BUG REPORT

CONTENTS

Test file not able to delete

Faulty code:

```
@BeforeClass
    public static void setUpBeforeClass() throws Exception {
        defaultWorkingDirectory = new File(System.getProperty(USER_DIRECTORY));

        // creating testFile 1 & 2 in sorted order
        File myFile1 = new File("testFile1.txt");
        myFile1.createNewFile();
        writeFile("testFile1.txt", "aaa\r\nbbb\r\nccc\r\nddd");
    }

    @AfterClass
    public static void tearDownAfterClass() throws Exception {
        // creating testFile 1 & 2 in sorted order
        File myFile1 = new File("testFile1.txt");
        Files.delete(testFile.toPath());
    }
```

Expected Behaviour: Test file will be deleted after the tearDownAfterClass() is executed.
Real Behaviour: The test file will be deleted irregularly.
Relevant Details:
The bug does not surface on a regular basis. This does not work due to unknown reason. Our speculation is that it involved file permission. Hence we use another approach to creation of the file.

Pipe Integration Testing:

Error encountered in Cut but status code still 0

Test Case that Detects the Fault:

```
@Test
public void integrateWithPipe_CutError_GetStatusCode1() {
            // cut file does not exist
       String commandline = "cat emptyFile | cut -c 15-20 file | paste -s ";

       Vector<String> results = Shell.shellTestExecution(commandline);
       assertEquals(1, Integer.parseInt(results.elementAt(1)));
}
```

Description of Bug:

Status code returned in the assertion is returning an invalid status code. This is caused by the absence of setting of the statusCode in the cut command's implementation code as seen below:

```
while (isValid) {
       if (files == null || files.isEmpty()) {
              fileContent = stdin;
              isValid = false;
              returnMessage = cutSpecfiedCharacters(list, fileContent);
       } else {
              fileToRead = new File(files.get(fileCount));

              if(fileToRead.exists()) {
                     fileContent = new String(Files.readAllBytes(fileToRead.toPath()));
                     returnMessage = cutSpecfiedCharacters(list, fileContent);
              } else {
                     returnMessage = "cut: " + fileToRead.getName()
                                   + ": No such file or directory";
              }
       } ……
```

Debugged Code:

```
while (isValid) {
       if (files == null || files.isEmpty()) {
              fileContent = stdin;
              isValid = false;
              returnMessage = cutSpecfiedCharacters(list, fileContent);
       } else {
              fileToRead = new File(files.get(fileCount));

              if(fileToRead.exists()) {
                     fileContent = new String(Files.readAllBytes(fileToRead.toPath()));
```

```
                    returnMessage = cutSpecfiedCharacters(list, fileContent);
            } else {
                    setStatusCode(FILE_NOT_EXISTS);
                    returnMessage = "cut: " + fileToRead.getName()
                                                + ": No such file or directory";
            }
      }.......
```

Pipe Integration Testing:
Uniq standard input cannot be executed

Test Case that Detects the Fault:

```
@Test
public void integrateWithPipe_GrepFilesUniqWcStdin_GetStatusCode0() {
        // grep files
        // uniq, wc with standard input

        String commandline = "grep -A 5 test file1 file2 | uniq - | wc -l ";

        Vector<String> results = Shell.shellTestExecution(commandline);
        assertEquals(0, Integer.parseInt(results.elementAt(1)));
}
```

Description of Bug:

Uniq command did not allow standard input (-) hence the code below was not implemented previously until this bug was discovered.

```
In UniqTool:

// Check for stdin argument
if (arg.equals("-")) {
        operands.add(arg);
        continue;
}
```

Pipe Integration Testing:
Error encountered in Uniq but status code still 0

Test Case that Detects the Fault:

```
@Test
public void integrateWithPipe_UniqError_GetStatusCode1() {
	// uniq file cannot be read
	file2.setReadable(false);
	String commandline = "cat file1 | uniq -f 12 file2 | grep \"a\" - | wc -l -";

	Vector<String> results = Shell.shellTestExecution(commandline);
	assertEquals(1, Integer.parseInt(results.elementAt(1)));
	file2.setReadable(true);
}
```

