Javaセキュアコーディングセミナー東京

第3回

入出力と例外時の動作

演習解説

2012年11月11日(日)
JPCERTコーディネーションセンター
脆弱性解析チーム
戸田 洋三

Hands-on Exercises

- コンパイルエラーに対処しよう
- ファイルからのデータ入力を実装しよう

Hands-on Exercise(1)

サンプルコードの
コンパイルエラーに対処しよう

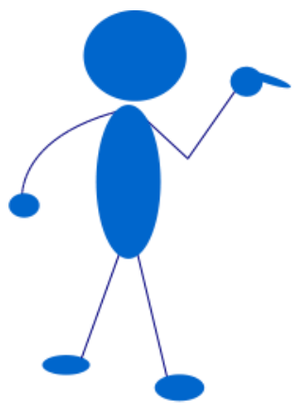
以下のコードをコンパイルできるように修正せよ.

```
class CatFile {
    public static void cat(File file) {
        RandomAccessFile input = null;
        String line = null;

        try {
            input = new RandomAccessFile(file, "r");
            while ((line = input.readLine()) != null) {
                System.out.println(line);
            }
            return;
        } finally {
                if (input != null) {
                    input.close();
                }
        }
    }
}
class testCatFile {
    public static void main(String[] args){
        if (args.length >= 1) {
          CatFile.cat(new File(args[0]));
        }
    }
}
```

コンパイルすると...

```
$ javac testCatFile_CantCompile.java
testCatFile_CantCompile.java:19: error: unreported exception IOException;
must be caught or declared to be thrown
            input.close();
                        ^
testCatFile_CantCompile.java:12: error: unreported exception
FileNotFoundException; must be caught or declared to be thrown
        input = new RandomAccessFile(file, "r");
                  ^
testCatFile_CantCompile.java:13: error: unreported exception IOException;
must be caught or declared to be thrown
        while ((line = input.readLine()) != null) {
                                         ^
3 errors
$
```



チェック例外である IOException と FileNotFoundException の扱いを記述する必要がある.

修正例その1

```
class CatFile {
    public static void cat(File file) {
        RandomAccessFile input = null;
        String line = null;

        try {
            input = new RandomAccessFile(file, "r");
            while ((line = input.readLine()) != null) {
                System.out.println(line);
            }
            return;
        } catch (FileNotfoundException e){
            System.err.println(e.toString());
        } catch (IOException e){
            System.err.println(e.toString());
        } finally {
            if (input != null) {
                try {
                input.close();
                } catch (IOException e){
                    System.err.println(e.toString());
                }
        }}}
}
class testCatFile {
    … 以下省略 ……
```

cat メソッドのなかで
例外を処理する修正例.

cat メソッドや
testCatFile クラスの
main メソッドで
throws 宣言する方法
もあり.

修正例その2(JavaSE7)

```
class CatFile {
    public static void cat(File file) {
      try (RandomAccessFile input = new RandomAccessFile(file, "r"); ) {

          String line = null;
          while ((line = input.readLine()) != null) {
              System.out.println(line);
          }
          return;
      } catch (FileNotFoundException e) {
          System.err.println(e.toString());
    } catch (IOException e) {
          System.err.println(e.toString());
    }
  }
}
```

try-with-resource 構文を使うことで, 明示的に close() を呼び出す必要がなくなる.

Hands-on Exercise(2)

(A)以下の testListOfNumbers.java について、コンパ イル実行できるようにコードを修正せよ.

(B) ListOfNumbers クラスで出力されるファイル OutFile.txt から整数を読み込み, その総和を計算するコードを書け.

(C)データを読み込んだ後の OutFile.txt を削除するように修正せよ. symlink 攻撃を防ぐにはどうすればよいか?

testListOfNumbers.java (1/3)

```
/*
 * Copyright (c) 1995, 2008, Oracle and/or its affiliates.
 * All rights reserved.
 */

import java.io.*;
import java.util.Vector;

class ListOfNumbers {
        private Vector<Integer> victor;
        private static final int SIZE = 10;

        public ListOfNumbers () {
                victor = new Vector<Integer>(SIZE);
                for (int i = 0; i < SIZE; i++)
                        victor.addElement(new Integer(i));
        }
```

testListOfNumbers.java (2/3)

```
    public void writeList() {
      PrintWriter out = null;
      try {
        System.out.println("Entering try statement");
        out = new PrintWriter(new FileWriter("OutFile.txt"));
              for (int i = 0; i < SIZE; i++)
                  out.println(victor.elementAt(i));
      } finally {
        if (out != null) {
            System.out.println("Closing PrintWriter");
            out.close();
        } else {
            System.out.println("PrintWriter not open");
        }
      }
}
```

testListOfNumbers.java (3/3)

```
class testListOfNumbers {
    public static void main(String[] args){
        ListOfNumbers lon = new ListOfNumbers();
        lon.writeList();
    }
}
```

