

## Analiza statică și dinamică

### - Web Server -

#### 1. Analiză statică

Pentru analiza statică a proiectului, am folosit pluginul SpotBugs din IntelliJ IDEA. După rulare, pluginul a depistat 6 bug-uri în 3 clase diferite, 4 dintre ele fiind Dodgy code, iar 2 Internationalization.

##### a) Dodgy code

The screenshot shows the IntelliJ IDEA interface with the `WebServer.java` file open. The code is a Java class that implements a simple web server. It has a `switch` statement with three cases: `"Running"`, `"Maintenance"`, and `"Stopped"`. The `"Running"` case has a `default` block. A SpotBugs warning is displayed on the right side of the IDE, indicating a "Dodgy code" issue. The warning message is: "Switch statement found where default case is missing". The warning is associated with the `WebServer` class, lines 124-154, and the `sendOutputStream` method. The priority is "Medium Confidence Dodgy code".

```
view WebServer.java:
    case "Running": {
        if (Files.exists(filePath)) {
            String contentType = Files.probeContentType(filePath);
            byte[] fileContent = Files.readAllBytes(filePath);
            out.write((version + "\r\n200 OK").getBytes());
            out.write(("Content-Type: " + contentType + "\r\n").getBytes());
            out.write("\r\n".getBytes());
            out.write(fileContent);
            out.write("\r\n\r\n".getBytes());
        } else {
            String contentType = Files.probeContentType(Paths.get(this.getHome(), "404.html"));
            byte[] fileContent = Files.readAllBytes(Paths.get(this.getHome(), "404.html"));
            out.write((version + "\r\n404 Not Found").getBytes());
            out.write(("Content-Type: " + contentType + "\r\n").getBytes());
            out.write("\r\n".getBytes());
            out.write(fileContent);
            out.write("\r\n\r\n".getBytes());
        }
    }
    case "Maintenance": {
        String contentType = Files.probeContentType(Paths.get(this.getHome(), "Maintenance.html"));
        byte[] fileContent = Files.readAllBytes(Paths.get(this.getHome(), "Maintenance.html"));
        out.write((version + "\r\n503 Service Unavailable").getBytes());
        out.write(("Content-Type: " + contentType + "\r\n").getBytes());
        out.write("\r\n".getBytes());
        out.write(fileContent);
        out.write("\r\n\r\n".getBytes());
    }
    case "Stopped": {
        System.out.println("Request made while Web Server is stopped.");
    }
    default: {
        System.out.println("The current Web Server status is not defined.");
    }
}
```

Switch statement found where default case is missing

Class: `WebServer` () lines 124-154

Method: `sendOutputStream` (`WebServer.sendOutputStream(Socket, Path, String)`)

Priority: Medium Confidence Dodgy code

Problem class: Dodgy code (through) SF\_SWITCH statement four missing) SwitchFallthro

Switch statement found where default case is missing

This method contains a switch statement where default case is missing. Usually you need to provide a default case.

Because the analysis only looks at the generated bytecode, this warning can be incorrect triggered if the default case is at the end of the switch statement and the switch statement doesn't contain break statements for other cases.

Cum această statement are default case la sfârșitul switch-ului, acesta a fost un incorrent warning și nu a avut nevoie de schimbări.

