LAB1

NHOM 3 - SWT301

Bug List					
Project Name: LaptopOnlineStore					
Tools : Sonarqube					
Enviroment: Netbeans IDE 13					
Issue Number	Method test	Issue Name	Level	Status	Note
1	BrandDAO.java	Multiple variables should not be declared on the same line	Minor	Done	line 63
2	ProductDAO.java	String literals should not be duplicated	Critical	Done	
3	ProductDAO.java	Nested blocks of code should not be left empty	Major	Done	line 75
4	ProductDisplayDAO.jsp	Standard outputs should not be used directly to log anything	Major	Done	line 62
5	Account.java	Local variables should not shadow class fields	Major	Done	line 233-238
6	HomeController.java	Sections of code should not be commented ou	Major	Done	line 54
7	CartController.java	Unused assignments should be removed	Major	Done	line 46, 47
8	Account.java	Conditionally executed code should be reachable	Major	Done	line 53
9	RegisterControl.java	Regular expressions should not overflow the stack	Major	Done	line 53
10	InsertTypeProController.java	URIs should not be hardcoded	Minor	Done	linr 124
11	ShopController.java	Local variable and method parameter names should comply with a naming convention	Minor	Done	line 38
12	ShopController.java	Unused local variables should be removed	Minor	Done	line 5
13	SupplierDAO.java	Methods should not be empty	Critical	Done	line 17
14	CartController.java	Cognitive Complexity of methods should not be too high	Critical	Done	
15	home.jsp	“<html>” element should have a language attribute	Major	Done	line 9
16	ProductTypeDAO.java	Local variables should not be declared and then immediately returned or thrown	Minor	Done	
17	Cart.java	Field names should comply with a naming convention	Minor	Done	
18	ProductType.java	Unnecessary imports should be removed	Minor	Done	
19	ProductDAO.java	"close()" calls should not be redundant	Minor	Done	line 154
20	CartController.java	rack uses of "TODO" tags	Info	Done	line 30

Issue 1:

Before

```java
55  public Product getProductBySeri(String seri) {
56      Product p = new Product();
57      xSql = "select ProductID, price, sell, Guarantee, images from Product WHERE Seri =?";
58      try {
59          ps = con.prepareStatement(xSql);
60          ps.setString(1, seri);
61          rs = ps.executeQuery();
62          String img;
            int pid, guarantee, sell;
64          float price;
65          while (rs.next()) {
66              pid = rs.getInt("ProductID");
67              img = rs.getString("images");
68              guarantee = rs.getInt("Guarantee");
69              sell = rs.getInt("sell");
70              price = rs.getFloat("price");
71              p = new Product(seri, img, pid, guarantee, sell, price);
72          }
73          rs.close();
74          ps.close();
        } catch (Exception e) {
            e.printStackTrace();
77      }
78      return (p);
79  }
```

After

```java
55  public Product getProductBySeri(String seri) {
56      Product p = new Product();
57      xSql = "select ProductID, price, sell, Guarantee, images from Product WHERE Seri =?";
58      try {
59          ps = con.prepareStatement(xSql);
60          ps.setString(1, seri);
61          rs = ps.executeQuery();
62          String img;
63          int pid;
64          int guarantee;
65          int sell;
66          float price;
67          while (rs.next()) {
68              pid = rs.getInt("ProductID");
69              img = rs.getString("images");
70              guarantee = rs.getInt("Guarantee");
71              sell = rs.getInt("sell");
72              price = rs.getFloat("price");
73              p = new Product(seri, img, pid, guarantee, sell, price);
74          }
75          rs.close();
76          ps.close();
        } catch (Exception e) {
            e.printStackTrace();
79      }
```

Issue 2:

Before

```java
public List<Course> listAllCourse() {
    String sql = "SELECT c.[ID]\n"
            + "        ,[Course_name]\n"
            + "        ,c.[Description]\n"
            + "        ,Type_Name\n"
            + "        ,[Price]\n"
            + "        ,[Public_Date]\n"
            + "        ,[Mentor_ID]\n"
            + "        ,[Image]\n"
            + "        ,c.[Status]\n"
            + "  FROM [dbo].[Course] c\n"
            + "  JOIN CourseType ct ON c.Type_ID = ct.ID";
    List<Course> lc = new ArrayList<>();
    try {
        PreparedStatement st = connection.prepareStatement(sql);
        ResultSet rs = st.executeQuery();
        while (rs.next()) {
            Course course = new Course();
            course.setID(rs.getInt(1));
            course.setCourse_name(rs.getString(2));
            course.setDescription(rs.getString(3));
            course.setType_name(rs.getString(4));
            course.setPrice(rs.getDouble(5));
            course.setPublic_date(rs.getString(6));
            course.setMentorID(rs.getInt(7));
            course.setImage(rs.getString(8));
            course.setStatus(rs.getInt(9));
            lc.add(course);
        }
    } catch (Exception e) {
        System.out.println(e);
    }
    return lc;
}
```

After

```java
ublic List<Course> listAllCourse() {
tring sql = "SELECT c.[ID], [Course_name], c.[Description], Type_Name, [Price], [Public_Date], [Mentor_ID], [Image], c.[Status] " +
            "FROM [dbo].[Course] c JOIN CourseType ct ON c.Type_ID = ct.ID";
ist<Course> lc = new ArrayList<>();
ry {
    PreparedStatement st = connection.prepareStatement(sql);
    ResultSet rs = st.executeQuery();
    while (rs.next()) {
        Course course = new Course();
        course.setID(rs.getInt(1));
        course.setCourse_name(rs.getString(2));
        course.setDescription(rs.getString(3));
        course.setType_name(rs.getString(4));
        course.setPrice(rs.getDouble(5));
        course.setPublic_date(rs.getString(6));
        course.setMentorID(rs.getInt(7));
        course.setImage(rs.getString(8));
        course.setStatus(rs.getInt(9));
        lc.add(course);
    }
 catch (Exception e) {
    System.out.println(e);

turn lc;
```

Issue 3:

Before

```java
54
55  public Product getProductBySeri(String seri) {
56      Product p = new Product();
57      xSql = "select ProductID, price, sell, Guarantee, images from Product WHERE Seri =?";
58      try {
59          ps = con.prepareStatement(xSql);
60          ps.setString(1, seri);
61          rs = ps.executeQuery();
62          String img;
63          int pid;
64          int guarantee;
65          int sell;
66          float price;
67          while (rs.next()) {
68              pid = rs.getInt("ProductID");
69              img = rs.getString("images");
70              guarantee = rs.getInt("Guarantee");
71              sell = rs.getInt("sell");
72              price = rs.getFloat("price");
73              p = new Product(seri, img, pid, guarantee, sell, price);
74          }
75          rs.close();
76          ps.close();
77      } catch (Exception e) {
78      }
79      return (p);
80  }
```

After

```java
55  public Product getProductBySeri(String seri) {
56      Product p = new Product();
57      xSql = "select ProductID, price, sell, Guarantee, images from Product WHERE Seri =?";
58      try {
59          ps = con.prepareStatement(xSql);
60          ps.setString(1, seri);
61          rs = ps.executeQuery();
62          String img;
63          int pid;
64          int guarantee;
65          int sell;
66          float price;
67          while (rs.next()) {
68              pid = rs.getInt("ProductID");
69              img = rs.getString("images");
70              guarantee = rs.getInt("Guarantee");
71              sell = rs.getInt("sell");
72              price = rs.getFloat("price");
73              p = new Product(seri, img, pid, guarantee, sell, price);
74          }
75          rs.close();
76          ps.close();
77      } catch (Exception e) {
78          e.printStackTrace();
79      }
80      return (p);
81  }
```

Issue 4:

Before

```java
42  public List<ProductDisplay> getNewestProductByCate(int sid) {
43      xSql = "Select top 10 p.Seri, pt.productName, s.supplierName, p.images, p.price from
44      List<ProductDisplay> t = new ArrayList<>();
45      try {
46          ps = con.prepareStatement(xSql);
47          ps.setInt(1, sid);
48          rs = ps.executeQuery();
          String seri, pname, sname, img;
50          float price;
51          ProductDisplay x;
52          while (rs.next()) {
53              seri = rs.getString(1);
54              pname = rs.getString(2);
55              sname = rs.getString(3);
56              img = rs.getString(4);
57              price = rs.getFloat(5);
58              x = new ProductDisplay(seri, pname, sname, img, price);
59              t.add(x);
60          }
        } catch (Exception e) {
            System.err.println("readCartDetail"+e.getMessage());
63      }
64      return (t);
65  }
```

After

```java
42  public List<ProductDisplay> getNewestProductByCate(int sid) {
43      xSql = "Select top 10 p.Seri, pt.productName, s.supplierName, p.images, p.price from
44      List<ProductDisplay> t = new ArrayList<>();
45      try {
46          ps = con.prepareStatement(xSql);
47          ps.setInt(1, sid);
48          rs = ps.executeQuery();
          String seri, pname, sname, img;
50          float price;
51          ProductDisplay x;
52          while (rs.next()) {
53              seri = rs.getString(1);
54              pname = rs.getString(2);
55              sname = rs.getString(3);
56              img = rs.getString(4);
57              price = rs.getFloat(5);
58              x = new ProductDisplay(seri, pname, sname, img, price);
59              t.add(x);
60          }
        } catch (Exception e) {
            e.printStackTrace();
63      }
64      return (t);
65  }
```

Issue 5:

Before

```java
224     public Account listAccount(String username) {
225         try {
226             String strSelect = "select * from Account "
227                     + "where [AccountName] = ? ";
228
229             pstm = cnn.prepareStatement(strSelect);
230             pstm.setString(1, username);
231             rs = pstm.executeQuery();
232             while (rs.next()) {
                    String acname = rs.getString(1);
                    String pass = rs.getString(2);
                    String per = rs.getString(3);
                    String fname = rs.getString(4);
                    String lname = rs.getString(5);
                    String email = rs.getString(6);
239                 return new Account(acname, pass, per, fname, lname, email);
240
241             }
            } catch (Exception e) {
                System.out.println("listAccount:" + e.getMessage());
244         }
245         return null;
246     }
```

After

```java
    public Account listAccount(String username) {
    try {
        String strSelect = "SELECT * FROM Account "
                + "WHERE [AccountName] = ? ";

        pstm = cnn.prepareStatement(strSelect);
        pstm.setString(1, username);
        rs = pstm.executeQuery();
        while (rs.next()) {
            String accountName = rs.getString(1);
            String password = rs.getString(2);
            String permission = rs.getString(3);
            String firstName = rs.getString(4);
            String lastName = rs.getString(5);
            String acEmail = rs.getString(6);
            return new Account(accountName, password, permission, firstName, lastName, acEmail);
        }
    } catch (Exception e) {
        System.out.println("listAccount: " + e.getMessage());
    }
    return null;
}
```

Issue 6:

Before

```
53          //setting cart
//            if (session.getAttribute("acc") != null) {
55  //            Account acc = (Account) session.getAttribute("acc");
56  //            if (cartDao.getCartByAccount(acc.getAcId()) != null) {
57  //                List<Cart> carts = cartDao.getCartByAccount(acc.getAcId());
58  //                session.setAttribute("Cart", carts);
59  //            }
60  //            }
61          //Set attribute
62          session.setAttribute("supList", supList);
```

After

```
//setting cart
if (session.getAttribute("acc") != null) {
    Account acc = (Account) session.getAttribute("acc");
    if (cartDao.getCartByAccount(acc.getAcId()) != null) {
        List<Cart> carts = cartDao.getCartByAccount(acc.getAcId());
        session.setAttribute("Cart", carts);
    }
}
//Set attribute
session.setAttribute("supList", supList);
```

Issue 7:

Before

```
OrderDAO orderDao = new OrderDAO();
OrderDetailDAO odDao = new OrderDetailDAO();
List<Cart> carts = (List<Cart>) session.getAttribute("Cart");
```

After

```
HttpSession session = request.getSession();
List<Cart> carts = (List<Cart>) session.getAttribute("Cart");
```

Issue 8:

Before

```
52          Account3 acc = new Account3(user, pass, per, fname, lname, email, phone, address);
            if (acc != null) {
54              request.setAttribute("username", user);
55              request.setAttribute("pass", pass);
56              request.setAttribute("repass", repass);
57              request.setAttribute("fname", fname);
58              request.setAttribute("lname", lname);
59              request.setAttribute("email", email);
60              request.setAttribute("phone", phone);
61              request.setAttribute("address", address);
```

After

```
52
53
54              request.setAttribute("username", user);
55              request.setAttribute("pass", pass);
56              request.setAttribute("repass", repass);
57              request.setAttribute("fname", fname);
58              request.setAttribute("lname", lname);
59              request.setAttribute("email", email);
60              request.setAttribute("phone", phone);
61              request.setAttribute("address", address);
```

Issue 9:

Before

```
52
54
Pattern checkemail = Pattern.compile("^[\\w-_\\.+]*[\\w-_\\.]\\@([\\w]+\\.)+[\\w]+[\\w]$");
```

After

```
52
53
54
55
Pattern checkemail = Pattern.compile("^[\\w.+]*[\\w.]@[\\w.]+$");
```

Issue 10:

Before

```
123
124        String savePath = getServletContext().getRealPath("/images");
125
```

After

```
124
           String savePath = "C:\\Users\\win\\Downloads\\DemoProject\\DemoProject\\web\\images";
126
```

Issue 11:

Before

```
35        HttpSession session = request.getSession();
36        SupplierDAO supDao = new SupplierDAO();
37        ProductDisplayDAO pdDao = new ProductDisplayDAO();
          Pagination Page;
```

After

```
35        HttpSession session = request.getSession();
36        SupplierDAO supDao = new SupplierDAO();
37        ProductDisplayDAO pdDao = new ProductDisplayDAO();
38        Pagination page;
```

Issue 12:

Before

```
List<ProductDisplay> product = pdDao.getAllProductDisplay();
List<ProductDisplay> topSell = pdDao.getTopSelling();
List<Cart> carts = (List<Cart>) session.getAttribute("Cart");
```

After

```
List<ProductDisplay> product = pdDao.getAllProductDisplay();
List<ProductDisplay> topSell = pdDao.getTopSelling();
```

Issue 13:

Before

```
16  public class SupplierDAO extends MyDAO{
        public List<Supplier> getSupplier(){
18
        }
```

After

```
public class SupplierDAO extends MyDAO{
    public List<Supplier> getSupplier(){
        List<Supplier> t = new ArrayList<>();
        Supplier s = null;
        xSql = "select supplierID, supplierName from Supplier";
        try {
            ps = con.prepareStatement(xSql);
            rs = ps.executeQuery();
            int sid;
            String sname;
            while (rs.next()) {
                sid = rs.getInt("supplierID");
                sname = rs.getString("supplierName");
                s = new Supplier(sid, sname);
                t.add(s);
            }
            rs.close();
            ps.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return (t);
    }
```

Issue 14:

Before

```java
43      @Override
44      protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
45          HttpSession session = request.getSession();
46          List<Cart> carts = (List<Cart>) session.getAttribute("Cart");
47
48          if (request.getParameter("del") != null) {
49              String seri = request.getParameter("seri");
50              List<Cart> itemsToRemove = new ArrayList<>();
51              for (Cart c : carts) {
52                  if (c.getSeri().equals(seri)) {
53                      itemsToRemove.add(c);
54                  }
55              }
56              carts.removeAll(itemsToRemove);
57              session.setAttribute("Cart", carts);
58              response.sendRedirect("home");
59          }
```

After

```java
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    HttpSession session = request.getSession();
    List<Cart> carts = (List<Cart>) session.getAttribute("Cart");

    if (request.getParameter("del") != null) {
        deleteCartItem(request, carts);
        response.sendRedirect("home");
    }
}

private void deleteCartItem(HttpServletRequest request, List<Cart> carts) {
    String seri = request.getParameter("seri");
    List<Cart> itemsToRemove = new ArrayList<>();
    for (Cart c : carts) {
        if (c.getSeri().equals(seri)) {
            itemsToRemove.add(c);
        }
    }
    carts.removeAll(itemsToRemove);
    request.getSession().setAttribute("Cart", carts);
}
```

Issue 15:

Before

```
8   <%@page import= "java.util.*" %>
    <html>
10      <head>
```

After

```
8   <%@page import= "java.util.*" %>
    <html lang="en">
10      <head>
```

