Assignment: Deliverable1

Name: Umesh Dhakal

Class: MSCS630

Date: 09/06/2025

Dr. Primus Vekuh

GitHub for Source Code and all the other files-

https://github.com/udhakal1s/ShellImplementation

Process Management and creation

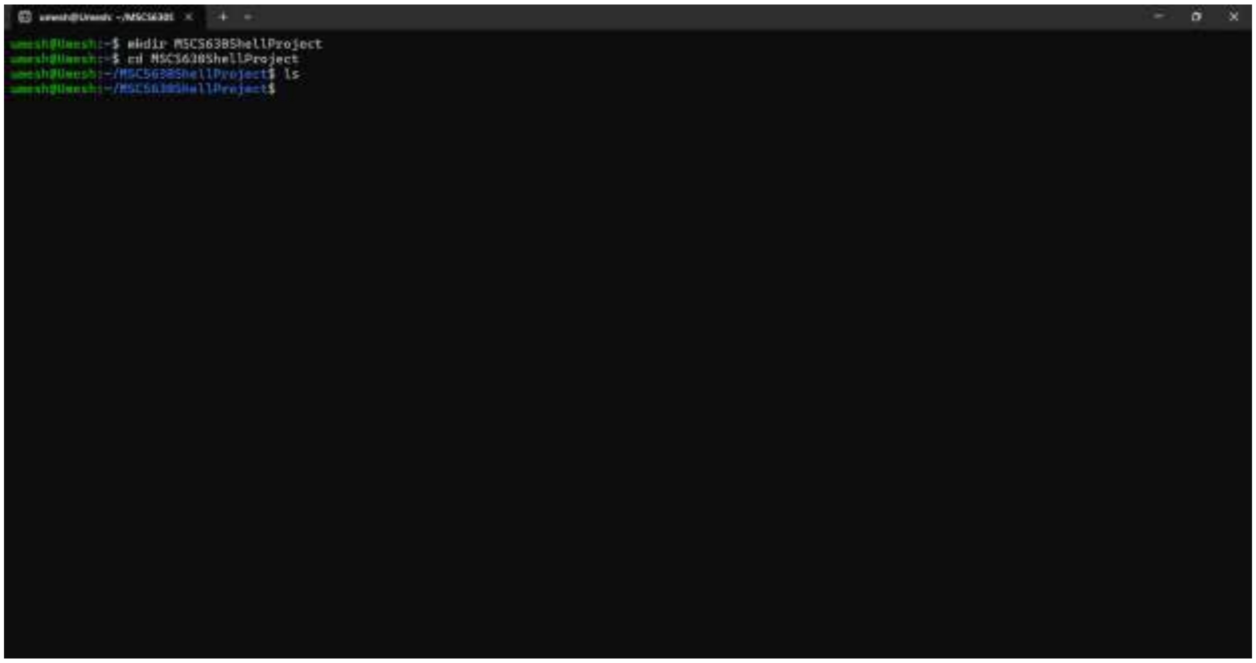
In this shell implementation we created the java file called MSCS630shellimplementation that accepts and executes user commands and implement the built in commands and Support foreground and background execution of commands. In the program, all the commands are executed using the ProcessBuilder and Process class. If the command is one of the built-in commands inside the Java program, the Java shell runs directly inside the program. If it is not one of the built-in commands, then the shell uses the Java ProcessBuilder to create and start a new process. In our implementation process, we run in the foreground. For example, if I type mkdir, the shell will create the directory and return to the shell prompt, where the user can type more commands. Our implementation also supports background processes. If we add & at the end of the command, it will not wait for the process to finish. This helps us to type other commands while some of the commands are still executing. If we type jobs, it will give us the list of jobs that are done or are still running. All the implementation is shown in the screenshot below.

Error Handling

The shell gives us a clear message if there is a command issue. For example, if we are not in any directory, and we type cd, it will give us a message saying missing command. Another good example of our implementation of error handling is removing a directory and removing a file. When we try to delete a file or folder that is not currently present in our current directory, it will give us a message saying no such file or no such folder to delete. The screenshot at the end of the document shows how errors are handled by our programs.

Making a new directory for Shell Implementation project

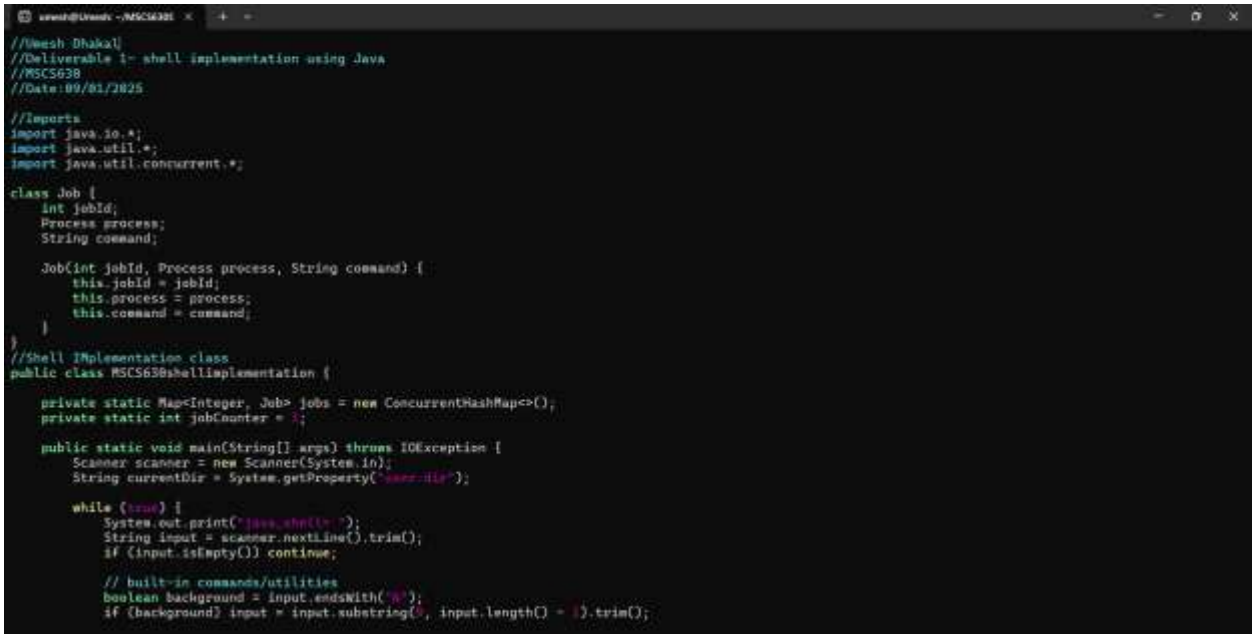
In the beginning we created the directory MSCS630ShellProject to store all our file.



