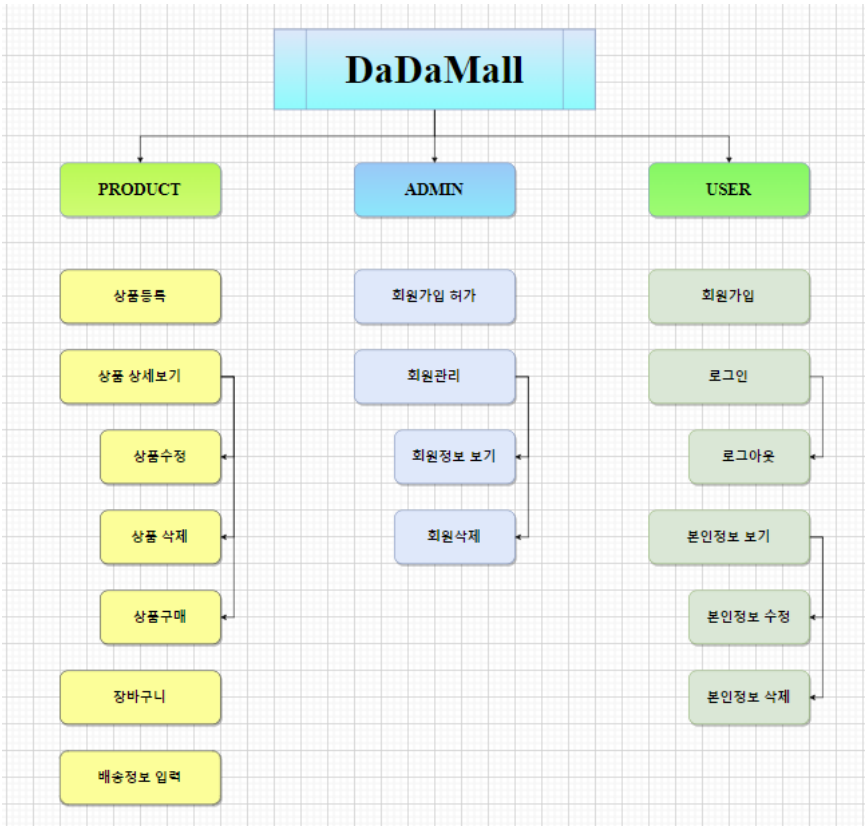
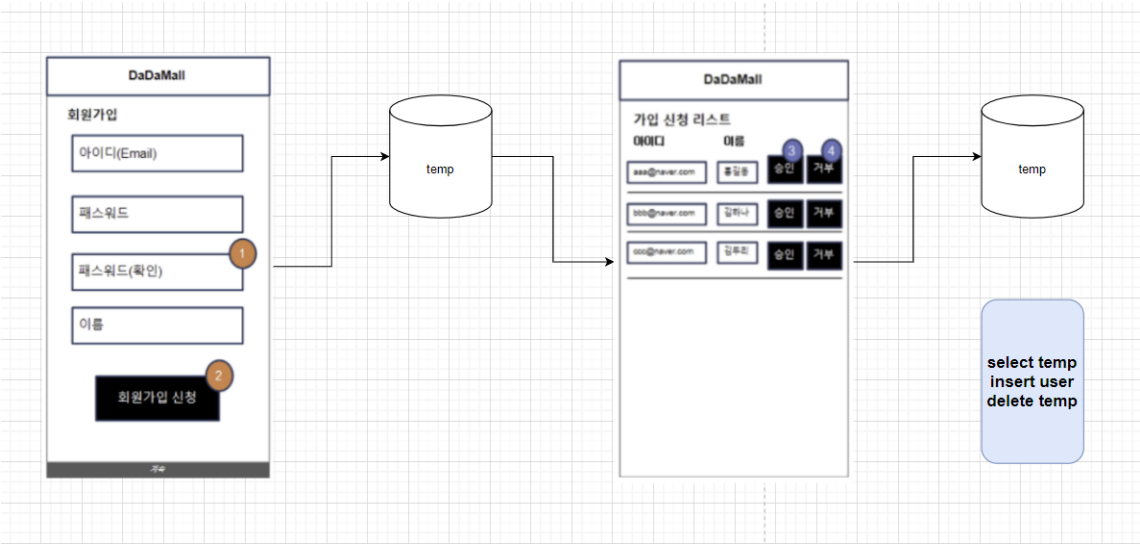
Class[DaDaMall4]