Description of Bug:
Status code returned in the assertion is returning status code 0 when there is an error where the file is unreadable.

```
public String readFile(String path, Charset encoding) {
	byte[] encoded;
	try {
		encoded = Files.readAllBytes(Paths.get(path));
	} catch (IOException e) {
		return e.getMessage();
	}
	return encoding.decode(ByteBuffer.wrap(encoded)).toString();
}
```

Pipe Integration Test: Cut does not allow -d option's argument without quotes but Shell removes the quotes while parsing

Test Case that Detects the Fault:

```
@Test
public void integrateWithPipe_SortFileUniqCutStdin_GetStatusCode0() {
	// sort file
	// uniq, cut standard input

	String commandline = "sort file1 file2 | uniq -f 1 | cut -d \"*\" -f 9,2-5,4";

	Vector<String> results = Shell.shellTestExecution(commandline);
	assertEquals(0, Integer.parseInt(results.elementAt(1)));
}
```

Description of Bug:

When the Shell parses the Cut command and encounters -d option with quoted arguments, it will remove all the quotes of arguments in cleanUpArguments(..) method. However, CutTool only allows -d option argument with quotes thus it returns an error upon checking that it is missing the quotes.

```
In Shell's getCutArgs(..) method:

delim = getPatternOrDelimiter(argsStr);
if (delim == null) {
	if (extraType.equals("DELIM"))
		return null;
}
else {
	argsVect.add(delim);
	argsStr = reduceString(delim, argsStr);
}
…

return cleanUpArguments(argsVect);
```

Integration Testing:

cd tool does not not accept null object as argument

Faulty code

```
public String execute(final File workingDir, final String stdin) {
    setStatusCode(0);
    String output = "";

    if(workingDir == null | !workingDir.exists() | !workingDir.isDirectory()){
        setStatusCode(DIRECTORY_ERROR_CODE);
        output = DIRECTORY_ERROR_MSG;
    }
    else if(this.args != null){
        final String newDirectory = this.args[0];
        if(    !newDirectory.equals(workingDir) &&  changeDirectory(newDirectory) == null){
               setStatusCode(DIRECTORY_ERROR_CODE);
               output = DIRECTORY_ERROR_MSG;
        }
    }
    return output;
}
```

Expected Behaviour: null argument should be reject and throw error message
Real Behaviour: Able to take in null and process it as empty string execution

Debugged code

```
  public String execute(final File workingDir, final String stdin) {
      setStatusCode(0);
      String output = "";

      if (workingDir == null || !workingDir.exists() || !workingDir.isDirectory() ||
this.args == null) {
        setStatusCode(DIRECTORY_ERROR_CODE);
        output = DIRECTORY_ERROR_MSG;
      }
      else{
        final String newDirectory;

        if("-".equals(this.args[0]) && previousDirectory != null){
               newDirectory = previousDirectory;
        }
        else{
               newDirectory = this.args[0];
        }

        if(    !newDirectory.equals(workingDir) && changeDirectory(newDirectory) == null){
               setStatusCode(DIRECTORY_ERROR_CODE);
               output = "cd: " + args[0] + ": " + FILE_NOT_FOUND;
        }
        else{
```

```
                previousDirectory = workingDir.getAbsolutePath();
                output = changeDirectory(newDirectory).getAbsolutePath();
            }
        }
        return output;
    }
```

State Change Integration Test:

Shell does not allow a valid directory name

Test Case that Detects the Fault:

```
@Test
public void cdCopyLsWithPiping_FilesListed() {

	final String complexCmdStr = "cd " + "\"" + tempDir.toString() + "\""
			+ "|copy \"" + input.toString() + "\" \"" + tempDir.toString()
			+ File.separator + "inputNew\""
			+ "|ls";

	Vector<String> finalResult = Shell.shellTestExecution(complexCmdStr);

	assertTrue(finalResult.get(0).contains("inputNew"));

}
```

Description of Bug:

The regular expression used to validate a directory name was too restrictive as it did not allow "-", "&" and had problems with the Windows OS format of an absolute path.

```
public class Shell implements IShell {

	final private static String DIR_REGEX_NO_SPACE  = "(((~|.|[a-zA-Z]:)[\\/])?([\\/]*[a-
zA-Z][a-zA-Z](\\w)*)+[\\/]*)";
	final private static String DIR_REGEX_SPACE     = "(((~|.|[a-zA-Z]:)[\\/])?([\\/]*[a-
zA-Z][a-zA-Z](\\w|\\s)*)+[\\/]*)";
	final private static String DIR_REGEX_Q_SPACE   = "\""+DIR_REGEX_SPACE+"\"";

...
```

Different help message from TDD test case

Faculty code:

```
@Test
public void getHelp() throws IOException {

    String result = ("-c : check that input is correctly sorted, even if all input lines are
pairable"
        + System.lineSeparator()+ "-d : do not check that the input is correctly sorted"
        + System.lineSeparator() + "");
    (result, commTool.getHelp());
}
```

Expected Behaviour: The help text should be the whole description of the help text.
Real Behaviour: The help text only contain the help text for option.

Debugged code:

```
private final String COMM_HELP_MESSAGE = "comm : Compares two sorted files "
    + "line by line. With no options, produce three-column "
    + "output. Column one contains lines unique to FILE1, "
    + "column two contains lines unique to FILE2, and column "
    + "three contains lines common to both files."
    + System.lineSeparator()
    + "Command Format - comm [OPTIONS] FILE1 FILE2"
    + System.lineSeparator()
    + "FILE1 - Name of the file 1"
    + System.lineSeparator()
    + "FILE2 - Name of the file 2"
    + System.lineSeparator()
    + "-c : check that the input is correctly sorted, even if "
    + "all input lines are pairable"
    + System.lineSeparator()
    + "-d : do not check that the input is correctly sorted"
    + System.lineSeparator()
    + "-help : Brief information about supported options";

@Corrected
// Swap result and expected, change the output of the help message
@Test
public void getHelp() throws IOException {
   assertEquals(COMM_HELP_MESSAGE, commTool.getHelp());
}
```

This is seen in both Comm and Sort, and it is fixed now.

Cd Tool

Failing Test case:

```
@Test
public void execute_ValidNavigateBack_StatusCodeZero(){

    String[] argument = {"src"};
    cdTool = new CdTool(argument);
    cdTool.execute(workingDirectory, null);

    argument[0] = "-";
    cdTool = new CdTool(argument);
    String result = cdTool.execute(workingDirectory, null);

    assertEquals(workingDirectory.toString(), result);
}
```

Faculty code:

```
public String execute(final File workingDir, final String stdin) {

    setStatusCode(0);
    String output = "";
    if (workingDir == null || !workingDir.exists() || !workingDir.isDirectory() || this.args
== null) {
         setStatusCode(DIRECTORY_ERROR_CODE);
         System.out.println(workingDir);
         output = DIRECTORY_ERROR_MSG;
    }
    else{
         final String newDirectory;
         if("-".equals(this.args[0]) && previousDirectory != null){
                newDirectory = previousDirectory;
         }
    }
}
```

Debugged code:

```
@Override
public String execute(final File workingDir, final String stdin) {
    setStatusCode(0);
    String output = "";

    if (workingDir == null || !workingDir.exists() || !workingDir.isDirectory() || this.args
== null) {
         setStatusCode(DIRECTORY_ERROR_CODE);
         System.out.println(workingDir);
         output = DIRECTORY_ERROR_MSG;
    }
    else{
         final String newDirectory;
```

```
        if("-".equals(this.args[0]) && previousDirectory != null){
                newDirectory = previousDirectory;
        }
        else{
                newDirectory = this.args[0];
        }

        if(     !newDirectory.equals(workingDir) && changeDirectory(newDirectory) == null){
                setStatusCode(DIRECTORY_ERROR_CODE);
                output = "cd: " + args[0] + ": " + FILE_NOT_FOUND;
        }
        else{
                previousDirectory = workingDir.getAbsolutePath();
                output = changeDirectory(newDirectory).getAbsolutePath();
        }
    }
    return output;
}
```