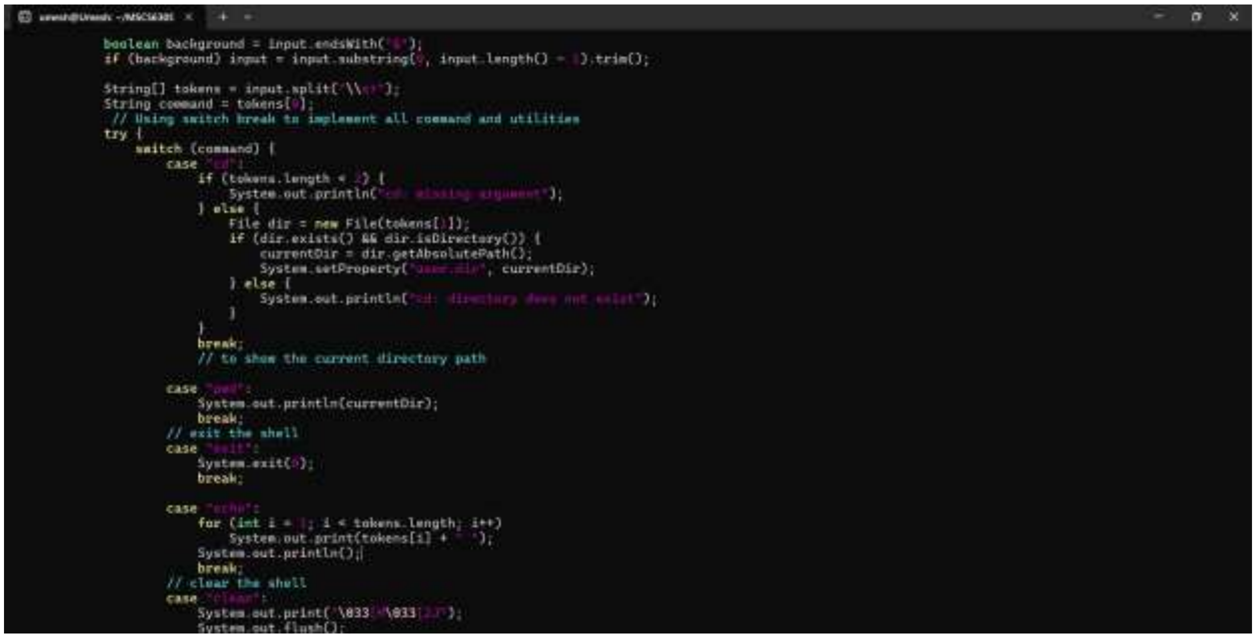
```
umesh@Umesh:~$ mkdir MSCS630ShellProject
umesh@Umesh:~$ cd MSCS630ShellProject
umesh@Umesh:~/MSCS630ShellProject$ ls
umesh@Umesh:~/MSCS630ShellProject$
```

After that we created the java file called MSCS630shellimplementation, the source code is provided below and in the GitHub repository on the beginning of the page. On the java file we implemented all the required built-in command and foreground and background process management commands.

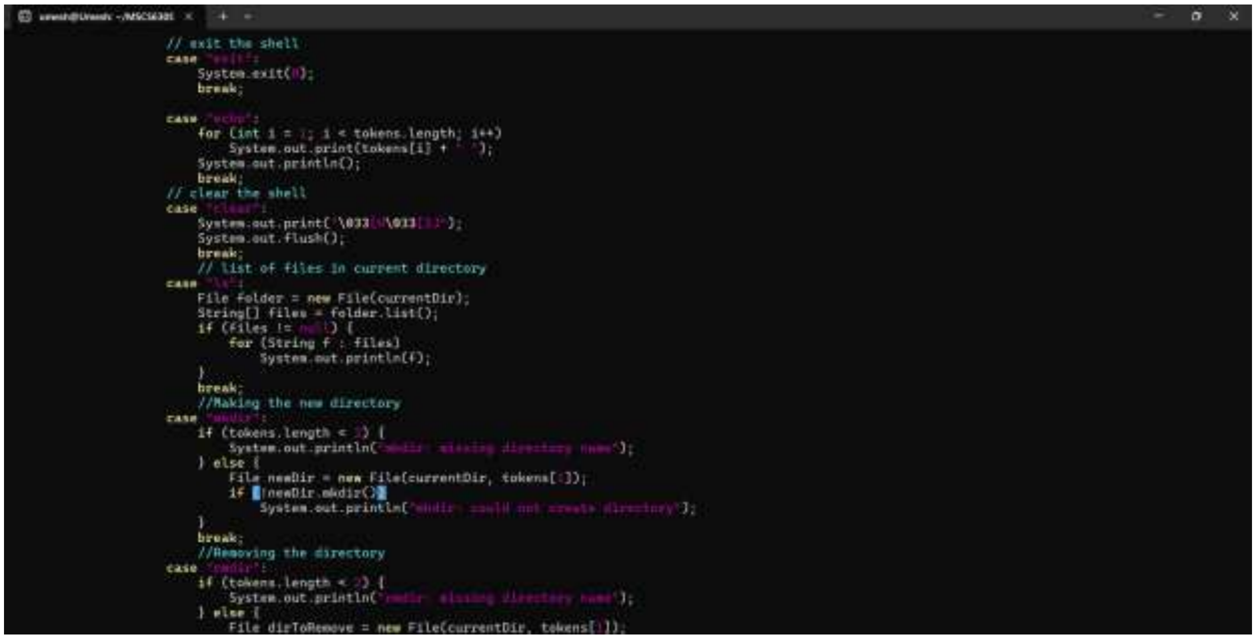
Source code



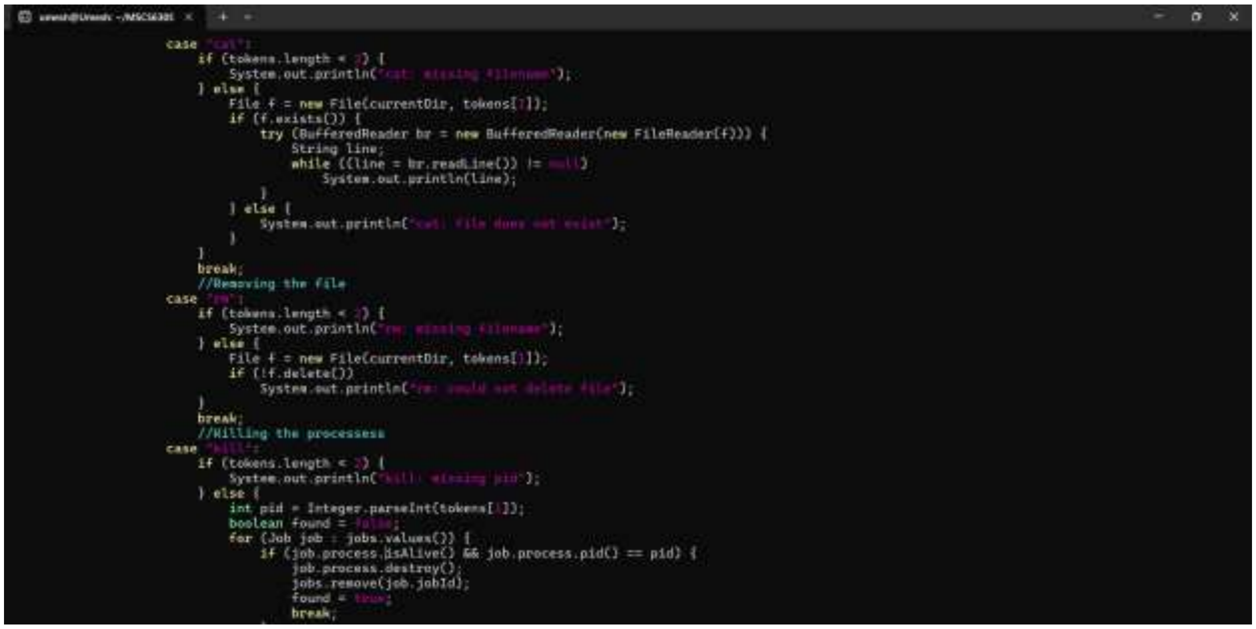
```java
//Umesh Dhakal
//Deliverable 1- shell implementation using Java
//MSCS630
//Date:09/01/2025

//Imports
import java.io.*;
import java.util.*;
import java.util.concurrent.*;

class Job {
    int jobId;
    Process process;
    String command;

    Job(int jobId, Process process, String command) {
        this.jobId = jobId;
        this.process = process;
        this.command = command;
    }
}
//Shell IMplementation class
public class MSCS630shellimplementation {

    private static Map<Integer, Job> jobs = new ConcurrentHashMap<>();
    private static int jobCounter = 1;

    public static void main(String[] args) throws IOException {
        Scanner scanner = new Scanner(System.in);
        String currentDir = System.getProperty("user.dir");

        while (true) {
            System.out.print("java_shell> ");
            String input = scanner.nextLine().trim();
            if (input.isEmpty()) continue;

            // built-in commands/utilities
            boolean background = input.endsWith("&");
            if (background) input = input.substring(0, input.length() - 1).trim();
```



