Project Description: Personal Virtual File System

Department of Computing

January 18, 2017

1 Overview

The goal of this project is to develop a single-file Virtual File System (VFS) in Java. A VFS is usually built on top of a host file system to enable uniform access to files located in different host file systems. A single-file VFS could also be used as a manageable container with the functionality of a concrete file system, and typical applications of such single-file VFSs include implementations of virtual machines, emulators, and even programming IDEs (e.g., IntelliJ IDEA[1]). You can find more information about VFSs on Wikipedia[2].

2 Important Time

- Team formation: before 23:59:59, Wednesday, Feb. 8, 2017
- VFS and tests: due at 23:59:59, Monday, Apr. 10, 2017
- Project report: due at 23:59:59, Monday, Apr. 10, 2017
- Design review session: takes place between 8:30 and 11:30, Tuesday, Apr. 11, 2017
- Design review report: due at 23:59:59, Monday, Apr. 17, 2017
- Presentation: takes place between 8:30 and 11:30, Tuesday, Apr. 18, 2017

3 Project Grading

You can earn in total at most 100 points in the project. The points are composed as follows:

- Requirements coverage: 10 points for requirement REQ_{12}; 1.5 points for each other requirement. In total, 34 points for 17 requirements (Section 5.1 and 5.2).
- Code quality: 12 points for Object-Oriented Design and 12 points for code inspection results (Section 3.1), in total 24 points.
- Statement coverage of unit tests (Section 3.2): 12 points.
- Design review report (Section 4.4): 10 points.
- Project report (Section 4.3): 10 points.
- Presentation (Section 4.5): 10 points.
- Bonus points: 18 points in total for three bonus features (Section 5.3).
  Note: Bonus points can only be used to *reach*, but not to *exceed*, 100 points.

[1] http://www.jetbrains.org/intellij/sdk/docs/basics/virtual_file_system.html
[2] https://en.wikipedia.org/wiki/Virtual_file_system

3.1 Code Inspection

The inspection tool[3] of IntelliJ IDEA IDE (Section 4.2.1) checks your program to identify potential problems in the source code. `COMP3222_PROJECT.xml` from the project material bundle defines the set of rules to be used in grading. We encourage you to import this file into your IntelliJ IDEA IDE and check your code against the rules during your development. See `quick_help.pdf` from the project material bundle for instructions on how to import inspection rules.

The inspection tool is able to detect many potential problems, but only a few of them are considered errors by the provided rule set. 2 Points will be deducted for each error in your code reported by the tool.

3.2 Statement Coverage of Unit Tests

Statement coverage[4] is a measure used in software testing to describe the percentage of statements that has been tested. Roughly speaking, the higher the coverage, the more guarantee the tests offer.

In this project, you need to use JUnit[5] to write unit tests for the code in package `hk.edu.polyu.comp3222.vfs.core`, i.e., the code implementing all the functionalities of the VFS core (Section 4.2). JUnit libraries are already included in the distribution of IntelliJ IDEA IDE, you just need to configure the libraries to use JUnit[6]. We will use the built-in tool[7] of the IntelliJ IDEA IDE to collect statement coverage information of your tests. See `quick_help.pdf` from the project material bundle for a short introduction to how to create JUnit tests.

Your will get 12 base points for statement coverage between 95% and 100%, and 11 base points for coverage between 90% and 94%, and so on. You will get 0 base points for statement coverage below 40%. Your final grade will be calculated as your base points multiplied by the percentage of your coverage of requirements for the VFS core (Section 5.1). For example, if you only implement 50% of requirements REQ_1 through REQ_{17} and you achieved 95% statement coverage of package `hk.edu.polyu.comp3222.vfs.core`, then your points for this part will be $6 = 12 * 50\%$.

4 Organization

All project work must be done in teams of 3 to 4 people. Each team needs to hand-in the source code and tests of the VFS, submit a project report, participate in a design review, and make a presentation during the project.

4.1 Team

If you intend to form a team by yourselves, do this under Blackboard/Groups before or on Feb. 8, 2017. On Feb. 9, students not belonging to any team and teams with fewer than 3 members will be randomly grouped to form teams of at least 3. All the teams will be announced on Feb. 10.