Issue 16:

Before

```java
220    public int getMaxProductID(){
221        xSql = "select MAX(productID) as maxID from ProductType";
222        try {
223            ps = con.prepareStatement(xSql);
224            rs = ps.executeQuery();
225            while (rs.next()) {
               int pid = rs.getInt("maxID");
227            return pid;
228            }
229            rs.close();
230            ps.close();
           } catch (Exception e) {
               e.printStackTrace();
233        }
234        return -1;
235    }
```

After

```java
220    public int getMaxProductID(){
221        int pid=0;
222        xSql = "select MAX(productID) as maxID from ProductType";
223        try {
224            ps = con.prepareStatement(xSql);
225            rs = ps.executeQuery();
226            while (rs.next()) {
227                pid = rs.getInt("maxID");
228            }
229            rs.close();
230            ps.close();
           } catch (Exception e) {
               e.printStackTrace();
233        }
234        return (pid);
235    }
```

Issue 17:

Before

```
11  public class Cart {
    private String Seri;
13  private String pName;
14  private String img;
15  private int quantity;
16  private float price;
17
```

After

```
11  public class Cart {
12  private String seri;
13  private String pName;
14  private String img;
15  private int quantity;
16  private float price;
17
```

Issue 18:

Before

```
6
import java.sql.Date;
8
/**
10 *
11 * @author win
12 */
13 public class ProductType {
    int pid, sid, quantity;
```

After

```
6
7 /**
8 *
9 * @author win
10 */
11 public class ProductType {
```

Issue 19:

Before

```java
138 public int insert(Product p, Connection con) throws SQLException {
        int row = 0;
140     xSql = "insert into Product (Seri, ProductID, price, CreateDate, ModifiedDate, "
141             + "CreateBy, ModifiedBy, sell, Guarantee, images) values (?,?,?,?,?,?,?,?,?,?)";
        try(PreparedStatement ps = con.prepareStatement(xSql)) {
143         ps.setString(1, p.getSeri());
144         ps.setInt(2, p.getPid());
145         ps.setFloat(3, p.getPrice());
146         ps.setDate(4, p.getCredate());
147         ps.setDate(5, p.getModdate());
148         ps.setString(6, p.getCreby());
149         ps.setString(7, p.getModby());
150         ps.setInt(8, p.getSell());
151         ps.setInt(9, p.getGuarantee());
152         ps.setString(10, p.getImg());
153         row = ps.executeUpdate();
            ps.close();
155     }
156     return row;
157 }
```

After

```java
138 public int insert(Product p, Connection con) throws SQLException {
        int row = 0;
140     xSql = "insert into Product (Seri, ProductID, price, CreateDate, ModifiedDate, "
141             + "CreateBy, ModifiedBy, sell, Guarantee, images) values (?,?,?,?,?,?,?,?,?,?)";
        try(PreparedStatement ps = con.prepareStatement(xSql)) {
143         ps.setString(1, p.getSeri());
144         ps.setInt(2, p.getPid());
145         ps.setFloat(3, p.getPrice());
146         ps.setDate(4, p.getCredate());
147         ps.setDate(5, p.getModdate());
148         ps.setString(6, p.getCreby());
149         ps.setString(7, p.getModby());
150         ps.setInt(8, p.getSell());
151         ps.setInt(9, p.getGuarantee());
152         ps.setString(10, p.getImg());
153         row = ps.executeUpdate();
154     }
155     return row;
156 }
```

Issue 20:

Before

```
26      protected void processRequest(HttpServletRequest request, HttpServletResponse response)
27              throws ServletException, IOException {
28          response.setContentType("text/html;charset=UTF-8");
29          try ( PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
31              out.println("<!DOCTYPE html>");
32              out.println("<html>");
33              out.println("<head>");
34              out.println("<title>Servlet cart</title>");
35              out.println("</head>");
36              out.println("<body>");
37              out.println("<h1>Servlet cart at " + request.getContextPath() + "</h1>");
38              out.println("</body>");
39              out.println("</html>");
40          }
41      }
```

After

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try ( PrintWriter out = response.getWriter()) {
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet cart</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Servlet cart at " + request.getContextPath() + "</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

Thank You !