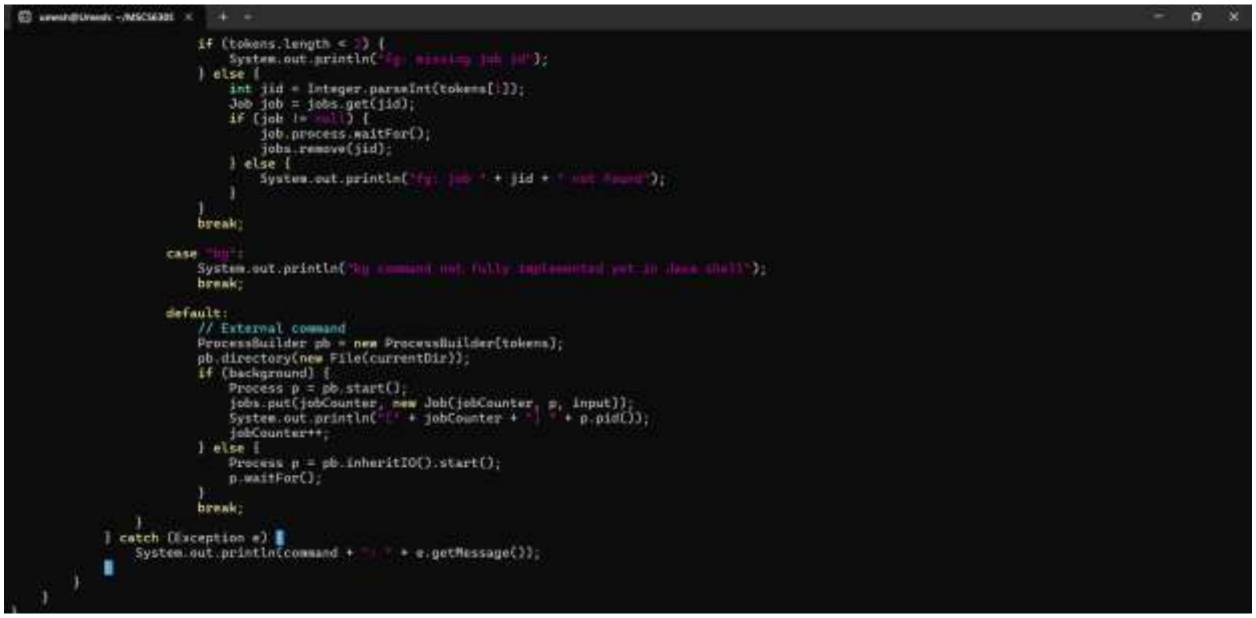
```
        boolean background = input.endsWith("&");
        if (background) input = input.substring(0, input.length() - 1).trim();

        String[] tokens = input.split("\\s+");
        String command = tokens[0];
         // Using switch break to implement all command and utilities
        try {
            switch (command) {
                case "cd":
                    if (tokens.length < 2) {
                        System.out.println("cd: missing argument");
                    } else {
                        File dir = new File(tokens[1]);
                        if (dir.exists() && dir.isDirectory()) {
                            currentDir = dir.getAbsolutePath();
                            System.setProperty("user.dir", currentDir);
                        } else {
                            System.out.println("cd: directory does not exist");
                        }
                    }
                    break;
                    // to show the current directory path

                case "pwd":
                    System.out.println(currentDir);
                    break;
                // exit the shell
                case "exit":
                    System.exit(0);
                    break;

                case "echo":
                    for (int i = 1; i < tokens.length; i++)
                        System.out.print(tokens[i] + " ");
                    System.out.println();
                    break;
                // clear the shell
                case "clear":
                    System.out.print("\033[H\033[2J");
                    System.out.flush();
```



```
                // exit the shell
                case "exit":
                    System.exit(0);
                    break;

                case "echo":
                    for (int i = 1; i < tokens.length; i++)
                        System.out.print(tokens[i] + " ");
                    System.out.println();
                    break;
                // clear the shell
                case "clear":
                    System.out.print("\033[H\033[2J");
                    System.out.flush();
                    break;
                    // list of files in current directory
                case "ls":
                    File folder = new File(currentDir);
                    String[] files = folder.list();
                    if (files != null) {
                        for (String f : files)
                            System.out.println(f);
                    }
                    break;
                    //Making the new directory
                case "mkdir":
                    if (tokens.length < 2) {
                        System.out.println("mkdir: missing directory name");
                    } else {
                        File newDir = new File(currentDir, tokens[1]);
                        if (!newDir.mkdir())
                            System.out.println("mkdir: could not create directory");
                    }
                    break;
                    //Removing the directory
                case "rmdir":
                    if (tokens.length < 2) {
                        System.out.println("rmdir: missing directory name");
                    } else {
                        File dirToRemove = new File(currentDir, tokens[1]);
```



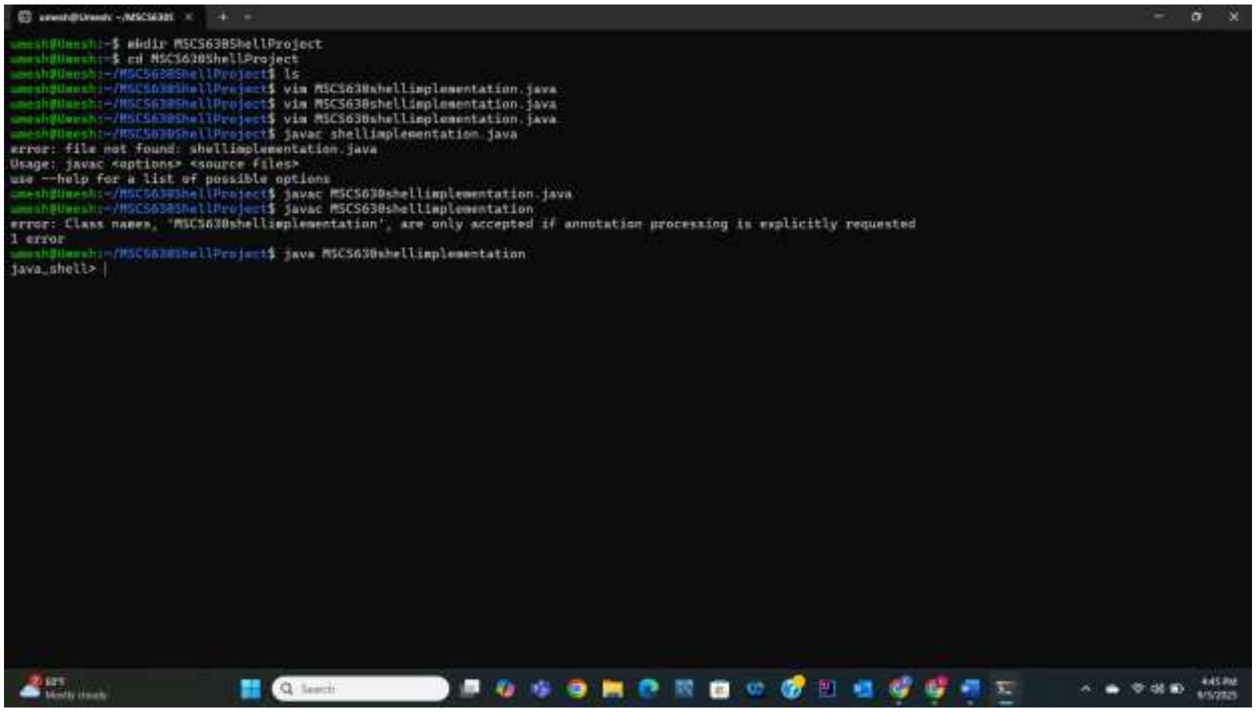
```
                case "cat":
                    if (tokens.length < 2) {
                        System.out.println("cat: missing filename");
                    } else {
                        File f = new File(currentDir, tokens[1]);
                        if (f.exists()) {
                            try (BufferedReader br = new BufferedReader(new FileReader(f))) {
                                String line;
                                while ((line = br.readLine()) != null)
                                    System.out.println(line);
                            }
                        } else {
                            System.out.println("cat: file does not exist");
                        }
                    }
                    break;
                    //Removing the file
                case "rm":
                    if (tokens.length < 2) {
                        System.out.println("rm: missing filename");
                    } else {
                        File f = new File(currentDir, tokens[1]);
                        if (!f.delete())
                            System.out.println("rm: could not delete file");
                    }
                    break;
                    //Killing the processess
                case "kill":
                    if (tokens.length < 2) {
                        System.out.println("kill: missing pid");
                    } else {
                        int pid = Integer.parseInt(tokens[1]);
                        boolean found = false;
                        for (Job job : jobs.values()) {
                            if (job.process.isAlive() && job.process.pid() == pid) {
                                job.process.destroy();
                                jobs.remove(job.jobId);
                                found = true;
                                break;
```



```
                    if (tokens.length < 2) {
                        System.out.println("fg: missing job id");
                    } else {
                        int jid = Integer.parseInt(tokens[1]);
                        Job job = jobs.get(jid);
                        if (job != null) {
                            job.process.waitFor();
                            jobs.remove(jid);
                        } else {
                            System.out.println("fg: job " + jid + " not found");
                        }
                    }
                    break;

                case "bg":
                    System.out.println("bg command not fully implemented yet in Java shell");
                    break;

                default:
                    // External command
                    ProcessBuilder pb = new ProcessBuilder(tokens);
                    pb.directory(new File(currentDir));
                    if (background) {
                        Process p = pb.start();
                        jobs.put(jobCounter, new Job(jobCounter, p, input));
                        System.out.println("[" + jobCounter + "] " + p.pid());
                        jobCounter++;
                    } else {
                        Process p = pb.inheritIO().start();
                        p.waitFor();
                    }
                    break;
            }
        } catch (Exception e) {
            System.out.println(command + ": " + e.getMessage());
        }
    }
}
```