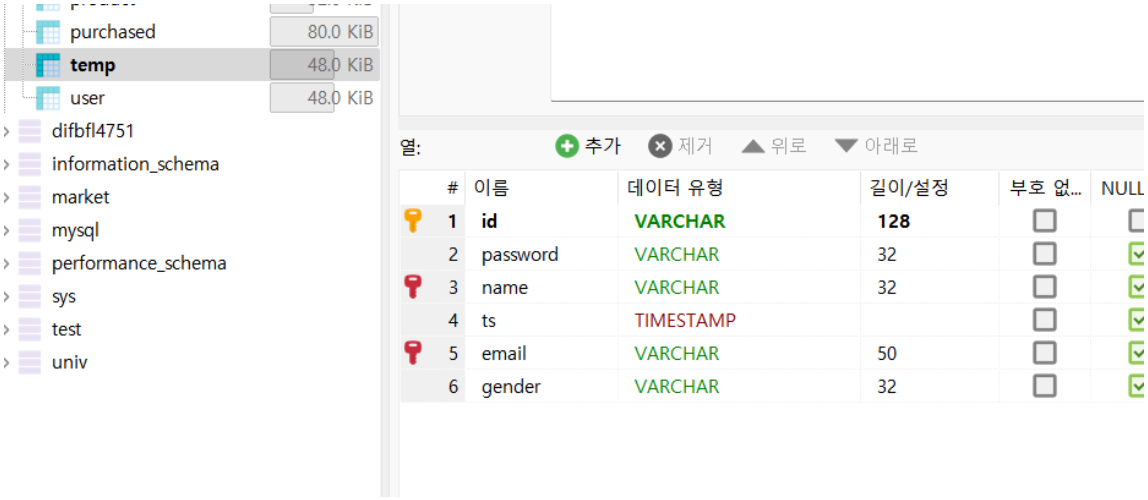
▼ Java - DaDaMall Project



▼ 회원가입 승인처리

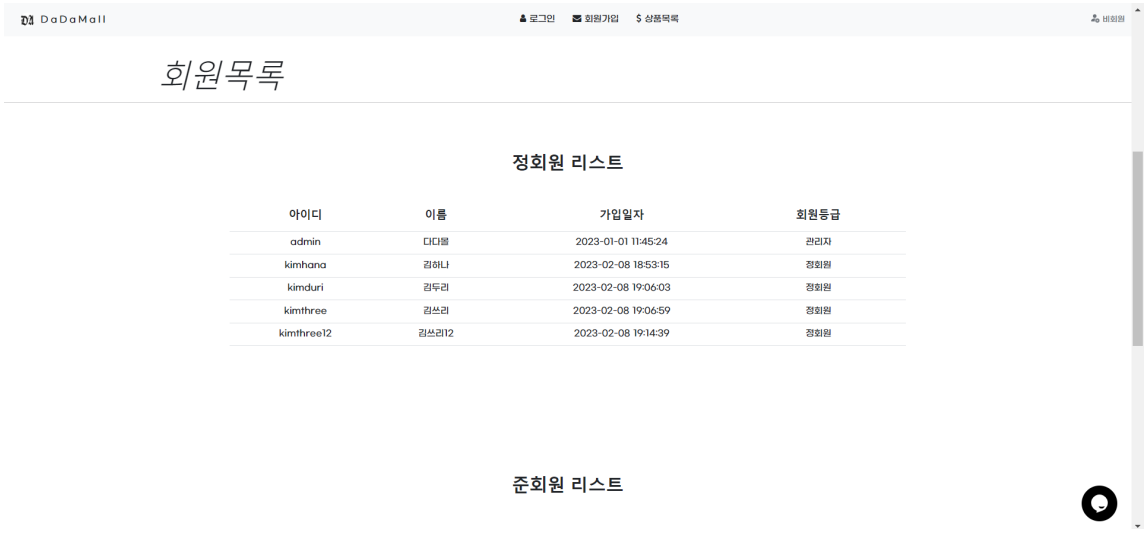


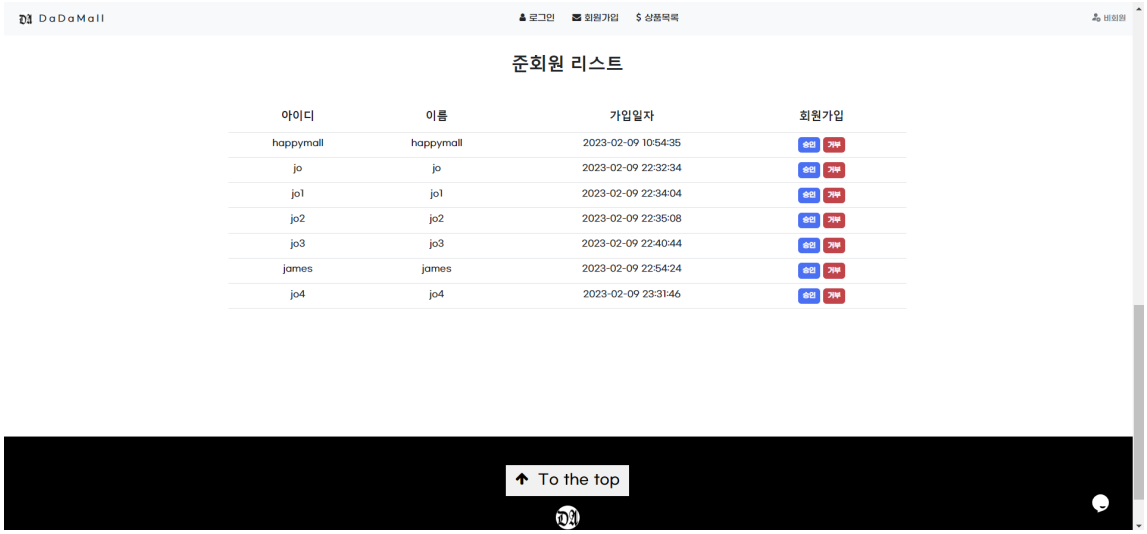
temp 테이블 생성(준회원 테이블)




```
CREATE TABLE `temp` (
  `id` VARCHAR(128) NOT NULL COLLATE 'utf8mb4_0900_ai_ci',
  `password` VARCHAR(32) NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci',
  `name` VARCHAR(32) NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci',
  `ts` TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP,
  `email` VARCHAR(50) NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci',
  `gender` VARCHAR(32) NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci',
  PRIMARY KEY (`id`) USING BTREE,
  UNIQUE INDEX `name` (`name`) USING BTREE,
  UNIQUE INDEX `email` (`email`) USING BTREE
)
ENGINE=InnoDB
;
```