The screenshot shows the IntelliJ IDEA interface with the `WebServer.java` file open. The code is a Java class that implements a simple web server. It has a `switch` statement with three cases: `"Running"`, `"Maintenance"`, and `"Stopped"`. The `"Running"` case has a `default` block. A SpotBugs warning is displayed on the right side of the IDE, indicating a "Dodgy code" issue. The warning message is: "Switch statement found where one case falls through to the next case". The warning is associated with the `WebServer` class, lines 141-145, and the `sendOutputStream` method. The priority is "Medium Confidence Dodgy code".

```
public void sendOutputStream(Socket acceptedSocket, final Path filePath, final String version) throws Exception {
    try {
        OutputStream out = acceptedSocket.getOutputStream();
        String status = this.getStatus();
        switch (status) {
            case "Running": {
                if (Files.exists(filePath)) {
                    String contentType = Files.probeContentType(filePath);
                    byte[] fileContent = Files.readAllBytes(filePath);
                    out.write((version + "\r\n200 OK").getBytes());
                    out.write(("Content-Type: " + contentType + "\r\n").getBytes());
                    out.write("\r\n".getBytes());
                    out.write(fileContent);
                    out.write("\r\n\r\n".getBytes());
                } else {
                    String contentType = Files.probeContentType(Paths.get(this.getHome(), "404.html"));
                    byte[] fileContent = Files.readAllBytes(Paths.get(this.getHome(), "404.html"));
                    out.write((version + "\r\n404 Not Found").getBytes());
                    out.write(("Content-Type: " + contentType + "\r\n").getBytes());
                    out.write("\r\n".getBytes());
                    out.write(fileContent);
                    out.write("\r\n\r\n".getBytes());
                }
            }
            case "Maintenance": {
                String contentType = Files.probeContentType(Paths.get(this.getHome(), "Maintenance.html"));
                byte[] fileContent = Files.readAllBytes(Paths.get(this.getHome(), "Maintenance.html"));
                out.write((version + "\r\n503 Service Unavailable").getBytes());
                out.write(("Content-Type: " + contentType + "\r\n").getBytes());
                out.write("\r\n".getBytes());
                out.write(fileContent);
                out.write("\r\n\r\n".getBytes());
            }
            case "Stopped": {
                System.out.println("Request made while Web Server is stopped.");
            }
        }
    } catch (Exception e) {
        System.out.println("Request made while Web Server is stopped.");
    }
}
```

Class: `WebServer` () lines 141-145

Method: `sendOutputStream` (`WebServer.sendOutputStream(Socket, Path, String)`)

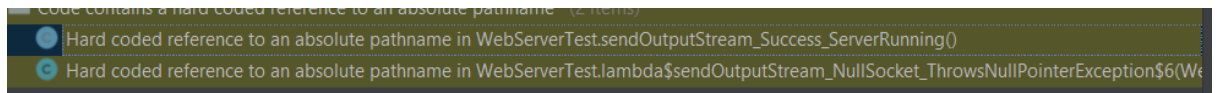
Priority: Medium Confidence Dodgy code

Problem class: Dodgy code (through) SF\_SWITCH statement four missing) SwitchFallthro

Switch statement found where one case falls through to the next case

This method contains a switch statement where one case branch will fall through to the next case. Usually you need to end this case with a break or return.

În acest caz problema nu există, căci nu e nevoie de un break sau return în cadrul case-urilor menționate.



Următoarele 2 bug-uri fac referire la valori harcodate ale unor path-uri pe care le-am folosit. Rezolvare este mutare acestora în fișiere de tip .env pentru a nu fi vizibile pe GitHub, de exemplu și a nu fi harcodate.

## b) Internationalization

Found reliance on default encoding: new java.io.InputStreamReader(InputStream)

**Class:**  
[WebServer](#) () line 103

**Method:**  
readInputStream  
(WebServer.readInputStream(Socket))

**Priority:**  
High Confidence Internationalization

**Problem classification:**  
Internationalization (Dubious method used)  
DM\_DEFAULT\_ENCODING (Reliance on default encoding)