Compiling the java file

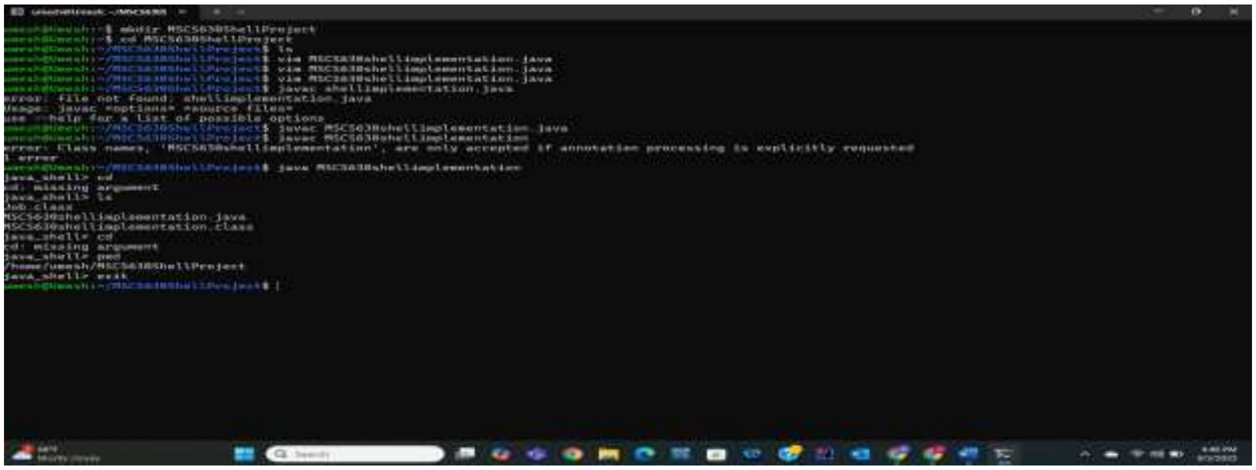
With the help of java compiling command javac shellimplementation.java followed by java shell implementation we compile the source file and ran all the built-in command and foreground and background process management commands.



```
umesh@Umesh:~$ mkdir MSCS638ShellProject
umesh@Umesh:~$ cd MSCS638ShellProject
umesh@Umesh:~/MSCS638ShellProject$ ls
umesh@Umesh:~/MSCS638ShellProject$ vim MSCS638shellimplementation.java
umesh@Umesh:~/MSCS638ShellProject$ vim MSCS638shellimplementation.java
umesh@Umesh:~/MSCS638ShellProject$ vim MSCS638shellimplementation.java
umesh@Umesh:~/MSCS638ShellProject$ javac shellimplementation.java
error: file not found: shellimplementation.java
Usage: javac <options> <source files>
use --help for a list of possible options
umesh@Umesh:~/MSCS638ShellProject$ javac MSCS638shellimplementation.java
umesh@Umesh:~/MSCS638ShellProject$ javac MSCS638shellimplementation
error: Class names, 'MSCS638shellimplementation', are only accepted if annotation processing is explicitly requested
1 error
umesh@Umesh:~/MSCS638ShellProject$ java MSCS638shellimplementation
java_shell> |
```

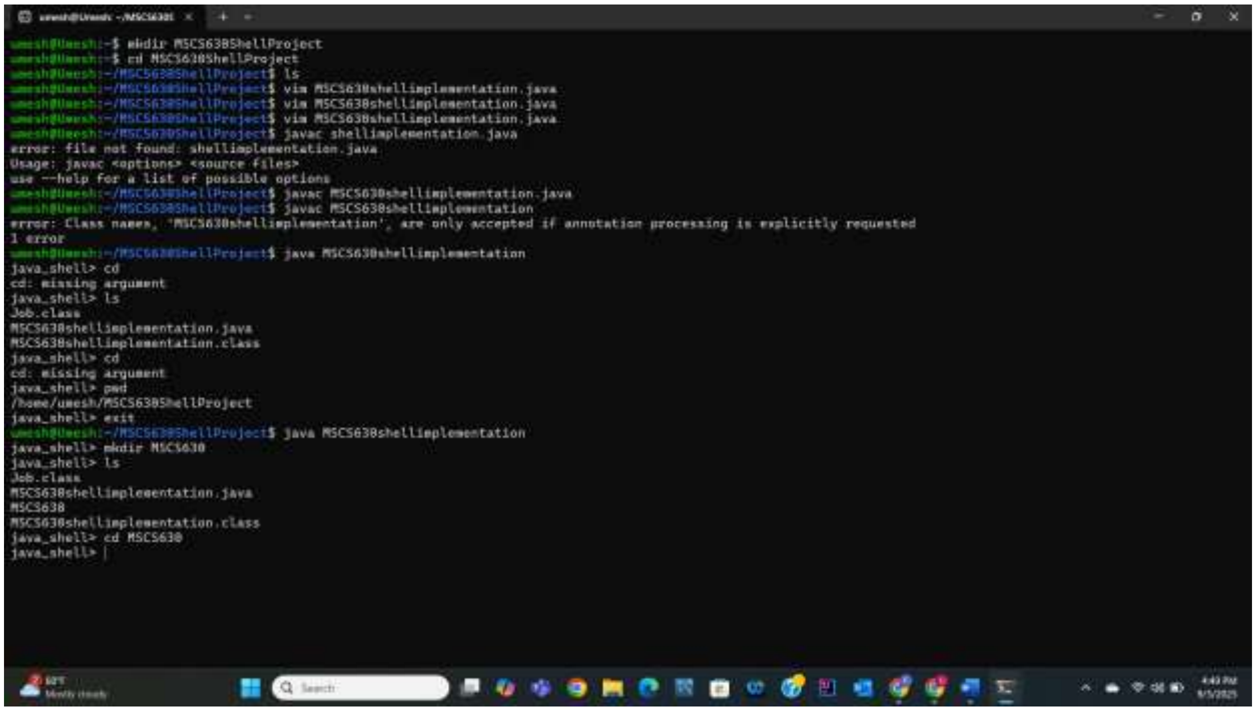
Running the built in command

Ls, pwd, cd, exit



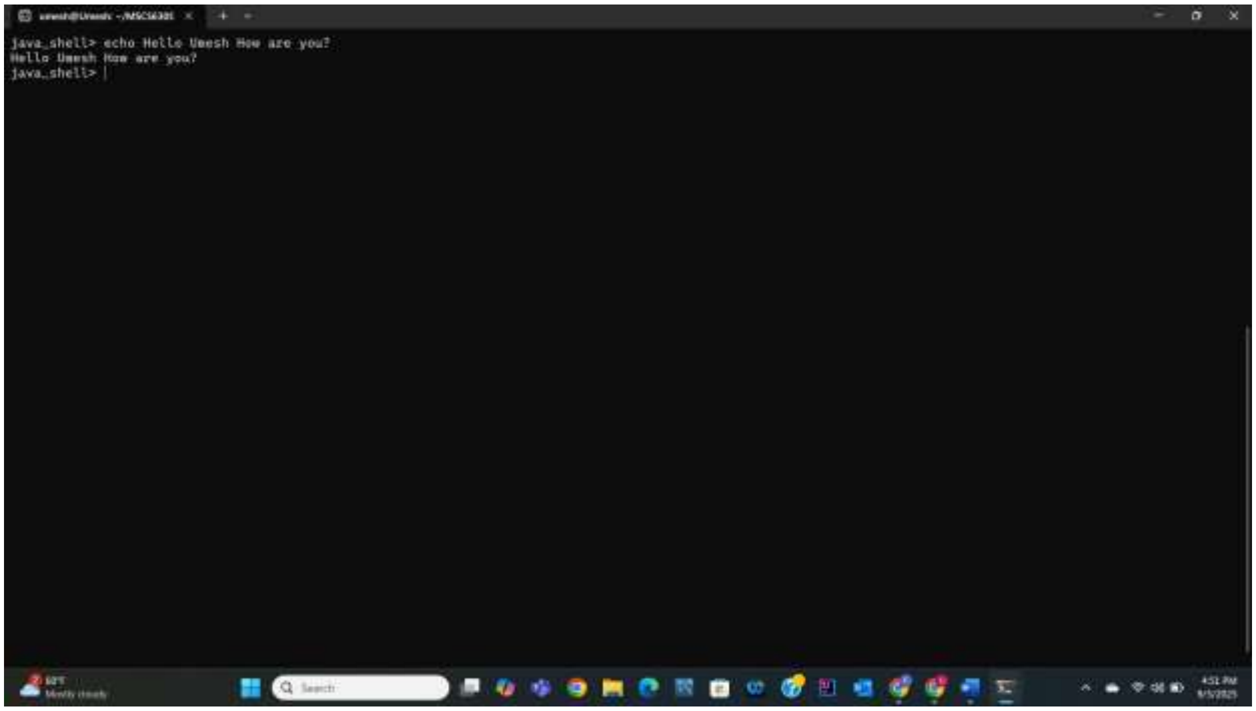
```
umesh@Umesh:~$ mkdir MSCS638ShellProject
umesh@Umesh:~$ cd MSCS638ShellProject
umesh@Umesh:~/MSCS638ShellProject$ ls
umesh@Umesh:~/MSCS638ShellProject$ vim MSCS638shellimplementation.java
umesh@Umesh:~/MSCS638ShellProject$ vim MSCS638shellimplementation.java
umesh@Umesh:~/MSCS638ShellProject$ vim MSCS638shellimplementation.java
umesh@Umesh:~/MSCS638ShellProject$ javac shellimplementation.java
error: file not found: shellimplementation.java
Usage: javac <options> <source files>
use --help for a list of possible options
umesh@Umesh:~/MSCS638ShellProject$ javac MSCS638shellimplementation.java
umesh@Umesh:~/MSCS638ShellProject$ javac MSCS638shellimplementation
error: Class names, 'MSCS638shellimplementation', are only accepted if annotation processing is explicitly requested
1 error
umesh@Umesh:~/MSCS638ShellProject$ java MSCS638shellimplementation
java_shell> cd
cd: missing argument
java_shell> ls
Job.class
MSCS638shellimplementation.java
MSCS638shellimplementation.class
java_shell> cd
cd: missing argument
java_shell> pwd
/home/umesh/MSCS638ShellProject
java_shell> exit
umesh@Umesh:~/MSCS638ShellProject$ |
```

Mkdir, cd, ls



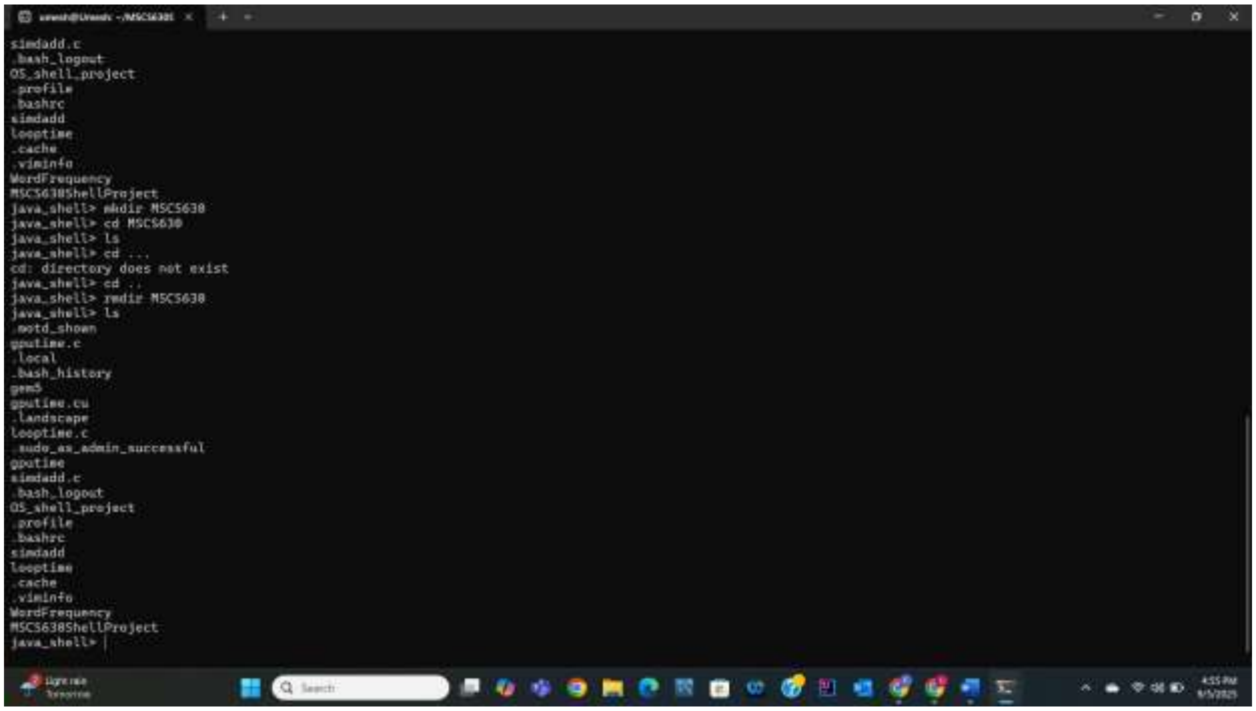
```
umesh@Umesh:~$ mkdir MSCS630SShellProject
umesh@Umesh:~$ cd MSCS630ShellProject
umesh@Umesh:~/MSCS630ShellProject$ ls
umesh@Umesh:~/MSCS630ShellProject$ vim MSCS630shellimplementation.java
umesh@Umesh:~/MSCS630ShellProject$ vim MSCS630shellimplementation.java
umesh@Umesh:~/MSCS630ShellProject$ vim MSCS630shellimplementation.java
umesh@Umesh:~/MSCS630ShellProject$ javac shellimplementation.java
error: file not found: shellimplementation.java
Usage: javac <options> <source files>
use --help for a list of possible options
umesh@Umesh:~/MSCS630ShellProject$ javac MSCS630shellimplementation.java
umesh@Umesh:~/MSCS630ShellProject$ javac MSCS630shellimplementation
error: Class names, 'MSCS630shellimplementation', are only accepted if annotation processing is explicitly requested
1 error
umesh@Umesh:~/MSCS630ShellProject$ java MSCS630shellimplementation
java_shell> cd
cd: missing argument
java_shell> ls
Job.class
MSCS630shellimplementation.java
MSCS630shellimplementation.class
java_shell> cd
cd: missing argument
java_shell> pwd
/home/umesh/MSCS630SShellProject
java_shell> exit
umesh@Umesh:~/MSCS630ShellProject$ java MSCS630shellimplementation
java_shell> mkdir MSCS630
java_shell> ls
Job.class
MSCS630shellimplementation.java
MSCS630
MSCS630shellimplementation.class
java_shell> cd MSCS630
java_shell> |
```

echo

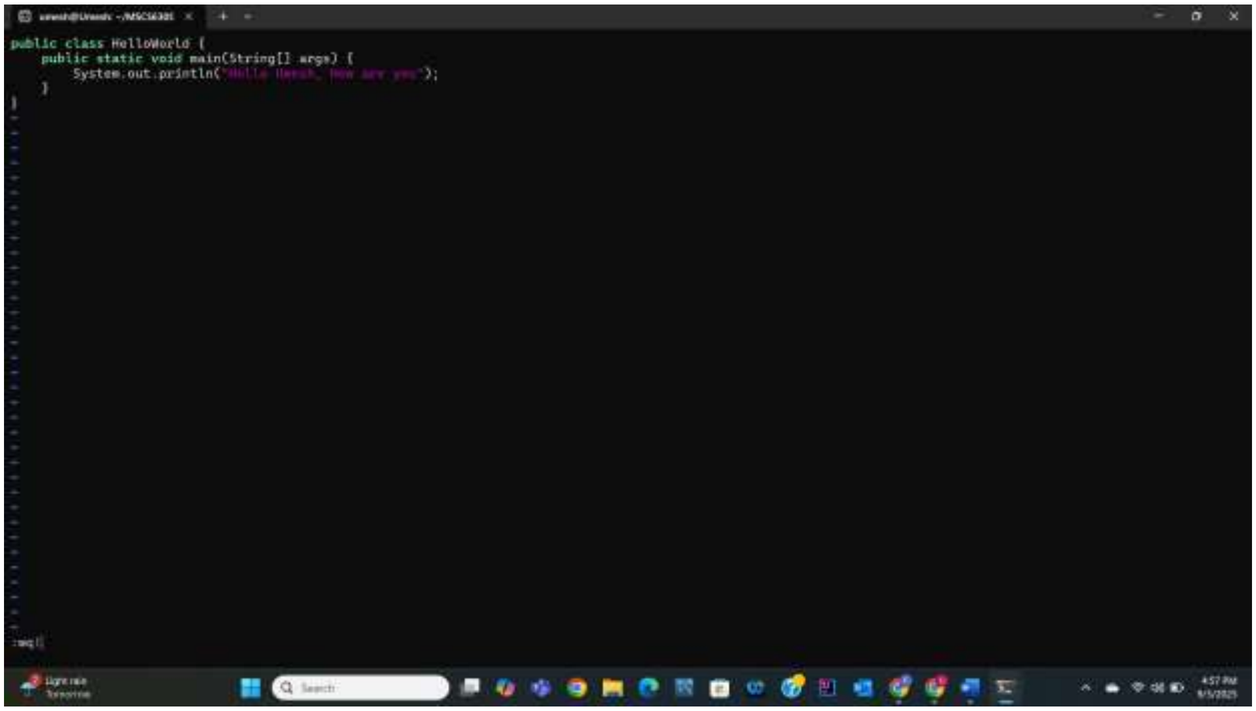


```
java_shell> echo Hello Umesh How are you?
Hello Umesh How are you?
java_shell> |
```

Rmdir

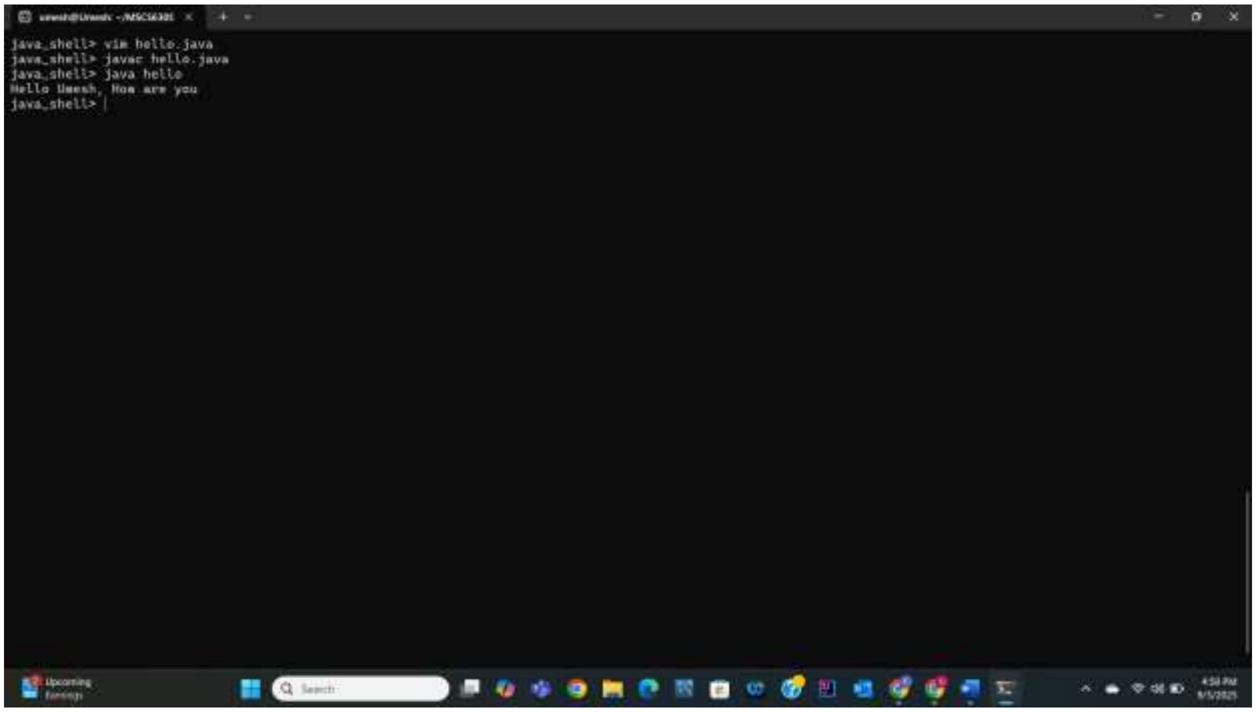


```
simdadd.c
.bash_logout
OS_shell_project
.profile
.bashrc
simdadd
looptime
.cache
.viminfo
WordFrequency
MSCS638ShellProject
java_shell> mkdir MSCS638
java_shell> cd MSCS638
java_shell> ls
java_shell> cd ...
cd: directory does not exist
java_shell> cd ..
java_shell> rmdir MSCS638
java_shell> ls
.motd_shown
gputime.c
.local
.bash_history
gem5
gputime.cu
.landscape
looptime.c
.sudo_as_admin_successful
gputime
simdadd.c
.bash_logout
OS_shell_project
.profile
.bashrc
simdadd
looptime
.cache
.viminfo
WordFrequency
MSCS638ShellProject
java_shell> |
```