JSP구성





userDAO.java

```
package jdbc;

import java.sql.*;
import java.util.*;

import javax.naming.NamingException;

import org.json.simple.JSONArray;
import org.json.simple.JSONObject;

import util.ConnectionPool;

public class userDAO {

  private static PreparedStatement pstmt;
  private static String sql;
  private static ResultSet rs;
  userDTO udto;
  private static Connection conn;

  public static boolean insert(String id, String password, String name, String email, String gender) throws SQLException, Namin

      try {
        sql = " INSERT INTO user (id, password, name, email, gender) "
            + " VALUES(?, ?, ?, ?, ?) ";

        conn = ConnectionPool.get();

        pstmt = conn.prepareStatement(sql);

        pstmt.setString(1, id);
        pstmt.setString(2, password);
        pstmt.setString(3, name);
        pstmt.setString(4, email);
        pstmt.setString(5, gender);

        int result = pstmt.executeUpdate();
        if (result == 1) {
          return true;
        }
      } catch (SQLException e) {
        e.printStackTrace();
      } finally {
        if(pstmt != null)
        pstmt.close();
        if(conn != null)
        conn.close();
      }

      return false;
    }

  public static ArrayList<userDTO> getAllList() throws SQLException, NamingException{

    ArrayList<userDTO> users = new ArrayList<userDTO>();
```

```
  try {
    sql = " SELECT * FROM user ORDER BY ts ASC ";

    conn = ConnectionPool.get();

    pstmt = conn.prepareStatement(sql);

    rs = pstmt.executeQuery();



    while(rs.next()) {
      users.add(new userDTO(rs.getString(1),
                rs.getString(2),
                rs.getString(3),
                rs.getString(4),
                rs.getString(5),
                rs.getString(6)));
    }

      return users;

  } catch (SQLException e) {
    e.printStackTrace();
  } finally {
    if(pstmt != null)
    pstmt.close();
    if(conn != null)
    conn.close();
  }

  return users;

  }

public static userDTO getOneList(String id) throws SQLException, NamingException{

  userDTO user = new userDTO();

  try {
    sql = " SELECT * FROM user where id=? ";

    conn = ConnectionPool.get();

    pstmt = conn.prepareStatement(sql);

    pstmt.setString(1, id);

    rs = pstmt.executeQuery();

    if (rs.next()) {
      user.setId((rs.getString("id")));
      user.setPassword((rs.getString("password")));
      user.setName((rs.getString("name")));
      user.setTs((rs.getString("ts")));
      user.setEmail((rs.getString("email")));
      user.setGender((rs.getString("gender")));

    }

    return user;


  } catch (SQLException e) {
    e.printStackTrace();
  } finally {
    if(pstmt != null)
    pstmt.close();
    if(conn != null)
    conn.close();
  }

  return user;

  }

public static boolean update(userDTO udto, String id) throws NamingException, SQLException {

  try {

    sql = "UPDATE user SET id=?, name=?, email=?, gender=? "
        + " WHERE id=? ";

    conn = ConnectionPool.get();

    pstmt = conn.prepareStatement(sql);
```

```
      pstmt.setString(1, id);
      pstmt.setString(2, udto.getName());
      pstmt.setString(3, udto.getEmail());
      pstmt.setString(4, udto.getGender());
      pstmt.setString(5, udto.getId());

      int result = pstmt.executeUpdate();
      if (result == 1) {
        return true;
      }
    } catch (SQLException e) {
      e.printStackTrace();
    } finally {
      if(pstmt != null)
      pstmt.close();
      if(conn != null)
      conn.close();
    }

    return false;

  }

  public static boolean delete(String id) throws NamingException, SQLException {

  try {

    sql = "DELETE from user where id=? ";

    conn = ConnectionPool.get();

    pstmt = conn.prepareStatement(sql);

    pstmt.setString(1, id);

    int result = pstmt.executeUpdate();
    if (result == 1) {
      return true;
    }
  } catch (SQLException e) {
    e.printStackTrace();
  } finally {
    if(pstmt != null)
    pstmt.close();
    if(conn != null)
    conn.close();
  }

  return false;

}

  //회원가입시 아이디가 이미 존재하는지 여부 확인
    public static boolean exist(String id) throws SQLException{

      String sql = "SELECT * FROM user WHERE id=?";
      ResultSet rs = null;
      Connection conn = null;
      PreparedStatement pstmt = null;
      try {



        conn = ConnectionPool.get();

        pstmt = conn.prepareStatement(sql);

        pstmt.setString(1, id);

        rs = pstmt.executeQuery();


        if(rs.next())
          return true;


      } catch (Exception e) {
        e.printStackTrace();
      } finally {
        if(pstmt != null)
        pstmt.close();
        if(conn != null)
        conn.close();
      }

      return false;
```

```
    }

    public static boolean existTemp(String id) throws SQLException{

      String sql = "SELECT * FROM temp WHERE id=?";
      ResultSet rs = null;
      Connection conn = null;
      PreparedStatement pstmt = null;
      try {



        conn = ConnectionPool.get();

        pstmt = conn.prepareStatement(sql);