4.2 Software

4.2.1 Development Environment

The project should be developed in pure Java. IntelliJ IDEA Community Version 2016.3.2 and Java SE 1.8 will be used in grading your project, so it is important that you make sure to use the same version of IDE and JDK in your development.

Your code for implementing the regular features should not rely on any database management software or any library that is not part of the standard API of Java SE 1.8.

[3] https://www.jetbrains.com/help/idea/2016.2/code-inspection.html
[4] http://en.wikipedia.org/wiki/Code_coverage
[5] https://www.junit.org/
[6] https://www.jetbrains.com/help/idea/2016.2/configuring-testing-libraries.html
[7] https://www.jetbrains.com/help/idea/2016.2/code-coverage.html

4.2.2 Repository

Create a GIT repository[8] (named after your team ID) on Bitbucket[9] to host your Java project. Your source code and tests will be pulled directly from the repository and graded after the deadline, so you need to make sure 1) the repository is accessible to the teaching team of the course (Bitbucket account: `se_polyu_hk`) and 2) the code in your repository is ready to be opened, compiled, and executed using IntelliJ IDEA without further adjustment. 50% of the points will be deducted from your project, if the code fails to compile.

4.2.3 Project Structure

VFS-template.zip from the project material bundle contains a template IntelliJ IDEA project. You are strongly encouraged to use the template as a starting point for your development. Note

- Source code of the VFS should be placed in folder `src`, while tests in folder `test`.
- Source code of the VFS should be split into three sub-packages, i.e., `core`, `client`, and `server` of package `hk.edu.polyu.comp3222.vfs`, for functionalities of the core, the client, and the server, respectively.
- Test code should be placed inside package `hk.edu.polyu.comp3222.vfs.test`.

4.3 Report

Towards the end of the project, each team needs to submit a report on the design and implementation of the VFS. A template for the report is available at the end of this description.

The report should be in PDF format and named in the form "COMP3222-2017S-TeamID.pdf". Only submissions through the PolyU Blackboard Learn System are accepted. Do *not* submit your report via email.

Late submissions of the project report will lead to deducted points. To be more specific, 20% of the points that your project report deserves will be deducted for delay of every other day. For example, suppose your project report deserves 8 points in and of itself, then delay by 1 day (rounded up) will cost you $8 * 0.2 = 1.6$ points, delay by 2 days will cost $8 * 0.4 = 3.2$ points, and so on.

Submissions with 50% or higher similarity as reported by the Blackboard Learn System will be treated as plagiarism cases. All members of the teams involved in plagiarism will receive 0 points for the WHOLE project, and will be reported to the Departmental Learning and Teaching Committee for further disciplinary actions.

4.4 Design Review

We will have a design review session from 8:30 to 11:30 on Apr. 11, 2017. Each team has to bring hard copies of its project report to the session to be reviewed by other teams. The review will be conducted in groups each team needs to review the designs of the other teams in its group and fill out the review reports. In a review report, a team needs to point out which design choices from another team are good, which are questionable, which should have been avoided, and explain the reasons. Details about the organization of the design review and the format of the review report will be announced before the review.

A team will get zero points for the design review report part if 1) none of its members participates in the design review, 2) the team fails to bring copies of their project report to the review session, or 3) the team fails to submit the review report before the deadline.

[8] https://git-scm.com/
[9] https://bitbucket.org/

4.5 Presentation

Presentation of the project will take place from 8:30 to 11:30 on Apr. 18, 2016. Each team has 5 minutes in total for both the presentation (4 min.) and the Q&A (1 min.). The presentation should focus on one or two important decisions in your design and the corresponding justifications.

A team will get zero points for the project presentation if none of its members shows up to deliver the presentation.

5 Project Details

This section elaborates the detailed requirements for a VFS core and a VFS built around that core. Note that the VFS client should handle incorrect user inputs gracefully. For example, when a user tries to import files from a nonexistent directory, the VFS should not crash.

> The requirements have been left incomplete on purpose. When designing the VFS, you need to decide on the missing details to make the interaction with the VFS as straightforward as possible.