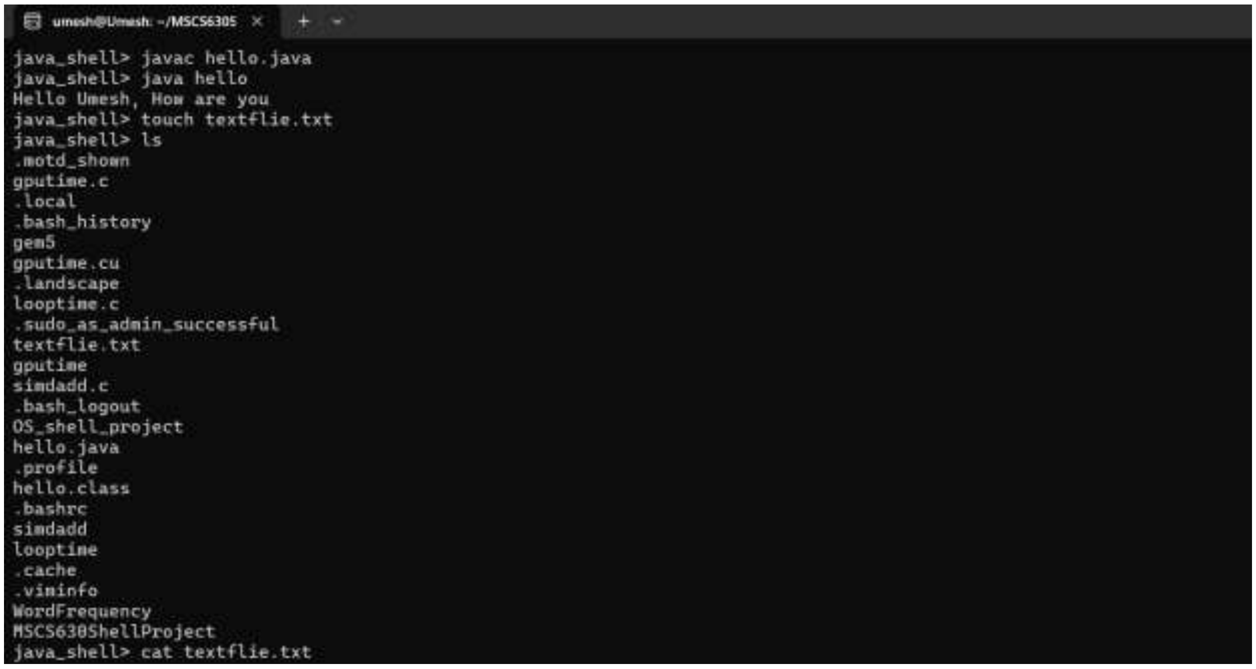
```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello Umesh, how are you");
    }
}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
:wq
```

Creating the java file and compiling it inside the java shell

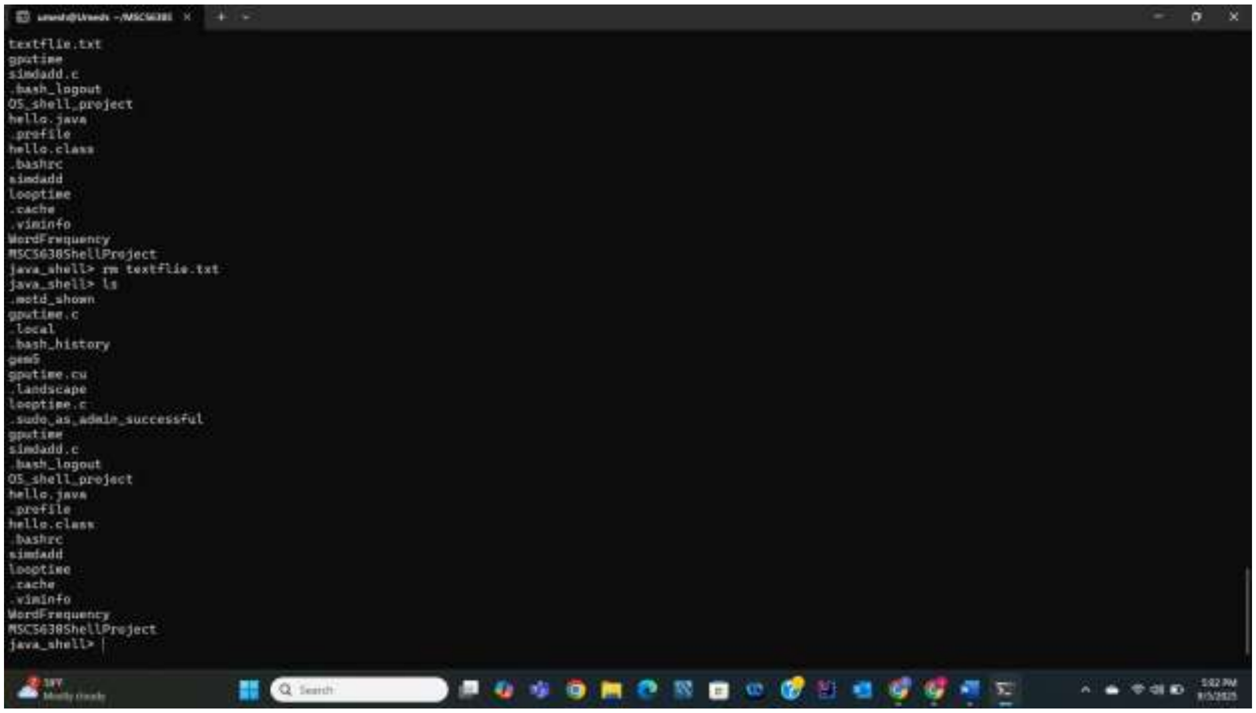


```
java_shell> vim hello.java
java_shell> javac hello.java
java_shell> java hello
Hello Umesh, How are you
java_shell> |
```

Creating text files, copying text file- touch, cat

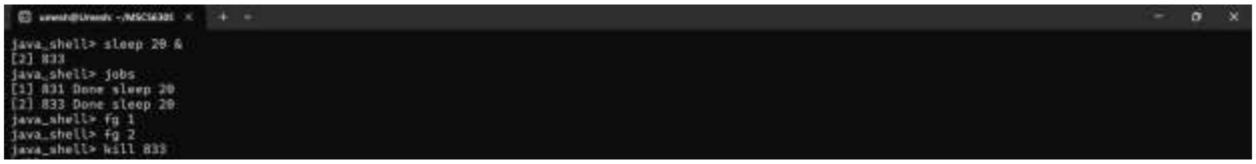


```
java_shell> javac hello.java
java_shell> java hello
Hello Umesh, How are you
java_shell> touch textflie.txt
java_shell> ls
.motd_shown
gputime.c
.local
.bash_history
gem5
gputime.cu
.landscape
looptime.c
.sudo_as_admin_successful
textflie.txt
gputime
simdadd.c
.bash_logout
OS_shell_project
hello.java
.profile
hello.class
.bashrc
simdadd
looptime
.cache
.viminfo
WordFrequency
MSCS6308ShellProject
java_shell> cat textflie.txt
```

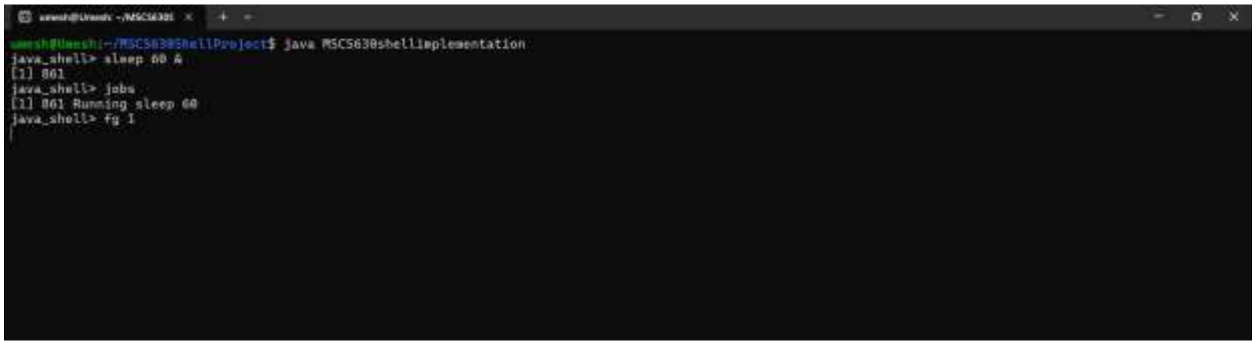


```
textflie.txt
gputime
simdadd.c
.bash_logout
OS_shell_project
hello.java
.profile
hello.class
.bashrc
simdadd
looptime
.cache
.viminfo
WordFrequency
MSCS630ShellProject
java_shell> rm textflie.txt
java_shell> ls
.motd_shown
gputime.c
.local
.bash_history
gem5
gputime.cu
.landscape
looptime.c
.sudo_as_admin_successful
gputime
simdadd.c
.bash_logout
OS_shell_project
hello.java
.profile
hello.class
.bashrc
simdadd
looptime
.cache
.viminfo
WordFrequency
MSCS630ShellProject
java_shell> |
```