        pstmt.setString(1, id);

        rs = pstmt.executeQuery();


        if(rs.next())
          return true;


      } catch (Exception e) {
        e.printStackTrace();
      } finally {
        if(pstmt != null)
        pstmt.close();
        if(conn != null)
        conn.close();
      }

      return false;
    }

  //회원 로그인
    public static int login(String id, String password) throws SQLException{

      String sql = "SELECT * FROM user WHERE id=?";
      try {

      int result;

      Connection conn = ConnectionPool.get();

      PreparedStatement pstmt = conn.prepareStatement(sql);

      pstmt.setString(1, id);

      ResultSet rs = pstmt.executeQuery();



      if(!rs.next()) {
        result = 1; //아이디가 존재하지 않는 경우
      }else if (!password.equals(rs.getString("password"))) { //아이디는 존재하지만 비번이 일치하지 않는 경우
        result = 2;
      }else {
        result = 0; //로그인 성공
      }

      return result;


      } catch (Exception e) {
        e.printStackTrace();
      } finally {
        if(pstmt != null)
        pstmt.close();
        if(conn != null)
        conn.close();
      }

      return 3;

//      if(!mpass.equals(rs.getString(2))) return 2; // 비번만 틀린 경우

    }

    public static int loginTemp(String id, String password) throws SQLException{

      String sql = "SELECT * FROM temp WHERE id=?";
      try {

      int result;
```

```
      Connection conn = ConnectionPool.get();

      PreparedStatement pstmt = conn.prepareStatement(sql);

      pstmt.setString(1, id);

      ResultSet rs = pstmt.executeQuery();


      if(!rs.next()) {
        result = 1; //아이디가 존재하지 않는 경우
      }else if (!password.equals(rs.getString("password"))) { //아이디는 존재하지만 비번이 일치하지 않는 경우
        result = 2;
      }else {
        result = 0; //로그인 성공
      }

      return result;

      } catch (Exception e) {
        e.printStackTrace();
      } finally {
        if(pstmt != null)
        pstmt.close();
        if(conn != null)
        conn.close();
      }

      return 3;

//      if(!mpass.equals(rs.getString(2))) return 2; // 비번만 틀린 경우

    }

  public static boolean inserttemp(String id, String password, String name, String email, String gender) throws SQLException, N

        try {
          sql = " INSERT INTO temp (id, password, name, email, gender) "
              + " VALUES(?, ?, ?, ?, ?) ";

          conn = ConnectionPool.get();

          pstmt = conn.prepareStatement(sql);

          pstmt.setString(1, id);
          pstmt.setString(2, password);
          pstmt.setString(3, name);
          pstmt.setString(4, email);
          pstmt.setString(5, gender);

          int result = pstmt.executeUpdate();
          if (result == 1) {
            return true;
          }
        } catch (SQLException e) {
          e.printStackTrace();
        } finally {
          if(pstmt != null)
          pstmt.close();
          if(conn != null)
          conn.close();
        }

        return false;
      }

  public static ArrayList<userDTO> getAlltemp() throws SQLException, NamingException{

    ArrayList<userDTO> users = new ArrayList<userDTO>();

    try {
      sql = " SELECT * FROM temp ORDER BY ts ASC ";

      conn = ConnectionPool.get();

      pstmt = conn.prepareStatement(sql);

      rs = pstmt.executeQuery();



      while(rs.next()) {
        users.add(new userDTO(rs.getString(1),
                  rs.getString(2),
                  rs.getString(3),
                  rs.getString(4),
```

```
                rs.getString(5),
                rs.getString(6)));
    }

      return users;

  } catch (SQLException e) {
    e.printStackTrace();
  } finally {
    if(pstmt != null)
    pstmt.close();
    if(conn != null)
    conn.close();
  }

  return users;

  }


public static userDTO getOneTemp(String id) throws SQLException, NamingException{

  userDTO user = new userDTO();

  try {
    sql = " SELECT * FROM temp where id=? ";

    conn = ConnectionPool.get();

    pstmt = conn.prepareStatement(sql);

    pstmt.setString(1, id);

    rs = pstmt.executeQuery();

    if (rs.next()) {
      user.setId((rs.getString("id")));
      user.setPassword((rs.getString("password")));
      user.setName((rs.getString("name")));
      user.setTs((rs.getString("ts")));
      user.setEmail((rs.getString("email")));
      user.setGender((rs.getString("gender")));

    }

    return user;


  } catch (SQLException e) {
    e.printStackTrace();
  } finally {
    if(pstmt != null)
    pstmt.close();
    if(conn != null)
    conn.close();
  }

  return user;

  }


public static boolean deleteTemp(String id) throws NamingException, SQLException {

try {

  sql = "DELETE from temp where id=? ";

  conn = ConnectionPool.get();

  pstmt = conn.prepareStatement(sql);

  pstmt.setString(1, id);

  int result = pstmt.executeUpdate();
  if (result == 1) {
    return true;
  }
} catch (SQLException e) {
  e.printStackTrace();
} finally {
  if(pstmt != null)
  pstmt.close();
  if(conn != null)
  conn.close();
}
```

```
  return false;

  }



}
```

