正　誤　表

令和5年4月16日実施

情報処理安全確保支援士試験　午後Ⅰ　問題

ページ	問題番号	行	誤	正	訂正の内容
6	1	下から6行目	図5中から選び	図4中から選び	下線部分を訂正する。

問1　Webアプリケーションプログラム開発に関する次の記述を読んで，設問に答えよ。

G社は，システム開発を行う従業員100名のSI企業である。このたび，オフィス用品を販売する従業員200名のY社から，システム開発を受託した。開発プロジェクトのリーダーには，G社の開発課のD主任が任命され，メンバーには，開発課から，Eさんと新人のFさんが任命された。G社では，セキュリティの品質を担保するために，プログラミング完了後にツールによるソースコードの静的解析を実施することにしている。

〔受託したシステムの概要〕

受託したシステムには，Y社の得意先がオフィス用品を注文する機能，Y社とY社の得意先が注文履歴を表示させる機能，Y社とY社の得意先が注文番号を基に注文情報を照会する機能（以下，注文情報照会機能という），Y社とY社の得意先が納品書のPDFファイルをダウンロードする機能などがある。

〔ツールによるソースコードの静的解析〕

プログラミングが完了し，ツールによるソースコードの静的解析を実施したところ，Fさんが作成した納品書PDFダウンロードクラスのソースコードに問題があることが分かった。納品書PDFダウンロードクラスのソースコードを図1に，静的解析の結果を表1に示す。

```
    (省略)  //package宣言，import宣言など
1:  public class DeliverySlipBL {
2:    private static final String PDF_DIRECTORY = "/var/pdf";  //PDFディレクトリ定義
      (省略)  //変数宣言など
3:    public DeliverySlipBean getDeliverySlipPDF(String inOrderNo, Connection conn) {
        (省略)  //変数宣言など
4:      DeliverySlipBean deliverySlipBean = new DeliverySlipBean();
5:      try {
          /* 検索用SQL文作成 */
6:        String sql = "SELECT ";
7:        sql = sql + (省略);  //抽出項目，テーブル名など
```

図1　納品書PDFダウンロードクラスのソースコード

```
 8:        sql = sql + " WHERE head.order_no = '" + inOrderNo + "' ";
 9:        sql = sql + (省略);  //抽出条件の続き
10:        Statement stmt = conn.createStatement();
11:        ResultSet resultObj = stmt.executeQuery(sql);
           (省略)  //注文情報の存在チェック（存在しないときはnullを返してメソッドを終了）
12:        String clientCode = resultObj.getString("client_code");  //得意先コード取得
13:        File fileObj = new File(PDF_DIRECTORY + "/" + clientCode +  "/" + "DeliverySlip"
   + inOrderNo + ".pdf");
           (省略)  //PDFファイルが既に存在しているかの確認など
14:        BufferedInputStream in = new BufferedInputStream(new FileInputStream(fileObj));
15:        byte[] buf = new byte[in.available()];
16:        in.read(buf);
17:        deliverySlipBean.setFileByte(buf);
18:      } catch (Exception e) {
           (省略)  //エラー処理（ログ出力など）
19:      }
20:      return deliverySlipBean;
21:    }
       (省略)
```

図1　納品書PDFダウンロードクラスのソースコード（続き）

表1　静的解析の結果

項番	脆弱性	指摘箇所	指摘内容
1	SQLインジェクション	（省略）	（省略）
2	ディレクトリトラバーサル	[a] 行目	ファイルアクセスに用いるパス名の文字列作成で，利用者が入力したデータを直接使用している。
3	確保したリソースの解放漏れ	（省略）	変数stmt，変数resultObj，変数 [b] が指すリソースが解放されない。

この解析結果を受けて，Fさんは，Eさんの指導の下，ソースコードを修正した。表1の項番1について図1の8行目から11行目を図2に示すソースコードに修正した。項番2と項番3についてもソースコードを修正した。

```
sql = sql + " [ c ] ";
sql = sql + (省略);  //抽出条件の続き
[ d ] ;
stmt.setString(1, inOrderNo);
ResultSet resultObj = stmt.executeQuery();
```

図2　納品書PDFダウンロードクラスの修正後のソースコード

再度，ツールによるソースコードの静的解析が実施され，表1の指摘は解消していることが確認された。

〔システムテスト〕

システムテストを開始したところ，注文情報照会機能において不具合が見つかった。この不具合は，ある得意先の利用者 ID でログインして画面から注文番号を入力すると，別の得意先の注文情報が出力されるというものであった。なお，ログイン処理時に，ログインした利用者 ID と，利用者 ID にひも付く得意先コード及び得意先名はセッションオブジェクトに保存されている。

注文情報照会機能には，業務処理を実行するクラス（以下，ビジネスロジッククラスという）及びリクエスト処理を実行するクラス（以下，サーブレットクラスという）が使用されている。注文情報照会機能が参照するデータベースの E-R 図を図3に，E さんが作成したビジネスロジッククラスのソースコードを図4に，サーブレットクラスのソースコードを図5に示す。

