=caption.length;
                             //텍스트 필드를 초기화 한다.
                             for(int i = 0; i<size-1 ; i++) {
                                 tf[i].setText("");
                             }
                 }
          }
        // 콤보 박스에 대한 이벤트 처리
        public void itemStateChanged(ItemEvent ie) {
                if(ie.getStateChange()==ItemEvent.SELECTED)
                        department=Integer.parseInt(ie.getItem().toString());
        }
}
```

사원 정보 조회를 위한 UI와 검색 처리를 하는 FindPane 클래스

메소드	설 명
FindPane()	사원 정보 조회 탭 객체는 검색을 위한 사원 정보를 입력 받기 위한 텍스트 필드와 사원 조회 버튼, 다시 쓰기 버튼으로 구성된다.
actionPerformed()	사원 조회 버튼을 누르면 텍스트 필드의 내용과 일치하는 데이터를 데어베이스로부터 읽어온다. 데이터베이스를 조회한 후 그 결과를 출력한다. 다시 쓰기 버튼을 누르면 텍스트 필드의 내용을 초기화 한다.

FindPane.java

```
package exam01;

import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.border.EtchedBorder;

public class FindPane extends JPanel implements ActionListener {
	//Swing 관련 멤버
	private JPanel jp[]=new JPanel[6];
	private JLabel jl[]=new JLabel[5];
	private JTextField tf[]=new JTextField[5];
	private JButton okb;
	private JButton rsb;
	String []caption={"사 번 :", "이 름 :", "직 책 :", "부 서 :", "메 일 :"};

	public FindPane(){
	  setLayout(new GridLayout(7,1));
	  EtchedBorder eb = new EtchedBorder();
	  setBorder(eb);
	  int size=caption.length;
	  for(int i=0; i<size; i++){
		jl[i] = new JLabel(caption[i]);
		tf[i]= new JTextField(15);
		jp[i] = new JPanel();
		jp[i].add(jl[i]);
		jp[i].add(tf[i]);
		add(jp[i]);
		tf[i].setEditable(false);  // 텍스트 필드를 입력 불가능한 상태로 만든다.
		if(i==0 || i==1)
			tf[i].setEditable(true);  //사번과 이름으로 검색해야 입력 가능한 상태로 만든다.
	  }
	  jp[size] = new JPanel();
	  okb = new JButton("사원정보조회");
```

```
        okb.addActionListener(this);
        rsb = new JButton("다시쓰기");
        rsb.addActionListener(this);
        jp[size].add(okb);
        jp[size].add(rsb);
        add(jp[size]);
    }
    public void actionPerformed(ActionEvent ae) {
        String ae_type = ae.getActionCommand(); //이벤트가 발생한 버튼의 캡션 값을 얻어온다.
                EmployeeVO evo=null;        // 검색한 사원 정보를 저장할 VO 객체 생성
                EmployeeDAO edvo=null;   // 데이터베이스 처리를 위한 DAO 객체 생성
                 if(ae_type.equals(okb.getText())) {   //성적조회 버튼이 클릭되었을 경우
                            try {
                                edvo=new EmployeeDAO();
                                String sno=tf[0].getText().trim();
                                String sname=tf[1].getText().trim();

                                if(!sno.equals("") && !sname.equals("")){     // 사번과 이름 입력
                                        int no=Integer.parseInt(sno);
                                        evo=edvo.getEmployeeCheck(no, sname);
                                }else if(!sno.equals("") && sname.equals("")){   //사번만 입력
                                        int no=Integer.parseInt(sno);
                                        evo=edvo.getEmployeeNo(no);
                                }else if(sno.equals("") && !sname.equals(""))
                                        evo=edvo.getEmployeeName(sname);
                            }catch (Exception e){
                                System.out.println("e=["+e+"]");
                            }
                        if(evo != null ){ //해당 사원이 존재하지 않는다면 텍스트 필드를 초기화.
                                tf[0].setText(evo.getNo()+"");
                                tf[1].setText(evo.getName());
                                tf[2].setText(evo.getJobGrade());
                                tf[3].setText(evo.getDepartment()+"");
                                tf[4].setText(evo.getEmail());
                            }else {
                                JOptionPane.showMessageDialog(this, "검색 실패");
                            }
                 } else if(ae_type.equals(rsb.getText())) {
                        int size=caption.length;
                            //텍스트 필드를 초기화 한다.
                            for(int i = 0; i<size; i++) {
```