userAllAJAX.jsp

```
<%@page import="jdbc.*"%>
<%@page import="java.util.*"%>
<%@page import="jdbc.userDAO"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>회원목록</title>
</head>
 <body class="text-center">
 <%@ include file="../includes/header.jsp"%>
  <header class="bgimg w3-display-container w3-grayscale-min" style="height: 80%" id="home">
 <img src="../images/스포츠이미지.jpg"  width="100%" height="90%">
  <div class="w3-display-bottomleft w3-center w3-padding-large w3-hide-small">
  </div>
  <div class="w3-display-middle w3-center">
    <span class="w3-text-white text-light" style="font-size:90px">DaDaMall</span>
  </div>
  <div class="w3-display-bottomright w3-center w3-padding-large">
  </div>
</header>
    <div class="col-md-6 px-0">
      <h1 class="display-4 fst-italic" style="padding-right: 220px;">회원목록</h1>
     </div>
        <hr class="featurette-divider">
          <br>
  <br>
  <br>



<main class="form-signin w-100 m-auto">


   <h2><b> 정회원 리스트 </b></h2>
     <br><br>
<table class="table table-hover" style="width: 60%; margin-left: auto; margin-right: auto;">
   <thead>
   <tr>
    <th scope="col"><h5><b>아이디</b></h5></th>
    <th scope="col"><h5><b>이름</b></h5></th>
    <th scope="col"><h5><b>가입일자</b></h5></th>
    <th><h5><b>회원등급</b></h5></th>
    <th></th>
    </tr>
  </thead>
   <tbody class="table table-hover" id="ajaxTable">



    </tbody>
    </table>



</main>

<main class="form-signin w-100 m-auto" style="padding-top: 200px; padding-bottom: 200px;">


   <h2><b> 준회원 리스트 </b></h2>
     <br><br>
<table class="table table-hover" style="width: 60%; margin-left: auto; margin-right: auto;">
   <thead>
   <tr>
    <th scope="col"><h5><b>아이디</b></h5></th>
    <th scope="col"><h5><b>이름</b></h5></th>
    <th scope="col"><h5><b>가입일자</b></h5></th>
    <th><h5><b>회원가입</b></h5></th>
    <th></th>
```

```
    </tr>
  </thead>
   <tbody class="table table-hover" id="ajaxTable2">



    </tbody>
    </table>



</main>




<%@ include file="../includes/footer.jsp"%>
</body>
  <script>

  function searchFunction() {

    $.ajax({
    type:'post',
    url:'userCheckAJAX.jsp',
    datatype : 'json',
    success:function(result){
      var str = result.split("__TEMP__");
      $("#ajaxTable").html(str[0]);
      $("#ajaxTable2").html(str[1]);


    }

    });

}




  window.onload = function() {
  searchFunction();
}

  $(document).on('click', '.regBtn', function(event) {
    $.ajax({
      type:'post',
      url:'tempCheckAJAX.jsp?check=승인&id=' + document.getElementById("tempId").innerText,
      success:function(result){
        searchFunction();
      }

      });
  });


  $(document).on('click', '.delBtn', function(event) {
    $.ajax({
      type:'post',
      url:'tempCheckAJAX.jsp?check=거부&id=' + document.getElementById("tempId").innerText,
      success:function(result){
        searchFunction();
      }

      });
  });
  </script>
</html>
```

