



Universidad Nacional Autónoma de México
Facultad de Contaduría y Administración
Sistema Universidad Abierta y Educación a Distancia

Alumno: López Acevedo Víctor Rafael

Grupo: 9696

Materia: INFORMATICA VI

Unidad: 5

Actividad: M5-02

Fecha: 14 de mayo 2025

Actividad: Conteo de calorías

Descripción de la actividad

1. A partir del siguiente problema, resolver la solución en lenguaje Java en Android Studio, imprimir el resultado en el logcat.

Descripción del problema --- Conteo de calorías ---

Los renos de Papá Noel suelen comer comida normal para renos, pero necesitan mucha energía mágica para entregar regalos en Navidad. Por eso, su bocadillo favorito es un tipo especial de carambola que solo crece en lo profundo de la jungla. Los Elfos te han traído en su expedición anual al bosque donde crece la fruta.

Para suministrar suficiente energía mágica, la expedición debe recuperar un mínimo de cincuenta estrellas antes del 25 de diciembre. Aunque los Elfos te aseguran que la arboleda tiene mucha fruta, decides agarrar cualquier fruta que veas por el camino, por si acaso.

Recoge estrellas resolviendo acertijos. Habrá dos rompecabezas disponibles cada día en el calendario de Adviento; el segundo rompecabezas se desbloquea cuando completas el primero. Cada rompecabezas otorga una estrella. ¡Buena suerte!

La jungla debe estar demasiado cubierta de maleza y ser difícil de transitar en vehículos o acceder desde el aire; La expedición de los Elfos tradicionalmente va a pie. A medida que sus barcos se acercan a tierra, los Elfos comienzan a hacer un inventario de sus suministros. Una consideración importante es la comida, en particular, la cantidad de calorías que lleva cada Elfo (su entrada de rompecabezas).

Los Elfos se turnan para anotar el número de Calorías que contienen las distintas comidas, meriendas, raciones, etc. que han traído consigo, un elemento por línea. Cada Elfo separa su propio inventario del inventario del Elfo anterior (si lo hay) con una línea en blanco.

Por ejemplo, supongamos que los Elfos terminan de escribir las Calorías de sus artículos y terminan con la siguiente lista:

1000

2000

3000

4000

5000

6000

7000

8000

9000

10000

Esta lista representa las Calorías de la comida que llevan cinco Elfos:

El primer Elfo lleva comida con 1000, 2000 y 3000 Calorías, un total de 6000 Calorías.

El segundo elfo lleva un alimento con 4000 calorías.

El tercer Elfo lleva comida con 5000 y 6000 Calorías, un total de 11000 Calorías.

El cuarto Elfo lleva comida con 7000, 8000 y 9000 Calorías, un total de 24000 Calorías.

El quinto Elfo lleva un alimento con 10000 Calorías.

En caso de que los Elfos tengan hambre y necesiten refrigerios adicionales, necesitan saber a qué Elfo preguntar: les gustaría saber cuántas Calorías lleva el Elfo que lleva la mayor cantidad de Calorías. En el ejemplo anterior, esto es 24000 (llevado por el cuarto Elfo).

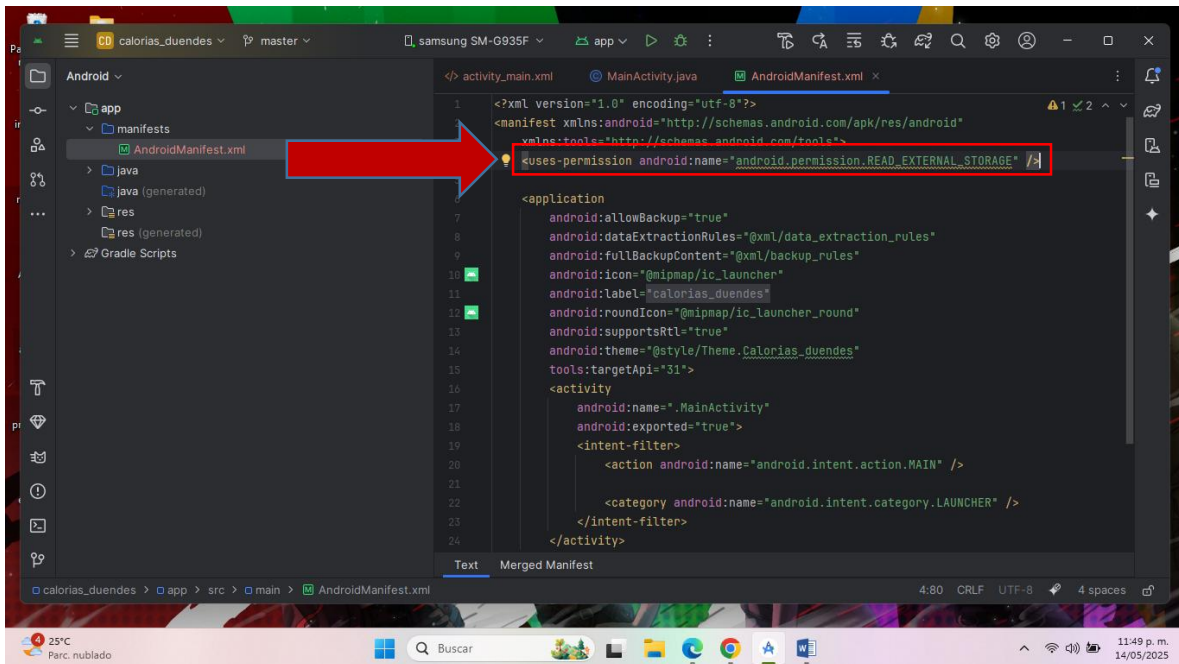
Encuentra al duende que lleva más calorías.

2. Pregunta a resolver ¿Cuántas calorías totales lleva ese elfo?

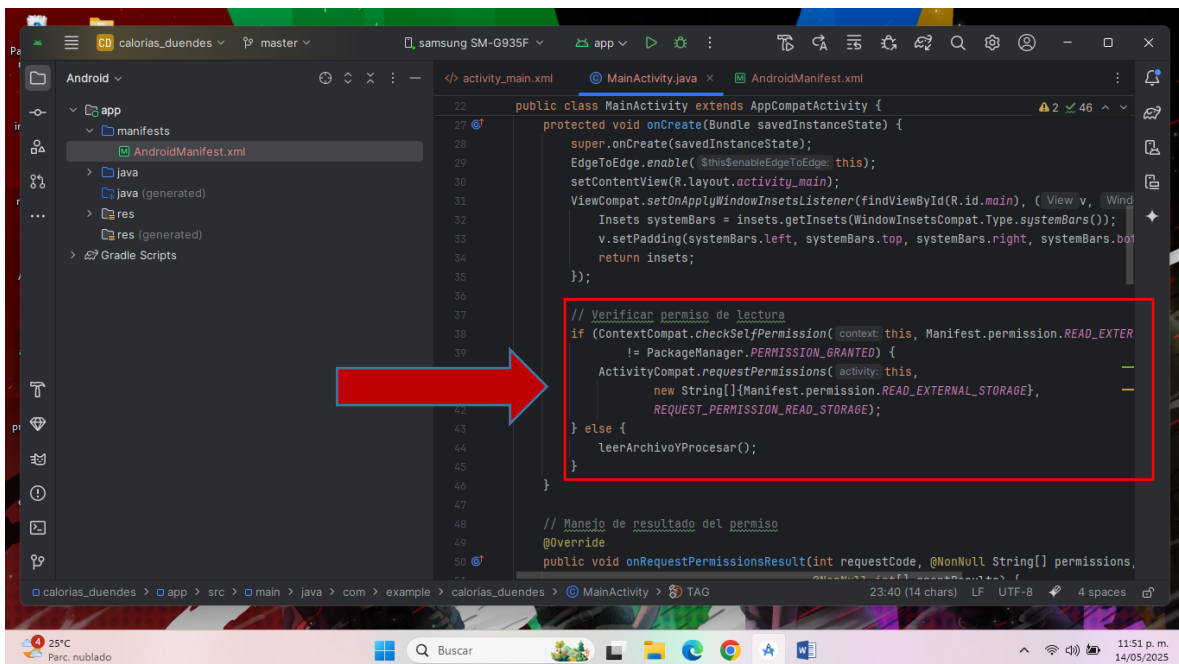
Para realizar esto descargamos un archivo que nos dio el profesor y venían varios datos para los duendes, donde cada uno traía una distinta cantidad de calorías y estas se tenían que sumar.

En mi caso decidí usar ese archivo .txt y guardarlo en una de las carpetas de la memoria del celular, para lo cual, la aplicación tiene que pedir un permiso para acceder al archivo y poder leer los datos.

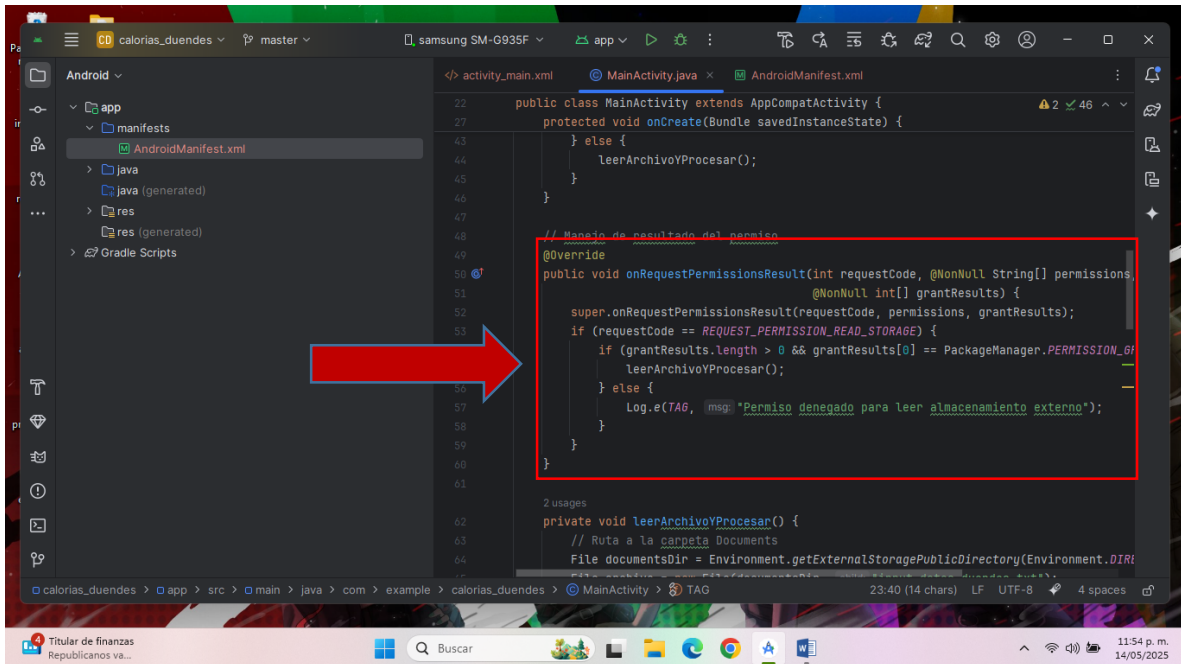
Para eso, abrimos el “AndroidManifest.xml” y agregamos una línea que es la que nos dará el permiso para leer archivos de una fuente de almacenamiento externo



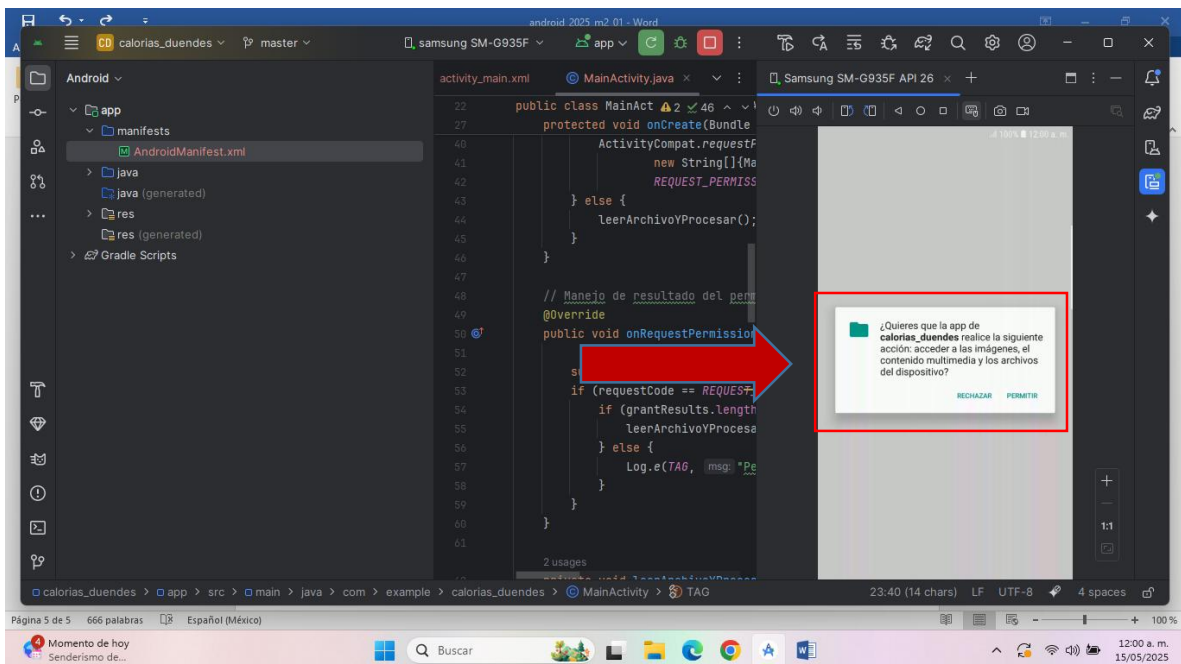
Una vez ponemos que se pueda acceder al almacenamiento externo, empezamos a codificar el código para realizar esta actividad



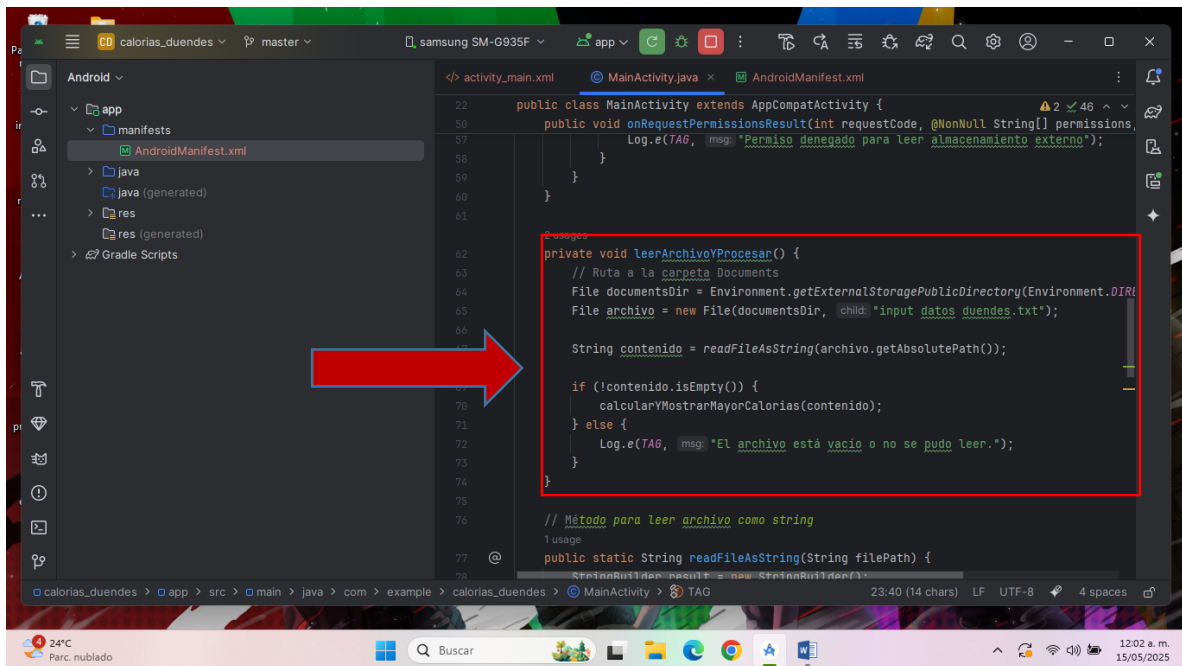
Primero necesitamos que la aplicación confirme que tiene que pedir permiso para acceder al almacenamiento externo para realizar la lectura del archivo



Una vez que se verifica, la aplicación entonces tiene que pedir el permiso al usuario para esto, así que le muestra un mensaje que le pide la autorización, en caso de ser rechazada, se mostrará un mensaje diciendo que el permiso fue denegado

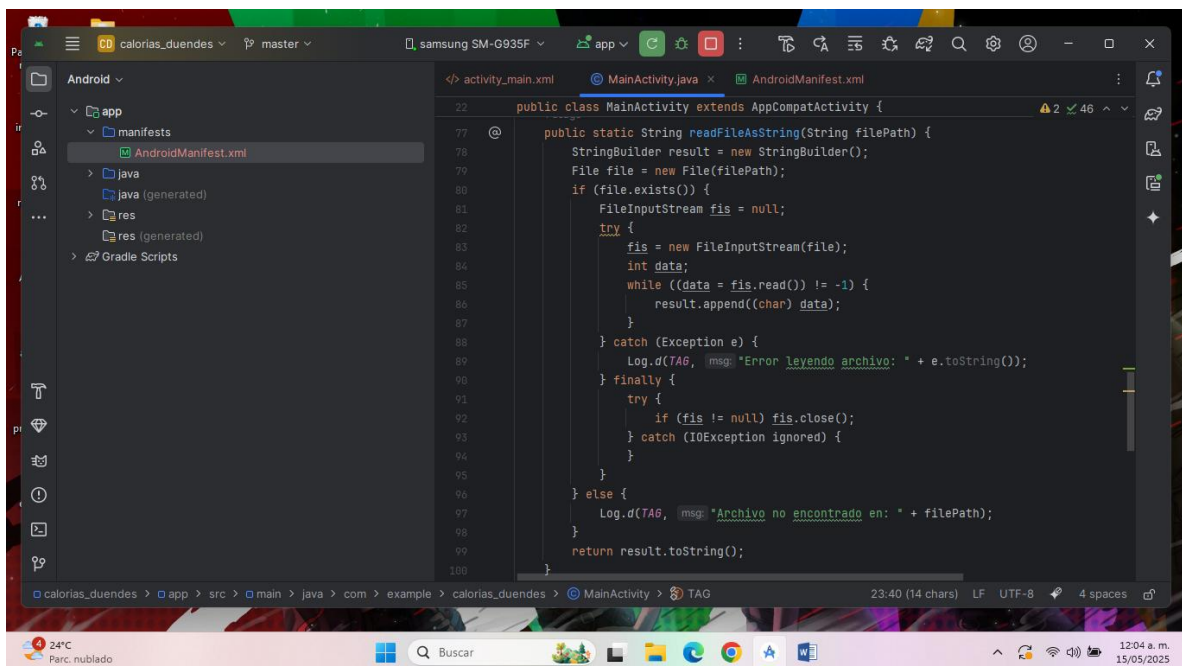


Aquí podemos ver como pide el permiso para leer el almacenamiento del dispositivo