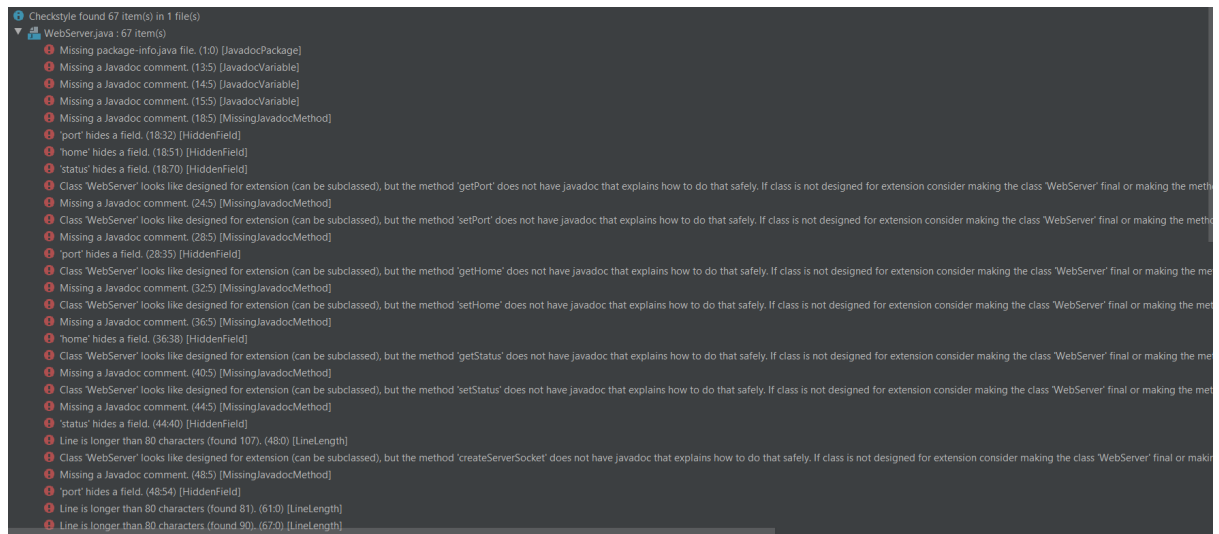
**Notes:**  
Called method new java.io.InputStreamReader(InputStream)  
DefaultEncodingDetector (Dm)

**Reliance on default encoding**  
Found a call to a method which will perform a byte to String (or String to byte) conversion, and will assume that the default platform encoding is suitable. This will cause the application behaviour to vary between platforms. Use an alternative API and specify a charset name or Charset object explicitly.

Bug-urile descoperite aici au apărut din cauza conversiei byte to String sau String to byte și nu a fost nevoie de rezolvarea lor.

De asemenea, pentru o mai clară analiză dinamică, am folosit tool-ul CheckStyle din IntelliJ IDEA.

Cu ajutorul acestui plugin am depistat 67 de posibile greșeli la proiect.