5.1 Part I. VFS Core

The goal of the first part is to develop the VFS core. The VFS core operates on virtual disks, each being a single file in the host file system. The core provides an API to facilitate creating, modifying, and deleting virtual disks. Particularly, the core provides means for its client to: 1) create a virtual disk of user-specified size, as well as delete an existing virtual disk; 2) import files and directories from the host file system to a virtual disk, as well as export files and directories from the VFS to the host file system; 3) navigate through the directories of a virtual disk, as well as rename, copy, remove or move existing files and directories in the virtual disk; and 4) search for files and folders based on user-given keywords. The size of a virtual disk is fixed during its lifetime, and errors should be reported when pending operations are infeasible due to this restriction.

Requirements for the VFS core include the following:

REQ_1 A virtual disk must be stored in a single file in the working directory in the host file system.

REQ_2 The VFS core must support the creation of a new disk with the specified maximum size at the specified location in the host file system.

REQ_3 The VFS core must support several virtual disks in the host file system.

REQ_4 The VFS core must support disposing of a virtual disk.

REQ_5 The VFS core must support creating/deleting/renaming directories and files.

REQ_6 The VFS core must support navigation: listing of files and folders, and going to a location expressed by a concrete path.

REQ_7 The VFS core must support moving/copying directories and files, including hierarchy.

REQ_8 The VFS core must support importing files and directories from the host file system.

REQ_9 The VFS core must support exporting files and directories to the host file system.

REQ_{10} The VFS core must support querying of free/occupied space in the virtual disk.

REQ_{11} The VFS core must support file/folder search based on user-given keywords. The search should support the following options:

- case sensitive or insensitive search;
- matching all keywords or matching any keyword;
- starting search from a specified folder;
- matching only files/subfolders or matching both files and subfolders;

5.2 Part II: VFS Client and Server

The goal of this part is to develop 1) a VFS client with a command-line interface that runs on a local machine and 2) a synchronization server that propagates changes in a virtual disk from one machine to others using a network connection. You should build the client based on the VFS core from the previous part.

We assume in this project that no virtual disk is linked to multiple user accounts, and that changes to a virtual disk always take place on a single machine at any point in time. The synchronization of VFSs is based on accounts, i.e. a user must log in to an account on the server for the changes s/he makes to be synchronized across machines.

Requirements for the VFS client and server include the following:

REQ_{12} The client should support all the operations from part I (VFS Core).

REQ_{13} The client should, when connected with the server, allow the user to create a new account or to log in to an existing account.

REQ_{14} The client should offer to switch to an offline mode, and be able to operate without a connection to the server.

REQ_{15} The client should support linking an existing virtual disk to an active account.

REQ_{16} The server should store all the created unique accounts (see REQ_{13}). For each account, the account name and the password should be saved.

REQ_{17} The server should track changes to linked virtual disks of existing accounts and synchronize, upon user request, the changes across machines.

5.3 Bonus Features

BFT_1 Support for file history and restoring a previous version (3 points).

BFT_2 Automatic and concurrent synchronization between the client and the server (6 points).

BFT_3 A VFS browser with GUI that supports all the operations of a VFS client. The browser should support mouse navigation and drag-and-drop (9 points).

Appendix: Project Report Template

Project Report

Group XYZ
COMP3222 Software Design Principles, 2017

Name1
Name2
Date

1. Introduction

This document describes the design and implementation of a single-file Virtual File System (VFS) from group XYZ. The project is part of the course COMP3222 Software Design Principles *at PolyU. The following sections describe the requirements that were implemented and the design decisions made. The last section describes the available commands in the VFS.*

2. The Single-File VFS

Give a short description of what the VFS is.

2.1 Requirements

Describe in this part which requirements (and bonus requirements if applicable) you have implemented. Give a short description (1-2 sentences) of each requirement. List the software elements (i.e., classes and/or functions) that are related to each requirement.

Req-x: Description

Software elements

2.2 Design

Give an overview of the design of the VFS and describe in general terms how the implementation works. You can include here design patterns used, class diagrams, and other things that are important for understanding the design.

Select two important aspects of your design, discuss alternative choices, and justify your decisions.

3. Quick Start Guide

Describe here a typical use scenario of the VFS.