```
                                    tf[i].setText("");
                                }
                        }
                }
}
```

사원 전체를 보기 위한 TotalPane 클래스와 모델 객체
사원 전체를 보기 위해서는 JTable를 사용한다. JTable에 출력할 데이터는 모델 객체로 설계한다.

EmployModel.java

```
package exam01;

import java.util.ArrayList;
import javax.swing.table.AbstractTableModel;

public class EmployModel extends AbstractTableModel {
        Object [][]data;
        Object []columnName;

        EmployeeDAO emDao=new EmployeeDAO();
        EmployeeVO  emVo;
        ArrayList<String> title;
        ArrayList<EmployeeVO> list;

        public EmployModel(){
            //열의 개수와 행의 개수를 알아야 2차원 배열선언
            //테이블에서 컬럼이름을 얻어와서 1차원 배열 선언
            title=emDao.getColumnName();
            columnName = title.toArray();
            int columnCount=title.size();

            list=emDao.getEmployeetotal();
            int rowCount=list.size();

            data=new Object[rowCount][columnCount];

            for(int index=0; index<rowCount; index++){
                emVo=list.get(index);
                data[index][0]=emVo.getNo();
                data[index][1]=emVo.getName();
                data[index][2]=emVo.getJobGrade();
```

```
                data[index][3]=emVo.getDepartment();
                data[index][4]=emVo.getEmail();
            }
        }
        public int getColumnCount() {
            if(columnName==null)
                return 0;
            else
                return columnName.length;
        }
        public int getRowCount() {
            if(data==null)
                return 0;
            else
                return data.length;
        }
        public Object getValueAt(int rowIndex, int columnIndex) {
            return data[rowIndex][columnIndex];
        }
        public String getColumnName(int column) {
            return (String)columnName[column];
        }
}
```

사원 전체 보기를 위한 UI와 검색 처리

TotalPane.java

```
package exam01;

import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;

public class TotalPane extends JPanel {
        public TotalPane(){
            JTable table=new JTable(new EmployModel());
            add(new JScrollPane(table));
        }
}
```

```
package exam01;
```

```
import java.awt.Rectangle;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.JTextField;

public class TotalPane extends JPanel {
	private JButton button1 = new JButton("확인");
	private JLabel label1 = new JLabel();
	private JTextField textField1 = new JTextField();

	public TotalPane(){
		this.setLayout(null);
		//label1.setFont(new java.awt.Font("Monospaced", 0, 12));
		label1.setText("아이디");
		label1.setBounds(new Rectangle(32, 21, 41, 18));
		textField1.setBounds(new Rectangle(75, 21, 100, 23));
		button1.setBounds(new Rectangle(250, 21, 70, 23));

		add(label1);
		add(textField1);
		add(button1);

		JTable table=new JTable(new EmployModel());
		JScrollPane jp = new JScrollPane(table);
		jp.setBounds(new Rectangle(10, 100, 500, 400));
		add(jp);
	}
}
```

MainFrame 클래스

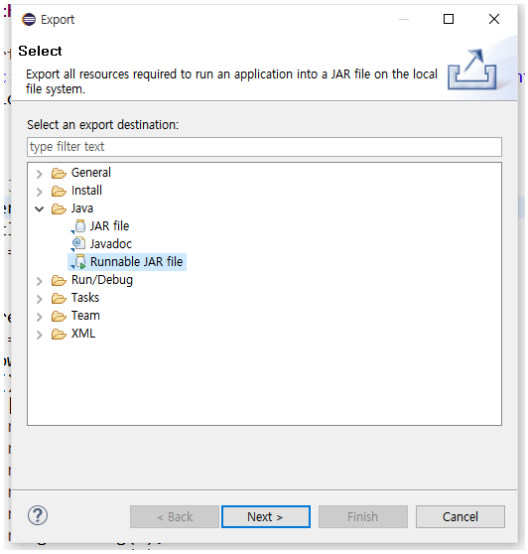
사원 정보, 사원 관리 프레임 객체는 사원 정보 입력 탭 객체와 사원 정보 조회 탭 객체를 속성으로 갖는다.

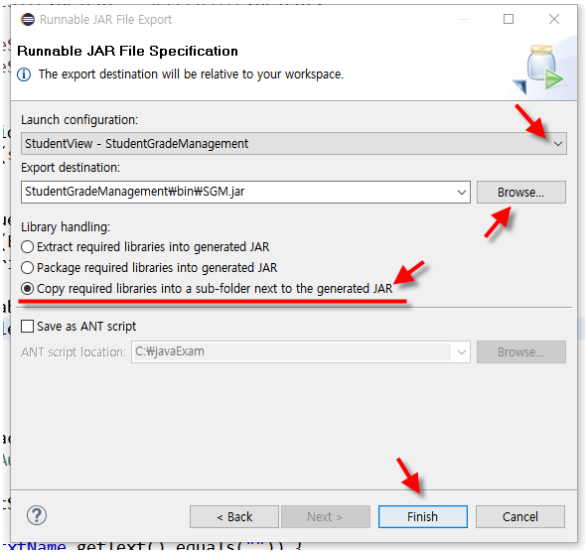
메소드	기 능
MainFrame()	생성자에서는 프레임에 사원 정보 입력 탭과 사원 정보 조회 탭 객체를 생성하고 프레임에 추가한다. 텝을 구현하기 위해 JTabbedPane 객체를 생성하고, JTabbedPane 객체인 tp에 사원 정보 입력 패널과 사원 정보 조회 패널을 추가한 후 프레임에 tp를 추가한다.
main()	메인 프레임을 활성화 시키고 하면에 출력한다.

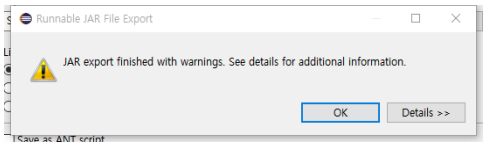
MainFrame.java

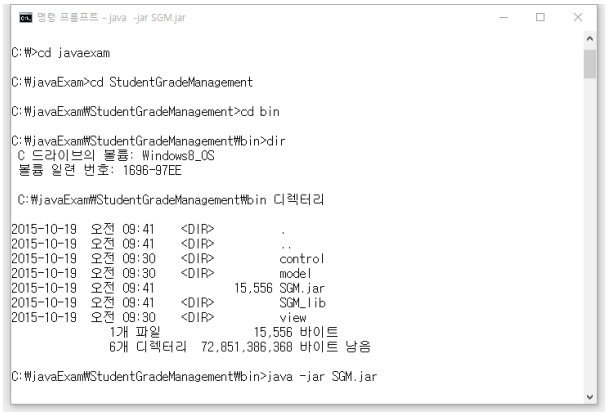
```
package exam01;

import java.awt.Dimension;
import javax.swing.JFrame;
import javax.swing.JTabbedPane;

public class MainFrame extends JFrame {
	private JTabbedPane tp;
	private AddPane ap;
	private FindPane fp;
	private TotalPane tpa;

	public MainFrame(){
		 //프레잉에 추가될 컴포넌트 초기화
		tp = new JTabbedPane();
		ap = new AddPane();
		fp = new FindPane();
		tpa = new TotalPane();

		//탭 추가
		tp.addTab("사원정보입력", ap);
		tp.addTab("사원정보조회", fp);
		tp.addTab("사원전체보기", tpa);

		//TabbedPane을 프레임에 추가
		getContentPane().add(tp);
		setTitle("사원 관리(JDBC 버전)");
		pack();
		setVisible(true);
		setDefaultCloseOperation(EXIT_ON_CLOSE);
	}
	public static void main(String[] args) {
			MainFrame mf = new MainFrame();
			mf.setSize(600, 600);
	}
}
```

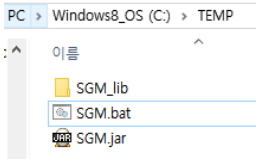
이클립스 없이 실행파일 만들기











메모장으로 SGM.bat 파일을 만들고 내용으로 java -jar SGM.JAR 을 기입한 후 저장한다.