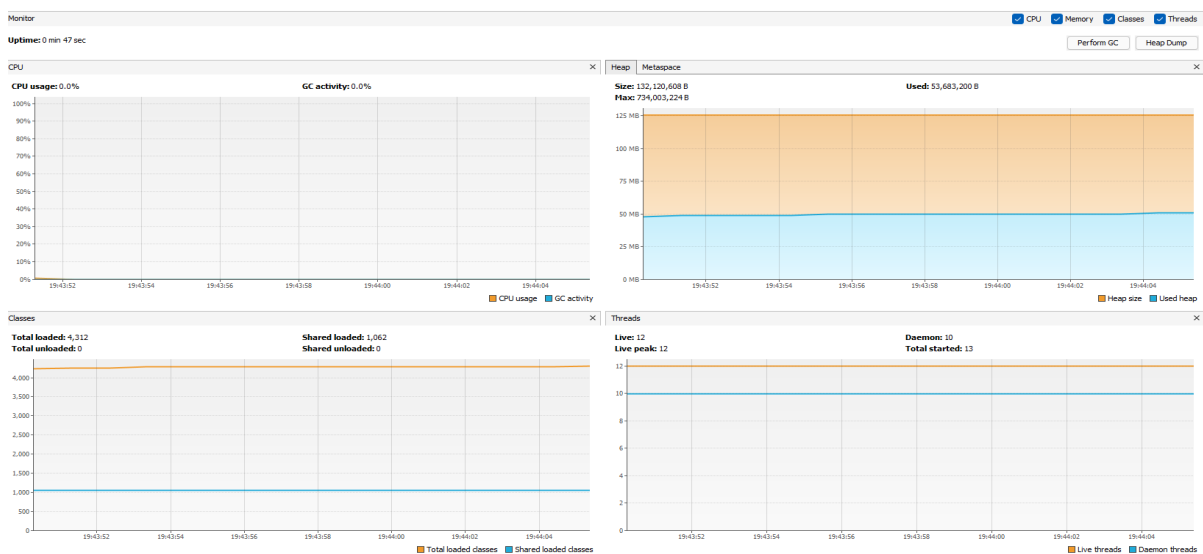


Cum majoritatea acestor bug-uri nu afectau în vreun mod modul de funcționare sau de înțelegere a proiectului, nu a fost nevoie de multe ajutări.

## 2. Analiză dinamică

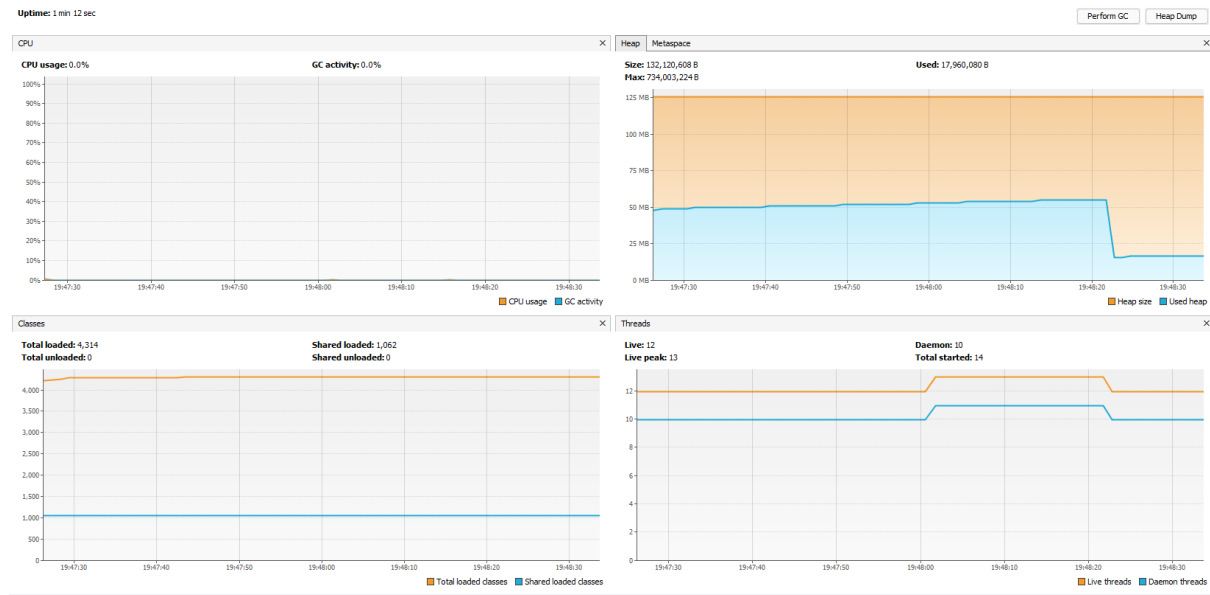
Pentru analiza dinamică am folosit tool-ul menționat la laboratorul de VVS, VisualVM, care are un plug-in pentru IntelliJ IDEA.

În modul Stopped, cu uptime de aproximativ un minut, aceasta este starea serverului:



În funcție de timp, memoria heap crește, dar are și mici scăderi la anumite intervale.

În modul Running, la aproximativ un mint uptime, serverul arată așa, având mai multe clase încărcate:



Deși memoria heap pare că scade de la un punct, după scăderea bruscă aceasta a crescut progresiv la același nivel ca anterior.

La rularea testelor, aceasta este activitatea monitorizată:

