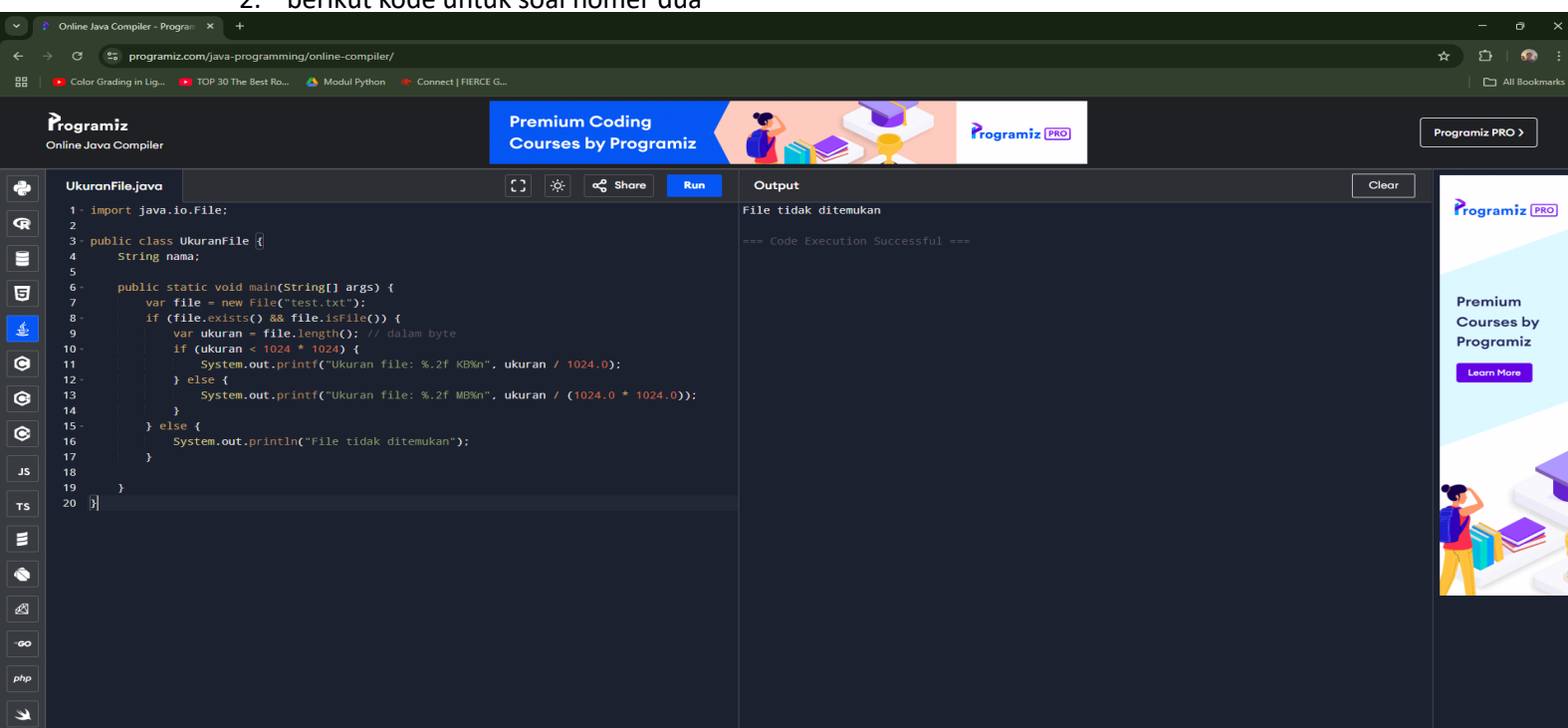


Nama : Oase Bimasena Ilhamaziiz
NIM : 245150707111059
Kelas : TI-D

1. Jika kita mengubah parameter false menjadi true yang ada di line 13 pada kode TulisFile.java, parameter tersebut adalah parameter yang disebut append. Append sendiri berarti "menambahkan", sehingga jika kita ubah parameter append yang semulanya adalah false menjadi true, maka teks baru yang dimasukkan oleh user akan ditambahkan (di- append) ke akhir isi file test.txt, tanpa menghapus isi sebelumnya. Hal tersebut terjadi karena mekanisme FileWriter, dimana append menentukan apakah file dibuka dalam mode: Append (false): Pointer file dimulai di awal file, sehingga menimpa data lama (overwrite). Append (true): Pointer file dimulai di akhir file, sehingga menulis data baru setelah teks yang sudah ada.

2. berikut kode untuk soal nomer dua



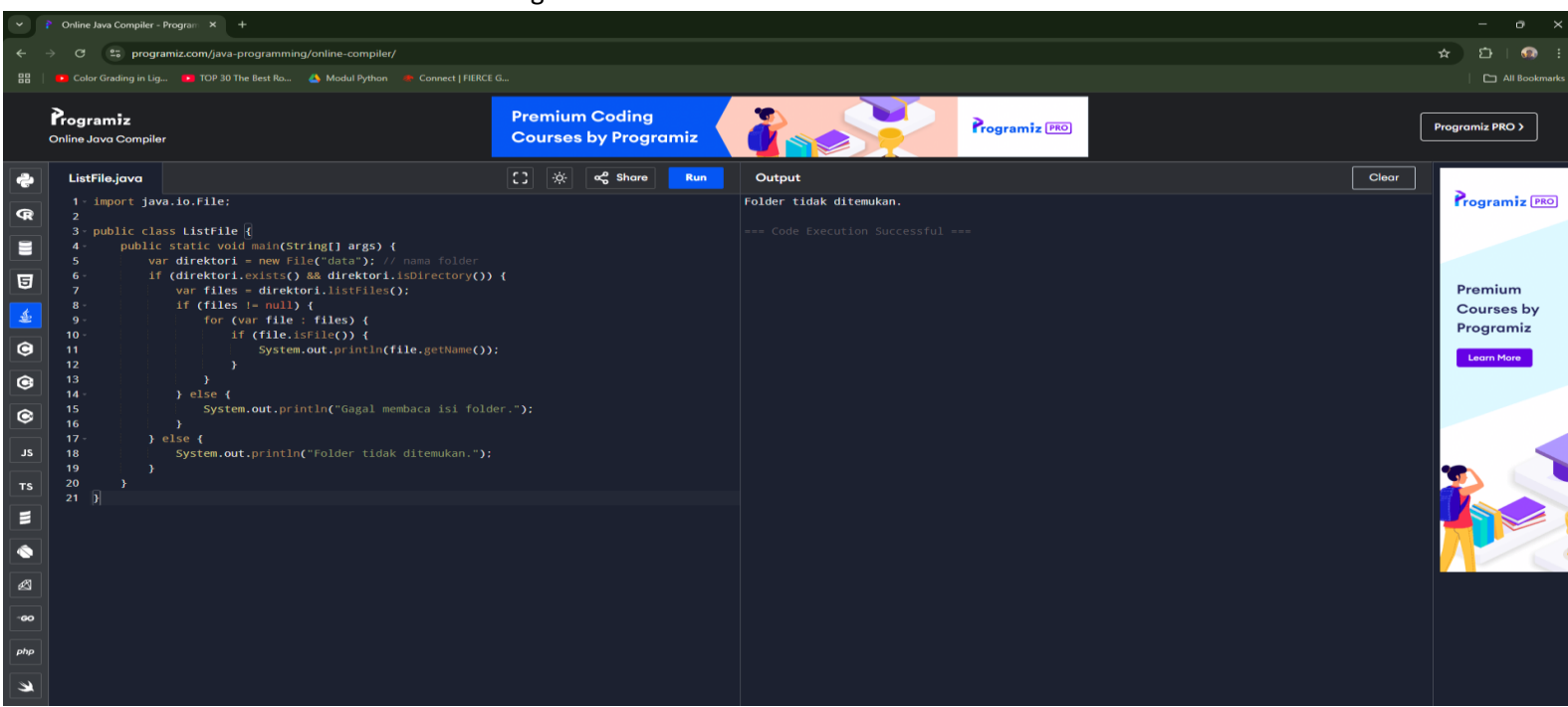
```
1- import java.io.File;
2-
3- public class UkuranFile {
4-     String nama;
5-
6-     public static void main(String[] args) {
7-         var file = new File("test.txt");
8-         if (file.exists() && file.isFile()) {
9-             var ukuran = file.length(); // dalam byte
10-            if (ukuran < 1024 * 1024) {
11-                System.out.printf("Ukuran file: %.2f KB\n", ukuran / 1024.0);
12-            } else {
13-                System.out.printf("Ukuran file: %.2f MB\n", ukuran / (1024.0 * 1024.0));
14-            }
15-        } else {
16-            System.out.println("File tidak ditemukan");
17-        }
18-    }
19- }
20- }
```

Output

File tidak ditemukan

=== Code Execution Successful ===

3. Kode untuk nomer tiga



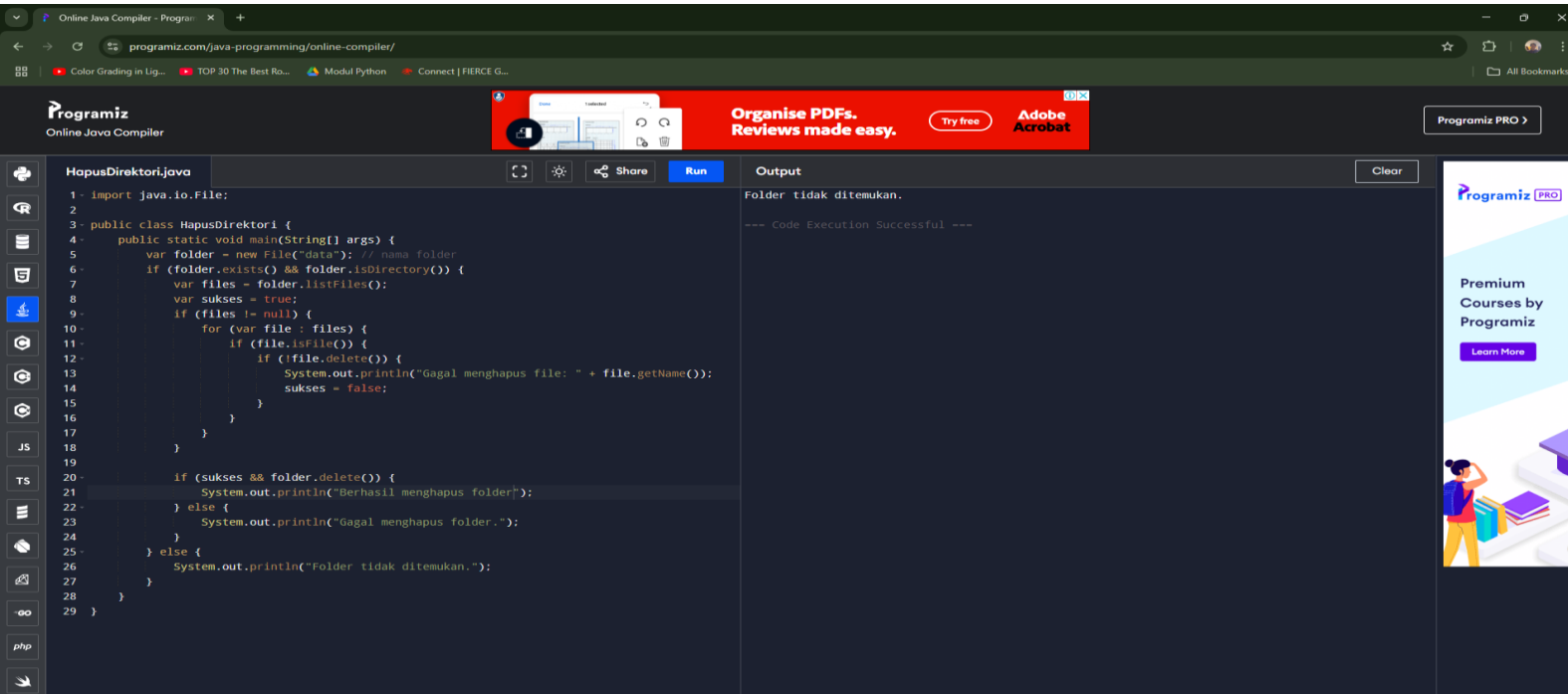
```
1- import java.io.File;
2-
3- public class ListFile {
4-     public static void main(String[] args) {
5-         var direktori = new File("data"); // nama folder
6-         if (direktori.exists() && direktori.isDirectory()) {
7-             var files = direktori.listFiles();
8-             if (files != null) {
9-                 for (var file : files) {
10-                    if (file.isFile()) {
11-                        System.out.println(file.getName());
12-                    }
13-                }
14-            } else {
15-                System.out.println("Gagal membaca isi folder.");
16-            }
17-        } else {
18-            System.out.println("Folder tidak ditemukan.");
19-        }
20-    }
21- }
```