tempCheckAJAX.jsp

```
<%@page import="java.util.*"%>
<%@page import="jdbc.*"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%

String id = request.getParameter("id");
String check = request.getParameter("check");
```

```
if(check.equals("승인")){

userDTO user = userDAO.getOneTemp(id);
userDAO.insert(id, user.getPassword(), user.getName(), user.getEmail(), user.getGender());
userDAO.deleteTemp(id);

}else{

  userDAO.deleteTemp(id);

}



%>
```

userCheckAJAX.jsp

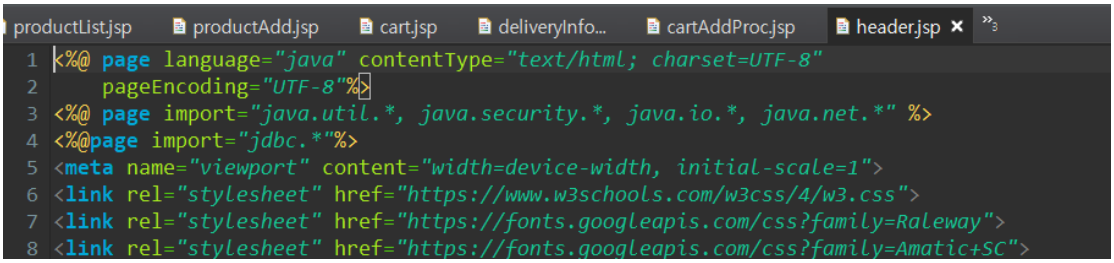
```
<%@page import="java.util.*"%>
<%@page import="jdbc.*"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%
ArrayList<userDTO> users = userDAO.getAllList();
ArrayList<userDTO> temps = userDAO.getAlltemp();


for(userDTO user : users) {%>
 <tr><td> <%=user.getId() %> </td>
<td> <%=user.getName() %> </td>
<td> <%=user.getTs() %> </td>
<% if(user.getId().equals("admin")){ %>
<td> 관리자 </td>
<%}else{%>
<td> 정회원 </td>
<%} %>
<td></td></tr>
<%}%>

__TEMP__

<%for(userDTO temp : temps) {%>
 <tr><td id="tempId"> <%=temp.getId() %> </td>
<td> <%=temp.getName() %> </td>
<td> <%=temp.getTs() %> </td>
<td> <span class="badge bg-primary regBtn" style="cursor : pointer;">승인</span> <span class="badge bg-danger delBtn" style="cur
<td></td></tr>
<%} %>
```

▼ 모바일 전용화면(Viewport)




```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

▼ Python - Inheritance, Exception

```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""

# # Inheritance  상속

# # 실제 세계와 동일한 의미로 사용된다. 마치 재산을 상속받는 것과 같은 의미이다.
# # 즉 다른 클래스의 기능을 물려받아 사용할 수 있다.


# class BigCal:

#   def __init__(self, first, second):
#     self.first = first
#     self.second = second

#   def add(self):
#     result = self.first + self.second
#     return result

#   def sub(self):
#     result = self.first - self.second
#     return result

#   def mul(self):
#     result = self.first * self.second
#     return result

#   def div(self):
#     result = self.first / self.second
#     return result

# class SmallCal(BigCal):
#   pass

# sc1 = SmallCal(3, 2) # sc1은 SmallCal로 생성하고 SmallCal 은 BigCal을 상속하기 때문에
#                      # BigCal 의 생성자의 규칙에 맞게 생성해야 한다.


# sc1.add()

# # 왜 상속을 사용할까

# # 보통 상속을 사용하면 기존 클래스는 그대로 놔두고 기능을 추가하거나 또는 기존 기능을 변경하고자 할때 사용된다.

# # 그냥 원래 클래스를 변경하면 되지 않을까? 라고 생각할 수 있으나 고급 사용 방식에서는 기존 클래스가
# # 라이브러리 형태로 제공되기 때문에 수정이 허용되지 않는 상황이 발생하게 된다.

# class MidCal(BigCal):
#   def pow(self):  # 기존 BigCal 에는 사칙연산만 있었다. 거기에 제곱을 계산하는 기능 추가.
#     result = self.first ** self.second
#     return result

# cal1 = BigCal(2,3)
# #cal1.pow()     # 부모가 자식의 자원을 사용할 수 없다.
# print(cal1.add())

# cal2 = MidCal()
# print(cal1.pow())     # 자식은 부모의 자원을 사용할 수 있다.
# print(cal1.add())


# # 상속은 기존 클래스의 내용은 그대로 두고 클래스의 기능을 확장시킬 때 주로 사용된다.



# # 파이썬은 자바와는 달리 다중상속을 지원한다.

# class A():
#     pass

# class B():
#     pass
```

```
# class C(A,B):
#     pass




# # 매서드 오버라이딩


# class BigCal:

#   def __init__(self, first, second):
#     self.first = first
#     self.second = second

#   def add(self):
#     result = self.first + self.second
#     return result

#   def sub(self):
#     result = self.first - self.second
#     return result

#   def mul(self):
#     result = self.first * self.second
#     return result

#   def div(self):
#     result = self.first / self.second
#     return result


# a = BigCal(2, 3)
# print(a.div())

# a = BigCal(2, 0)
# # print(a.div())  # ZeroDivisionError: division by zero


# # 매서드 오버라이딩

# class OkCal(BigCal):
#     def div(self):  # 부모 클래스의 div 매서드가 무시되고 자식의 같은 이름의 매서드가 사용된다.
#         if self.second == 0:
#             print("0으로는 나눌 수 없습니다.")
#         else:
#             return self.first / self.second


# c = OkCal(2, 0)
# print(c.div())


# # 클래스 변수

# # 객체 변수는 다른 객체들의 영향을 받지 않고 독립적으로 그 값을 유지한다는
# # 특징이 있다. 그렇지만 객체변수와는 그 성격이 완전히 다른 클래스 변수도 있다.

# class Family:
#     lastname = "김"

# # 클래스 변수는 공유된다.

# a = Family()
# b = Family()

# print(a.lastname)
# print(b.lastname)
# print(Family.lastname)

# Family.lastname = "박"

# print(a.lastname)
# print(b.lastname)


# # 클래스 변수는 중요하게 쓰이지 않는다. 객체변수는 훨씬 중요하다.
# # 실무에서도 클래스 변수보다 객체변수를 사용하는 비율이 훨씬 많다.

# # 클래스 변수 : 공유가능. 즉, 그 클래스로부터 생성된 모든 인스턴스들이 접근할 수 있다.
# # 객체 변수 : 클래스로부터 생성된 각각의 객체/인스턴스에 속해 있는 변수.
```

```
# 예외 처리 Exception

# 프로그램을 만들다 보면 수많은 오류 / 예외 / 에러 가 발생된다.
# 오류는 프로그램이 잘못 동작하는 것을 사전에 막아주는 일종의 배려로 볼 수도 있다.
# 파이썬은 오류. 즉, 예외를 어떻게 처리하는지 살펴 보자.

# 파이썬에서 자주 접하게 되는 예외
# ZeroDivisionError : 0으로 나누는 예외.
# IndexError : 없는 인덱스를 호출하였을 때 발생.


# ZeroDivisionError : 0으로 나누는 예외.
# 4 / 0


# IndexError : 없는 인덱스를 호출하였을 때 발생.
# l1 = [1, 2, 3, 4, 5]
# l1[5]



# 파이썬에서의 예외 처리 기본

# try:
#     수행구문
# except:
#     예외 발생 시 처리 구문


# try:
#     print(4 / 0)
# except:     # 모든 예외 처리
#     print('0으로는 나눌 수 없습니다.')


# try:
#     print(4 / 0)
# except ZeroDivisionError:     # 특정 예외 처리
#     print('0으로는 나눌 수 없습니다.')


# try:
#     print(4 / 0)
# except ZeroDivisionError as e:     # 특정 예외를 변수 'e'에 담는다.
#     print(e)


# # finally


# try:
#     print(4 / 0)
# except ZeroDivisionError as e:     # 특정 예외를 변수 'e'에 담는다.
#     print(e)
# finally:    # 예외 발생의 여부와는 관계없이 항상 마지막에 거친다.
#     print('프로그램을 종료합니다. ')



# # 여러 예외 처리

# try:
#     a = [1,2,3,4,5]
#     print(a[5])
#     4/0
# except ZeroDivisionError:
#     print('0으로는 나눌 수 없습니다.')
# except IndexError:
#     print('인덱싱 할 수 없습니다.')

#     # 인덱싱 오류가 먼저 발생했기 때문에 4/0 오류는 실행 되지 않는다.



# try:
#     a = [1,2,3,4,5]
#     print(a[5])
#     4/0
# except (ZeroDivisionError, IndexError) as e:
#     print(e)




# try else
```

```
# try:
#     실행구문
# except:
#     예외 발생시 처리 구문
# else:
#     오류가 없을 때만 실행되는 구문
# finally:
#     실행구문

# 예외 발생시   try -> except -> Finally
# 예외 미발생시 try -> else   -> Finally


# # 오류 발생시키기

# class Bird:
#     def fly(self):
#         raise NotImplementedError

#         # 파이썬에 미리 정의된 에러.
#         # 꼭 작성해야 하는 부분이 구현되지 않았을 경우
#         # 일부러 예외 상황이 아닌 상황에서도 예외 강제 발생시키기.


# class Eagle(Bird):
#     pass

# eagle = Eagle()
# eagle.fly()



# 예외 만들기

# class MyError(Exception):   # 파이썬 내장 클래스를 상속
#     pass

# def say_nick(nick):
#     if nick == "바보":
#         raise MyError()
#         return ""
#     print(nick)
#     return ""

# print(say_nick("천사"))
# print(say_nick("바보"))


# Quiz 1 사용자에게 숫자를 입력받아서 그 값을 num 이라는 변수에 저장하는 함수를 만드세요.
# 그리고 num 을 사용하여 1 부터 num 에 저장된 숫자까지 세는 또 다른 함수를 정의하세요.
# Hint : 함수 3개가 필요. 실행을 위한 함수.



# def input_num():
#     num = input('숫자 입력 > ')
#     return num

# def count(num):
#     for i in range(1, num + 1):
#         print(i)

#     # n = 1
#     # while n <= num:
#     #     print(n, end=" ")
#     #     n += 1

# def main(num):
#     input_num()
#     count(num)


# main(3)


# import random


# Quiz 2 낮은 숫자와 높은 숫자를 입력받아
# 두 값 사이의 임의의 숫자를 생성하여 comp_num 이르는 변수에 저장하는 pick_num 함수를 정의하세요.



# '하나의 숫자를 생각하세요' 라는 메세지를 출력하고 사용자가 생각한 숫자를 입력받는
#  first_guess 함수를 정의하세요.
```

```
# 사용자가 입력한 숫자와 comp_num 이 같은지 확인하여 같다면 '정답'이라는 메세지를 출력하고
# 틀리면 그 입력한 값이 너무 낮은지 너무 높은지 알려주고 다시 입력하도록 하여
# 숫자를 맞출 때까지 계속 반복하는 함수를 만드세요.

# def pick_num():
#     num1 = int(input("낮은 숫자를 하나 입력하세요. > "))
#     num2 = int(input("높은 숫자를 하나 입력하세요. > "))
#     comp_num = random.randint(num1, num2)
#     return comp_num

# def first_guess():
#     guess = int(input("하나의 숫자를 생각하세요. > "))
#     return guess

# def check_answer(comp_num, guess):
#     try_again = True
#     while try_again == True:
#         if comp_num == guess:
#             print("정답")
#             try_again = False
#         elif comp_num < guess:
#             guess = int(input("너무 높습니다. 다시 시도하세요."))
#         else:
#             guess = int(input("너무 낮습니다. 다시 시도하세요."))

# def main():
#    comp_num = pick_num()
#    guess = first_guess()
#    check_answer(comp_num, guess)

# main()


# 세 개의 숫자를 입력받아 가장 큰 수를 출력하는 print_max 함수를 정의하라.
#  단 if 문을 사용해서 수를 비교하라.


# def print_max(a, b, c):
#     big = a
#     if b > big:
#         big = b
#     if c > big:
#         big = c
#     return big

# print(print_max(1, 2, 3))



# 입력된 문자열을 역순으로 출력하는 print_reverse 함수를 정의하라.

# def print_reverse(str):
#     return str[::-1]


# print(print_reverse("hi"))
```