Comp Virtual File System (CVFS)

Group Pre

COMP2021 Object-Oriented Programming (2020 Fall)

MAN Furui, WANG Meng, XING Shiji, ZHANG Yubo

Group Information

- MAN Furui 19081789d
- WANG Meng 19078543d 19079008d
- XING Shiji 19079784d
- ZHANG Yubo

Contents

1 Introduction

This document describes the design and implementation of the Comp Virtual File System (CVFS) by group XYZ. The project is part of the course COMP2021 Object-Oriented Programming at PolyU.

2 The Comp Virtual File System (CVFS)

In this section, we describe first the **overall design** of the CVFS and then the **implementation** details of the requirements. Class diagrams will be presented during the design analysis, to reflect the gist of the system's function details; Some code snippets are inserted while indicating the implementation of all the requirements of the system, and they are well-documented with inline comments, but still with simplicity and understandability.

External Link to Github Repository: https://github.com/tools-max-0/COMP2021_Group_Project

2.1 Design

This section gives an overview of the system design using class diagrams, with explanatory texts to describe in general terms how different components fit together.

⚒️ 2.1 Design Content

2.2 Requirements

In this section, the details for each (compulsory and bonus) requirement are examined. Each part contains: 1) whether the requirement is implemented and, when yes, 2) how the requirement is implemented as well as 3) how various error conditions are handled.

> 💡 Our CVFS has satisfied **ALL** the requirements, include all the **BONUS** ones.

Common Error Conditions

In the system, some errors are common and applicable to most commands.

The `Wrong Number of Parameters` error is universal, and therefore this situation will be mentioned here as a summary and not repeated in later sections. `Controller` will detect such errors using `if` statement: `if (command.length!=2) throw new IllegalArgumentException(...)`.

Some commands are dependent on the `newDisk` command. They are handled by the `if (cvfs.getCwd()==null) throw new IllegalStateException(...);` to point out the `Disk not initialized` Error.

Error Handling

Error handling is also universal to all types of commands. When each error occurs, each piece of error checking code will throw the corresponding exception, and the `CVFSController` will catch the exception using a `try-catch` block and print its error information if any.

```
package hk.edu.polyu.comp.comp2021.cvfs.control;
//file CVFSController
public void terminal() {
    // ...codes for prompt and processing input...

    // error handling, using a try-catch block
    // once detecting an error, terminal will print the error message in red.
    try {
        processCommand(type, command);
    } catch (Exception e) {
        System.out.println("\033[31m" + "Error: " + e.getLocalizedMessage() + "\033[0m");
    }
}
```

REQs 1-14 && BONs 1-2

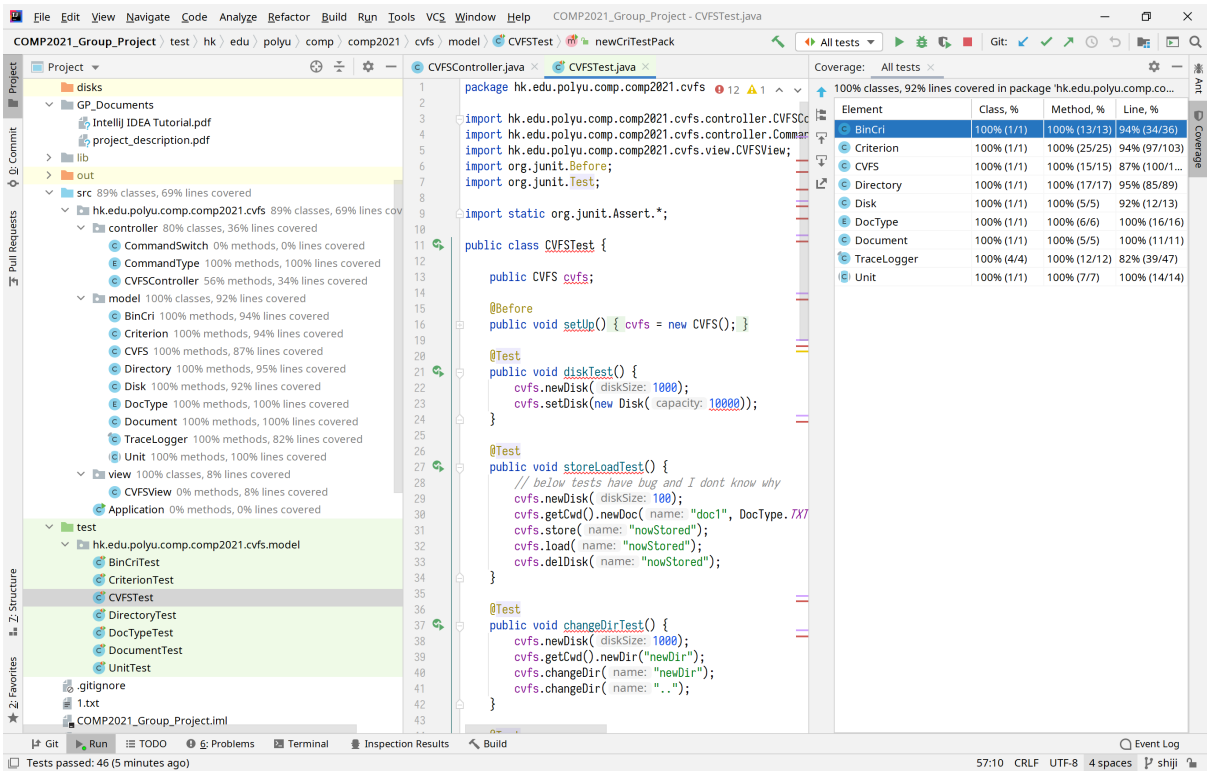
📏 REQs 1-14 + BONs 1-2 [DONE]

2.3 Test Coverage

Follow the requirement , we put all java classes for the model into `hk.edu.polyu.comp.comp2021.cvfs.model` package. The test achieves a ***92%*** line

coverage and an ***100%*** method/class coverage.

For the classes in the `hk.edu.polyu.comp.comp2021.cvfs.controller` and `hk.edu.polyu.comp.comp2021.cvfs.view`, it's hard to just use junit to test the "interface cases", so the test for such classes will be better presented detailed in the user manual.



,Figure 2.3.1: Test Coverage in `cvfs.model` package

3 User Manual

In this section, we explain how the CVFS works from a user's perspective. All the commands that the system supports are described in this section. For each command, demostration screenshots are presented to show the result under different inputs.

Please follow the link 📖 User Manual or find **User Manual.pdf** in unzipped package to see the complete User Manual.

📖 User Manual

4 Revision & Acknowledgment

In this report, we have contained the whole implementation of CVFS. We use the test cases (in the attached codes) and user manual to double check our codes. Overall, our team feels the project has been a success. A great deal of knowledge about OOP and JAVA has been gained through the trials of this project, these lessons will be invaluable in future endeavors.

We would like to express our special hanks of gratitude to Dr. PEI Yu and tutors who gave us the golden guidance and ideas during this project.

Secondly we would also like to thank the reviewers who give suggestions for modification.

Last but not the least, we would like to thanks each group member's participation during this project.

If you have any questions or concerns please feel free to contact us at anytime.