問2　Javaアプレットに関する次の記述を読んで，設問1～4に答えよ。

B社は，従業員数100名の電子部品の卸業者で，メーカから商品を仕入れ，小売業者へ販売している。B社では，2年前にリリースされたC社のソフトウェアパッケージに独自機能をもつモジュール（以下，カスタムモジュールという）を追加した在庫管理システム（以下，Xシステムという）を，3か月前に導入した。その運用と保守はC社に委託している。

〔Xシステムの概要〕

Xシステムは，ソフトウェアパッケージの本体モジュールとカスタムモジュールから構成され，それらはJavaを用いて実現されている。Xシステムの利用の操作は，普段B社従業員らがインターネットの閲覧にも用いる社内標準仕様のブラウザを用いて，クライアントPC上で行っている。

B社では，在庫管理のために，以前から携帯型のバーコード読取り端末を利用していた。読取り端末で読み取ったデータのファイルを，B社従業員がクライアントPC上でXシステム用に編集している。Xシステムでは，そのファイルを用いて在庫管理情報を更新している。

〔Xシステムのカスタムモジュール〕

カスタムモジュールのうち，ファイルが決められた書式であることとサイズが規定値以下であることの適合性チェック（以下，入力チェックという）機能，入力チェック結果を利用者に通知する機能と，適合したファイルをアップロードする機能はJavaアプレット（以下，入力チェックアプレットという）で実現されており，クライアントPC上のブラウザで実行される。また，サーバマシン上では，Xシステムとして，本体モジュールに加え，クライアントPC上の入力チェックアプレットからのファイル受信と本体モジュールへのファイル引渡しを行うモジュール（以下，ファイル受信モジュールという）があり，Javaアプレットではないアプリケーションで実現されている。したがって，Xシステム全体は，入力チェックアプレット，本体モジュールとファイル受信モジュールで構成されている。図1にXシステムの構成を示す。

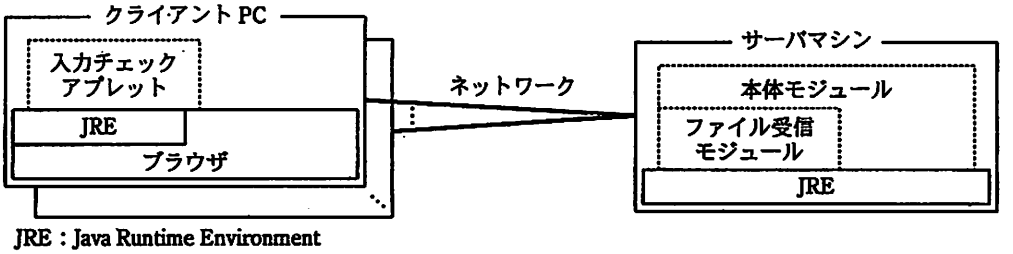

図1　Xシステムの構成

図2は，入力チェックアプレットのJavaのクラスのソースコードである。Xシステムでは，入力チェックアプレットとして構成するクラス群を［　a　］形式で一つにまとめ，［　b　］によって署名した［　a　］形式ファイルをサーバマシン上の特定のディレクトリに置き，ブラウザにダウンロードさせる。ここで，［　b　］は，有限体上の離散対数問題に基づく署名方式である。

```
public class InputCheck extends JApplet {
   private static final long MAX_FILE_SIZE = 1000000; //読み込むことを許すファイルの最大サイズ
   public void init() {
      setSize(500, 200);
      getContentPane().setLayout(null);
  (ア) JButton loader = new JButton("ファイル読込み");
      loader.setBounds(4, 3, 200, 25);
      loader.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent e) {
            JFileChooser fileChose = new JFileChooser();
            int selected = fileChose.showOpenDialog(null);
            if (selected == JFileChooser.APPROVE_OPTION) {
               File file = fileChose.getSelectedFile();
               String filePath = file.getPath(); //ファイルのパス名の取得
               if (filePath != null) {
                  loadFile(filePath);
               } else {
                (省略) //エラー処理1
               }
            }
         }
      });
      getContentPane().add(loader);
   }

   private void loadFile(String filePath) {
      byte[] fileData = null;
      int errNo = 0;
      int length, offset = 0;
      File file = new File(filePath);
      long size = file.length();
```

```
    if ( c  MAX_FILE_SIZE) {
       try {
          FileInputStream in = new FileInputStream(filePath);
       (イ) fileData = new byte[(int)size];
          //ファイルの読込み
          while ((length = (in.read(fileData, offset, (int)size - offset))) > 0) {
             offset += length;
          }
          in.close();
       } catch (FileNotFoundException e) {
          (省略) //例外処理1
       } catch (IOException e) {
          (省略) //例外処理2
       }
    (ウ) errNo = inputCheck(fileData); //入力チェック
     } else {
        (省略) //エラー処理2
     }
     (省略) //入力チェック結果の表示，適合時はアップロード，不適合時は破棄
  }

  private int inputCheck(byte[] filePath) {
     (省略)
  }
}
```