Output

Folder tidak ditemukan.

=== Code Execution Successful ===

4. Kode program untuk soal nomer empat



```
1- import java.io.File;
2-
3- public class HapusDirektori {
4-     public static void main(String[] args) {
5-         var folder = new File("data"); // nama folder
6-         if (folder.exists() && folder.isDirectory()) {
7-             var files = folder.listFiles();
8-             var sukses = true;
9-             if (files != null) {
10-                 for (var file : files) {
11-                     if (file.isFile()) {
12-                         if (!file.delete()) {
13-                             System.out.println("Gagal menghapus file: " + file.getName());
14-                             sukses = false;
15-                         }
16-                     }
17-                 }
18-             }
19-             if (sukses && folder.delete()) {
20-                 System.out.println("Berhasil menghapus folder");
21-             } else {
22-                 System.out.println("Gagal menghapus folder.");
23-             }
24-         } else {
25-             System.out.println("Folder tidak ditemukan.");
26-         }
27-     }
28- }
29 }
```

Output

```
Folder tidak ditemukan.
--- Code Execution Successful ---
```

- masalah utama terletak pada penggunaan backslash (`\`) dalam path file tanpa proper escaping di Java. Karena backslash merupakan karakter escape dalam string Java (misalnya `\n` untuk newline), path `"C:\Data\Java\teks.txt"` akan diinterpretasikan salah oleh compiler - karakter `D`, `J`, dan `t` akan dibaca sebagai escape sequence yang tidak valid. Solusinya adalah dengan menggunakan double backslash (`\\`) untuk setiap separator path (menjadi `"C:\\Data\\Java\\teks.txt"`) atau menggunakan forward slash (`/`) yang juga didukung Java di semua sistem operasi (menjadi `"C:/Data/Java/teks.txt"`). Selain itu, meskipun penggunaan `var` diperbolehkan sejak Java 10, lebih aman menggunakan tipe eksplisit `File` untuk kompatibilitas dengan versi Java yang lebih lama.
- jika kita mencoba membaca isi file yang tidak ada tanpa melakukan pengecekan terlebih dahulu, program akan menghasilkan `FileNotFoundException` (untuk operasi I/O tradisional) atau `NoSuchFileException` (untuk NIO package). Exception ini termasuk checked exception, yang berarti compiler memaksa kita untuk menanganinya baik dengan try-catch block atau dengan mendeklarasikannya di throws clause. Jika tidak ditangani, program akan berhenti secara tiba-tiba (crash) dan menampilkan stack trace error. Solusi yang baik adalah selalu memverifikasi keberadaan file menggunakan `file.exists()` sebelum operasi baca, atau menggunakan try-with-resources untuk penanganan error yang lebih elegan sekaligus memastikan resource seperti file stream ditutup dengan benar.