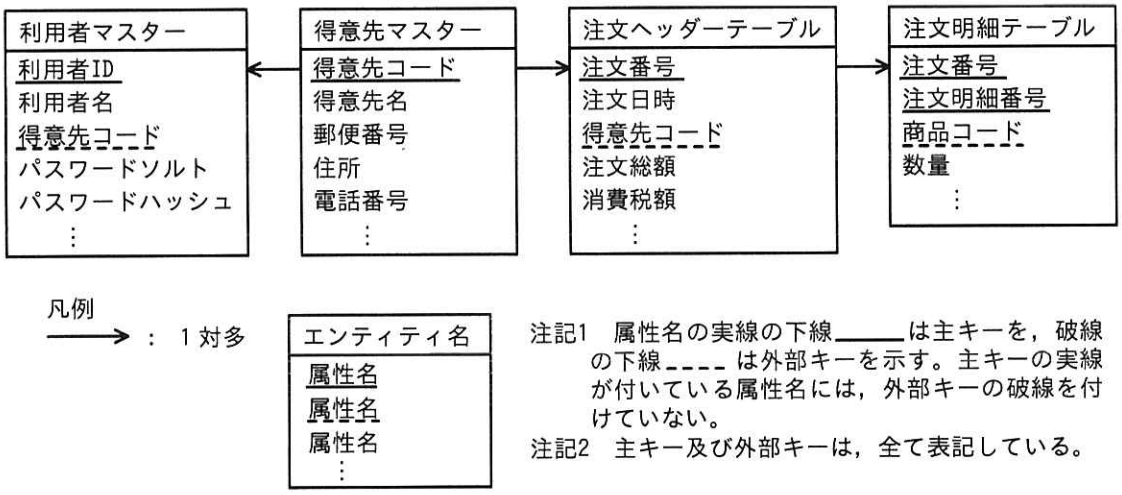

図3 注文情報照会機能が参照するデータベースの E-R 図

```
    (省略) //package宣言, import宣言など
 1: public class OrderInfoBL {
 2:   private static String orderNo;  //注文番号
      /* 注文番号の設定メソッド */
 3:   public static void setOrderNo(String inOrderNo) {
 4:     orderNo = inOrderNo;
 5:   }
      /* 注文情報の取得メソッド */
 6:   public static OrderInfoBean getOrderInfoBean() {
 7:     PreparedStatement psObj;
       (省略) //try文, 変数定義など
 8:     String sql = "SELECT ";
 9:     sql = sql + (省略);  //SQL文構築
10:     sql = sql + " WHERE head.order_no = ?";  //抽出条件：注文ヘッダーテーブルの注文番
  号と画面から入力された注文番号との完全一致
       (省略) //PreparedStatementの作成
11:     psObj.setString(1, orderNo);  //検索キーに注文番号をセット
12:     ResultSet resultObj = psObj.executeQuery();
       (省略) //例外処理やその他の処理
```

図4　ビジネスロジッククラスのソースコード

```
    (省略) //package宣言, import宣言など
 1: public class OrderInfoServlet extends HttpServlet {
     (省略) //変数定義
 2:   public void doPost(HttpServletRequest reqObj, HttpServletResponse resObj) throws
  IOException, ServletException {
 3:     String orderNo;  //注文番号
       (省略) //try文, リクエストから注文番号を取得
 4:     OrderInfoBL.setOrderNo(orderNo);
 5:     OrderInfoBean orderInfoBeanObj = OrderInfoBL.getOrderInfoBean();
       (省略) //例外処理やその他の処理
```

図5　サーブレットクラスのソースコード

D 主任，E さん，F さんは，不具合の原因が特定できず，セキュアプログラミングに詳しい技術課の H さんに協力を要請した。

H さんはアプリケーションログ及びソースコードを解析し，不具合の原因を特定した。

原因は，図4で変数［　e　］が［　f　］として宣言されていることである。この不具合は，<u>①並列動作する複数の処理が同一のリソースに同時にアクセスしたとき，想定外の処理結果が生じる</u>ものである。

原因を特定することができたので，E さんは，H さんの支援の下，次の 4 点を行った。

(1) 図4の2行目から5行目までのソースコードを削除する。

(2) 図4の6行目を，図6に示すソースコードに修正する。

```
public OrderInfoBean getOrderInfoBean( g ) {
```

図6 ビジネスロジッククラスの修正後のソースコード

(3) 図5の4行目と5行目を，図7に示すソースコードに修正する。

```
OrderInfoBL orderInfoBLObj = h OrderInfoBL();
OrderInfoBean orderInfoBeanObj = orderInfoBLObj. i ;
```

図7 サーブレットクラスの修正後のソースコード

(4) 保険的な対策として，図4の10行目の抽出条件に，セッションオブジェクトに保存された［ j ］と注文ヘッダーテーブルの［ j ］の完全一致の条件をAND条件として追加する。

ソースコードの修正後，改めてシステムテストを実施した。システムテストの結果は良好であり，システムがリリースされた。

設問1 〔ツールによるソースコードの静的解析〕について答えよ。

(1) 表1中の［ a ］に入れる適切な行番号を，図1中から選び，答えよ。

(2) 表1中の［ b ］に入れる適切な変数名を，図1中から選び，答えよ。

(3) 図2中の［ c ］，［ d ］に入れる適切な字句を答えよ。

設問2 〔システムテスト〕について答えよ。

(1) 本文中の［ e ］に入れる適切な変数名を，図5中から選び，答えよ。

(2) 本文中の［ f ］に入れる適切な字句を，英字10字以内で答えよ。

(3) 本文中の下線①の不具合は何と呼ばれるか。15字以内で答えよ。

(4) 図6中の［ g ］，図7中の［ h ］，［ i ］に入れる適切な字句を答えよ。

(5) 本文中の［ j ］に入れる適切な属性名を，図3中から選び，答えよ。