Hands-on Exercise(2)

(A)以下の testListOfNumbers.java について、コンパ イル実行できるようにコードを修正せよ.

- FileWriter() が IOException をスローする可能性
- elementAt() が ArrayIndexOutOfBoundsException をスローする可能性

修正例: writeList()

```
public void writeList() {
 PrintWriter out = null;
 try {
    System.out.println("Entering try statement");
    out = new PrintWriter(new FileWriter("OutFile.txt"));
          for (int i = 0; i < SIZE; i++)
              out.println(victor.elementAt(i));
 } catch (ArrayIndexOutOfBoundsException e){
       System.err.println(“Caught ArrayIndexOutOfboundsException: “
                                + e.getMessage());
 } catch (IOException e){
       System.err.println(“Caught IOException: “ + e.getMessage());
    } finally {
       if (out != null) {
          System.out.println("Closing PrintWriter");
          out.close();
       } else {
          System.out.println("PrintWriter not open");
       }
    }
  }
}
```

Hands-on Exercise(2)

(A)以下の testListOfNumbers.java について、コンパ イル実行できるようにコードを修正せよ.

(B) ListOfNumbers クラスで出力されるファイル OutFile.txt から整数を読み込み, その総和を計算するコードを書け.

以下のコード例 sum.java では writeList() に対応して readList() をつくった. main() メソッドで OutFile.txt を生成するところも一緒に入れた.

コード例 sum.java (1/2)

```
/*
 * sum.java
 *   call ListNumbers to generate outfile.txt
 *   read outfile.txt and compute and print the sum of the numbers
 */

class readList {
  public Vector<Integer> readList(String filename) throws IOException {
    Vector<Integer> victor = new Vector<Integer>();
    try ( BufferedReader in =
                new BufferedReader(new FileReader(filename)); ) {
       String line;
       while ((line = in.readLine()) != null) {
          victor.addElement(Integer.parseInt(line));
       }
    }
    return victor;
  }
}
```

コード例 sum.java (2/2)

```
class sum {
    private static final String FILENAME = “OutFile.txt”;

    public static void main(String[] args) throws IOException {
        ListOfNumbers lon = new ListOfNumbers();
        lon.writeList(); // file generated

        System.out.println(“reading...”);
        readList rl = new readList();
        Vector<Integer> v = rl.readList(FILENAME);
        Integer total = 0;
        for (Integer i : v){ total = total + i; }
        System.out.println(“total: “ + total);
    }
}
```

Hands-on Exercise(2)

以下のコード例 sum1.java では
•main() メソッドの最初で, ファイル操作を行うディレクトリがセキュアディレクトリであることを確認
　✓具体的なコードについては FIO-00J を参照
•main() メソッドの最後でファイルを削除

(C)データを読み込んだ後の OutFile.txt を削除するように修正せよ. symlink 攻撃を防ぐにはどうすればよいか?

コード例 sum1.java

```
class sum1 {
  private static final String FILENAME = “OutFile.txt”;

  public static void main(String[] args) throws IOException {

    // まず最初にカレントディレクトリがセキュアディレクトリであることを確認する
    // Java セキュアコーディングスタンダード FIO00-J で紹介している
    // isInSecureDir() メソッドを参照 (POSIX系のファイルシステムの場合)
    // https://www.jpcert.or.jp/java-rules/fio00-j.html

    ListOfNumbers lon = new ListOfNumbers();
    lon.writeList(); // file generated

    System.out.println(“reading...”);
    readList rl = new readList();
    Vector<Integer> v = rl.readList(FILENAME);
    Integer total = 0;
    for (Integer i : v){ total = total + i; }
    System.out.println(“total: “ + total);

    File outfile = new File(FILENAME);
    outfile.delete();
  }
}
```