図2　入力チェックアプレット（抜粋）

入力チェックアプレットが，クライアントPC上のブラウザ上で実行されると，図2中の下線(ア)の文の処理によってボタンが作成される。利用者がそのボタンを押すと，クライアントPCのローカルファイルを選択するダイアログボックスが図3のように表示される。次に，利用者がファイルを選択すると入力チェックが行われ，適合している場合，その旨が利用者に通知され，ファイル受信モジュールを通してアップロードされ，サーバマシン上にファイルとして保存される。適合していない場合，その旨が通知され，読み込んだデータは破棄されアップロードされない。

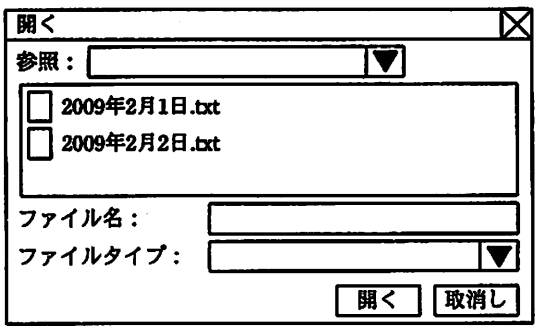

図3　JFileChooserによるファイルダイアログボックス

〔サンドボックスの概要とその脆弱性の公表〕

X システムの導入後，X システムの実行環境として想定するバージョンの JRE において，サンドボックスを回避する脆弱性（以下，当該脆弱性という）が公表され，修正された JRE の新バージョンがリリースされた。JRE のサンドボックスと当該脆弱性の概要は次のとおりである。

Java 2 以降の JRE では，サンドボックス機構における［ d ］又は SecurityManager を用いて，任意のセキュリティポリシを適用できる。JRE に用意されたデフォルトのポリシファイルを用いると，Java アプレットではない Java アプリケーションを実行する場合と，署名された Java アプレットを実行する場合，ローカルのコンピュータリソースにアクセスが可能である。他方，署名がない Java アプレットを実行する場合，そのアクセスは制限される。例えば，クライアント PC のブラウザ上で，入力チェックアプレットのようなローカルファイルへアクセスする Java アプレットを，署名なし Java アプレットとして実行した場合，［ d ］クラスによって，［ e ］がスローされる。ところが，当該脆弱性を悪用すると，署名なし Java アプレットであっても，この制限を回避して，ローカルのコンピュータリソースにアクセスできてしまう。

〔X システムでの対処〕

B 社は，当該脆弱性への対策に関して C 社と検討を行った。まず，①X システムのサーバ側とクライアント側の JRE のバージョンアップに伴う確認作業の負荷を考慮した結果，現段階では，バージョンアップに伴う作業工数の確保は難しいと判断した。しかし，②X システムのクライアント PC の JRE をバージョンアップしない場合，クライアント PC 上のブラウザで当該脆弱性が悪用される可能性がある。そこで，③Java アプレットを利用しない代替方式を検討したが，予定する期間内に実現することは難しいと判断した。よって，B 社は，④当該脆弱性への暫定処置として，サーバ側の JRE はバージョンアップせず，JRE の利用が Java アプレットに限定されているクライアント側の JRE だけバージョンアップすることとした。サーバ側の JRE のバージョンアップは，次回のシステム更新時に行うこととし，X システムを再稼働した。

設問1　本文中の［ a ］，［ b ］，［ d ］，［ e ］に入れる適切な字句を解答群の中から選び，記号で答えよ。

解答群

ア　AccessController　　イ　DH　　ウ　DSA　　エ　jar

オ　java.io.IOException　　カ　java.security.AccessControlException

キ　policytool　　ク　RSA　　ケ　xml

設問2　Xシステムの Java アプレットについて，(1)，(2)に答えよ。

(1)　図2中の［ c ］に入れる適切な字句を解答群の中から選び，記号で答えよ。

解答群

ア　`file <=`　　イ　`file >=`　　ウ　`size <=`　　エ　`size >=`

(2)　当該脆弱性のない JRE を用いた X システムにおいて，仮に署名がない入力チェックアプレットを実行する場合，図 2 中の下線(ア)～(ウ)の中で実行される文をすべて選び，記号で答えよ。ここで，実行の際，JRE に用意されたデフォルトのポリシファイルを用いていることとする。

設問3　当該脆弱性への対策について，(1)，(2)に答えよ。

(1)　クライアント PC の JRE をバージョンアップしない場合，どのようなときに，本文中の下線②で示す可能性が高まるか。B 社のクライアント PC の利用状況を考慮し，45 字以内で述べよ。

(2)　本文中の下線③で示す代替方式として，HTML フォームによってファイルを選択する機能と，アップロード後にサーバへファイルを保存する機能を追加したとする。このとき，X システムのサーバ側で，更にどのような機能を追加すれば代替方式となるか。追加すべき機能を二つ挙げ，それぞれ 20 字以内で述べよ。

設問4　JRE のバージョンアップについて，(1)，(2)に答えよ。

(1)　本文中の下線①にある，JRE のバージョンアップに伴う確認作業とは何か。X システムでの作業対象を図 1 中の用語を用いて示し，確認作業の内容を 55 字以内で述べよ。

(2)　本文中の下線④に関して，X システムのサーバ側の JRE をバージョンアップしなくても問題ないと判断するためには何を確認すべきか。60 字以内で述べよ。