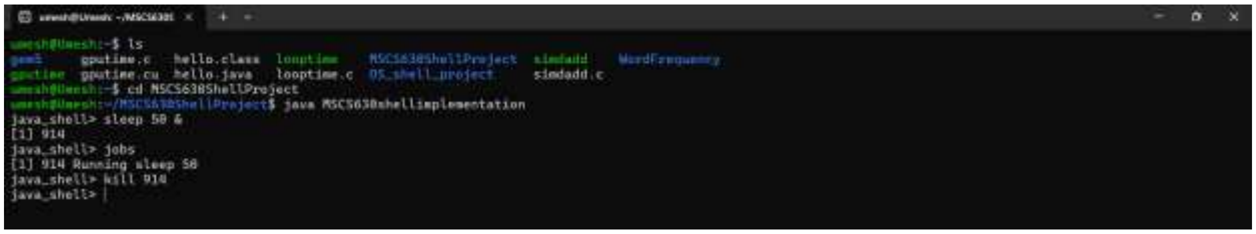
Process Management



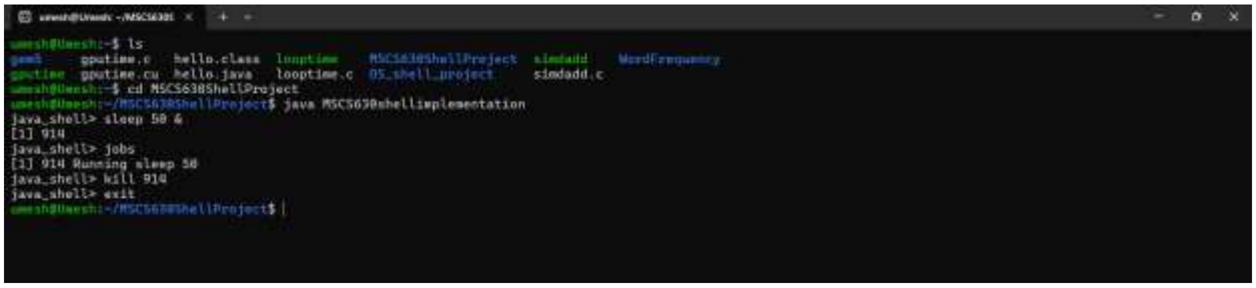
```
java_shell> sleep 20 &
[2] 833
java_shell> jobs
[1] 831 Done sleep 20
[2] 833 Done sleep 20
java_shell> fg 1
java_shell> fg 2
java_shell> kill 833
```



```
umesh@Umesh:~/MSCS630ShellProject$ java MSCS630shellimplementation
java_shell> sleep 60 &
[1] 861
java_shell> jobs
[1] 861 Running sleep 60
java_shell> fg 1
|
```



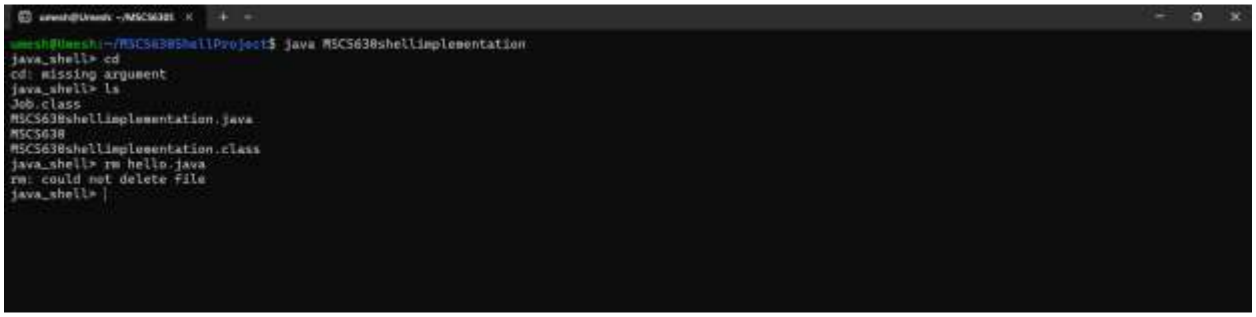
```
umesh@Umesh:~$ ls
gem5     gputime.c   hello.class  looptime    MSCS630ShellProject  simdadd    WordFrequency
gputime  gputime.cu  hello.java   looptime.c  OS_shell_project     simdadd.c
umesh@Umesh:~$ cd MSCS630ShellProject
umesh@Umesh:~/MSCS630ShellProject$ java MSCS630shellimplementation
java_shell> sleep 50 &
[1] 914
java_shell> jobs
[1] 914 Running sleep 50
java_shell> kill 914
java_shell> |
```



```
umesh@Umesh:~$ ls
gem5      gputime.c    hello.class  looptime     MSCS630ShellProject  simdadd    WordFrequency
gputime   gputime.cu   hello.java   looptime.c   OS_shell_project     simdadd.c
umesh@Umesh:~$ cd MSCS630ShellProject
umesh@Umesh:~/MSCS630ShellProject$ java MSCS630shellimplementation
java_shell> sleep 50 &
[1] 914
java_shell> jobs
[1] 914 Running sleep 50
java_shell> kill 914
java_shell> exit
umesh@Umesh:~/MSCS630ShellProject$ |
```

Error Handling

The shell gives us a clear message if there is a command issue. If we type wrong command that shell does not support, it will give us a message saying wrong command input.



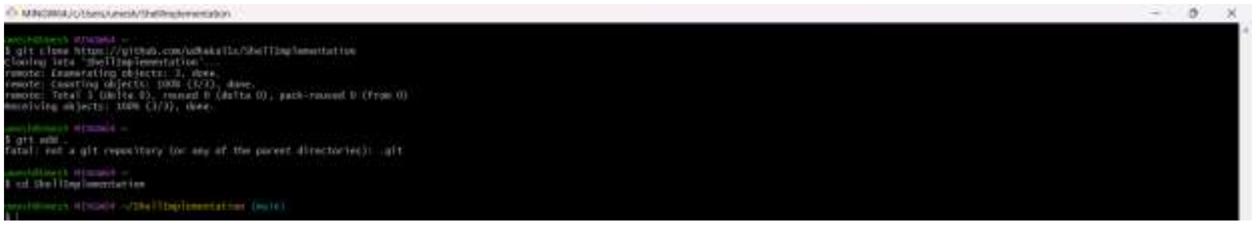
```
umesh@Umesh:~/MSCS630ShellProject$ java MSCS630shellimplementation
java_shell> cd
cd: missing argument
java_shell> ls
Job.class
MSCS630shellimplementation.java
MSCS630
MSCS630shellimplementation.class
java_shell> rm hello.java
rm: could not delete file
java_shell> |
```

Challenges

The main challenge we face was implementing foreground and background function. Our implementation supports background processes where if we add & at the end of the command, it will not wait for the process to finish. Implementing this part was little bit challenging. Also, in the beginning we didn't have java path variable set up properly and I kept on installing different dependencies thinking some dependencies were missing but that was not the case.

Git

Using the git version control we committed all the source files in the git repository.



```
[illegible] MINGW64 ~
$ git clone https://github.com/udhakalls/ShellImplementation
Cloning into 'ShellImplementation'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.

[illegible] MINGW64 ~
$ git add .
fatal: not a git repository (or any of the parent directories): .git

[illegible] MINGW64 ~
$ cd ShellImplementation

[illegible] MINGW64 ~/ShellImplementation (main)
$ |
```