

Exercise 9

hariesh - 23110344

Aim:

1. The Python script validates user credentials from credentials.txt. If valid, it writes command-line argument content to a specified file.
2. The script reads a file with 10 students and their marks, randomly assigned. It creates two files: best_performers for students with marks above 90, and low_performers for those with marks below 40. Exception handling ensures robust file operations.

Algorithm:

1. Q1:

Initialize: Set up the file (validation.txt) and input scanner.

Check Args: Ensure at least three arguments are provided (username, password, and content).

Read Args: Extract username, password, and content from arguments.

Validate: Check credentials against validation.txt. Print "Access Denied" if not found.

Write to File: If credentials are valid, write the content to secret.txt.

Handle exceptions as needed.

Close Resources: Close the scanner and file writers.

2. Q2:

Generate Marks: Create marks.txt with 10 student names and random marks (1-100).

Write Marks: Save the names and marks to marks.txt.

Classify Performances:

Open marks.txt and read the data.

Create best_performance.txt for students with marks > 90.

Create low_performance.txt for students with marks < 40.

Write Results: Write names to the appropriate files based on their marks.

Handle Exceptions: Manage file I/O errors and print appropriate messages.

Source Code:

1. Q1:

```
package Exercise9.Q1;

import java.util.Scanner;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.FileWriter;

public class main_program {
    public static void main(String[] args) {
        File validationFile = new File("validation.txt");
        Scanner scanner = new Scanner(System.in);
        String userName, password, data, userNameCurrent, passwordCurrent;
        boolean found = false;

        String content = "";

        if (args.length < 3){

            System.out.println("Insufficient number of args");
            scanner.close();
            return;
        }

        userName = args[0];
        password = args[1];
```

```

for (int i = 2; i < args.length; i++) content += args[i] + " ";

try {
    Scanner readValidationFile = new Scanner(validationFile);

    while (readValidationFile.hasNextLine()){
        data = readValidationFile.nextLine();

        userNameCurrent = data.split(" ")[0];
        passwordCurrent = data.split(" ")[1];
        if (userNameCurrent.equals(userName) &&
            passwordCurrent.equals(password)){
            System.out.println("Found: " + userNameCurrent + passwordCurrent);
            found = true;
            break;
        }
    }

    readValidationFile.close();
} catch (FileNotFoundException e){
    System.out.println("Error: " + e.getMessage());
}

if(!found) System.out.println("Access Denied");

if(found){

    try {

        File secretFile = new File("secret.txt");
        FileWriter secretFileWriter = new FileWriter("secret.txt");

        secretFile.createNewFile();
        secretFileWriter.write(content);
        System.out.println("Written Successfully");
    }
}

```

```
secretFileWriter.close();

} catch (IOException e){

System.out.println("Error: " + e.getMessage());
}
}

scanner.close();
}
}
```

2. Q2:

```
package Exercise9.Q2;

import java.util.Random;
import java.util.Scanner;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.FileNotFoundException;

public class main_program {

    public static void main(String[] args) {

        Random random = new Random();
        String[] names = {"Liam", "Olivia", "Noah", "Emma", "Oliver", "Charlotte",
            "James", "Amelia", "Elijah", "Sophia"};
        String data, studentName;
        int marks;
        try {

            File marksFile = new File("marks.txt");
```

```

FileWriter marksFileWriter = new FileWriter(marksFile);

marksFile.createNewFile();

for (String name : names) {
    marks = random.nextInt(100) + 1;
    data = name + " " + marks + "\n";

    marksFileWriter.write(data);
}

marksFileWriter.close();

} catch (IOException e){

    System.out.println("Error: " + e.getMessage());
}

try {

    File marksFile = new File("marks.txt");
    Scanner readMarks = new Scanner(marksFile);

    File bestPerformance = new File("best performace.txt");
    bestPerformance.createNewFile();
    FileWriter bestPerformanceWriter = new FileWriter(bestPerformance);

    File lowPerformance = new File("low performace.txt");
    lowPerformance.createNewFile();
    FileWriter lowPerformanceWriter = new FileWriter(lowPerformance);

    while (readMarks.hasNextLine()){

        data = readMarks.nextLine();
        marks = Integer.parseInt(data.split(" ")[1]);
        studentName = data.split(" ")[0];
    }
}

```

```
if (marks > 90) bestPerformanceWriter.write(studentName + "\n");
if (marks < 40) lowPerformanceWriter.write(studentName + "\n");
}

readMarks.close();
bestPerformanceWriter.close();
lowPerformanceWriter.close();

} catch (FileNotFoundException e){


System.out.println("Error: " + e.getMessage());
} catch (IOException e){


System.out.println("Error: " + e.getMessage());
}
}
}
```

Output:


1. Q1:


```
snucse@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/harizzzesh/Exercise9/Q1$ javac main_program.java
snucse@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/harizzzesh/Exercise9/Q1$ java main_program.java hariesh 123 hello this is a sample text
Found: hariesh123
Written Successfully
snucse@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/harizzzesh/Exercise9/Q1$
```

 validation.txt ✕

Exercise9 > Q1 >  validation.txt

```
1 hariesh 123
2 arun 321
3 userhello 345
4 userhi 987
```

 secret.txt ✕

Exercise9 > Q1 >  secret.txt

```
1 hello this is a sample text
```

2. Q2:

```
● snucse@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/harizzzesh/Exercise9/Q2$ javac main_program.java
● snucse@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/harizzzesh/Exercise9/Q2$ java main_program.java
○ snucse@snucse-HP-Pro-Tower-400-G9-PCI-Desktop-PC:~/harizzzesh/Exercise9/Q2$
```

Exercise9 > Q2 > marks.txt

```
1  Liam 12
2  Olivia 78
3  Noah 92
4  Emma 97
5  Oliver 27
6  Charlotte 87
7  James 57
8  Amelia 7
9  Elijah 18
10 Sophia 98
11
```

Exercise9 > Q2 > best performace.txt

```
1  Noah
2  Emma
3  Sophia
4
```

Exercise9 > Q2 > low performace.txt

```
1  Liam
2  Oliver
3  Amelia
4  Elijah
5
```

Result:

1. The program validates a username and password against credentials in a file. If the validation succeeds, it writes command-line arguments to a new file.
2. The program creates a file `marks.txt` with 10 students and random marks between 1 and 100. It then generates two additional files: `best_performers`, listing students with marks above 90, and `low_performers`, listing students with marks below 40. It handles file reading and writing errors with appropriate messages.