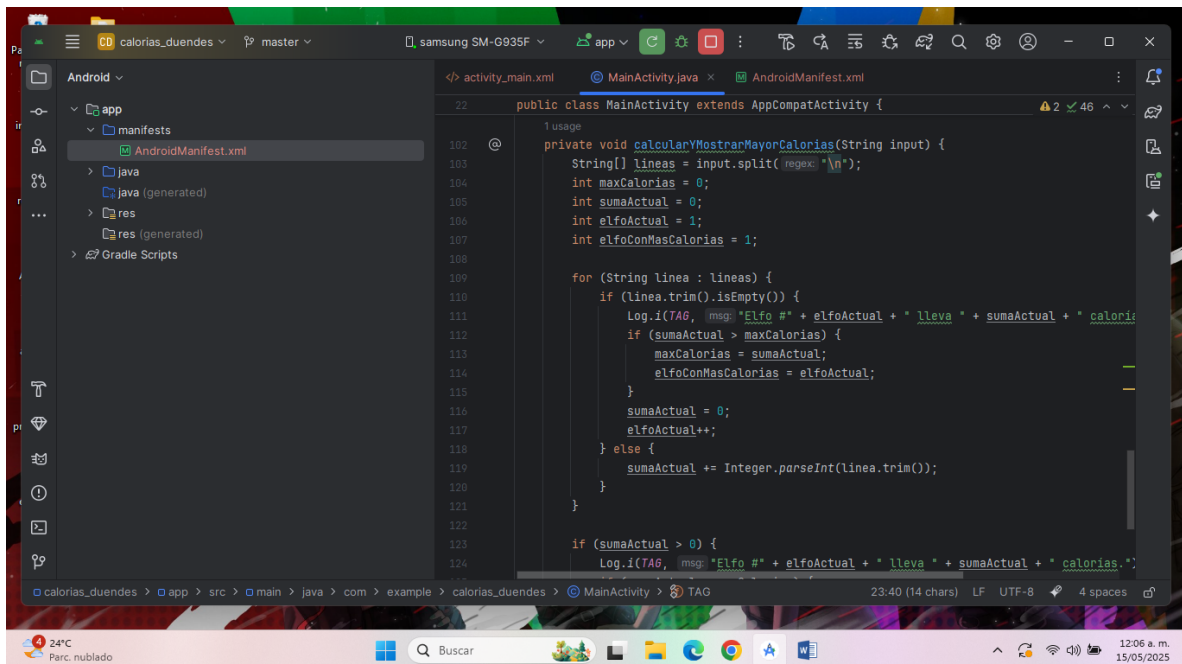
```
22 public class MainActivity extends AppCompatActivity {
58     public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
59                                           Log.e(TAG, msg: "Permiso denegado para leer almacenamiento externo");
60     }
61 }
62
63 private void leerArchivoProcesar() {
64     // Ruta a la carpeta Documents
65     File documentsDir = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOCUMENTS);
66     File archivo = new File(documentsDir, "input_datos_duendes.txt");
67
68     String contenido = readFileAsString(archivo.getAbsolutePath());
69
70     if (!contenido.isEmpty()) {
71         calcularYMostrarMayorCalorias(contenido);
72     } else {
73         Log.e(TAG, msg: "El archivo está vacío o no se pudo leer.");
74     }
75 }
76
77 // Método para leer archivo como string
78 1 usage
79 public static String readFileAsString(String filePath) {
80     StringBuilder result = new StringBuilder();
81     try {
82         FileInputStream fis = new FileInputStream(filePath);
83         int data;
84         while ((data = fis.read()) != -1) {
85             result.append((char) data);
86         }
87     } catch (Exception e) {
88         Log.d(TAG, msg: "Error leyendo archivo: " + e.toString());
89     } finally {
90         try {
91             if (fis != null) fis.close();
92         } catch (IOException ignored) {}
93     }
94 }
95
96 } else {
97     Log.d(TAG, msg: "Archivo no encontrado en: " + filePath);
98 }
99
100 return result.toString();
101 }
```

Aquí se ve la dirección del archivo que vamos a utilizar, y en caso de que el archivo este vacío o no se pueda leer nos mostrara un mensaje de error



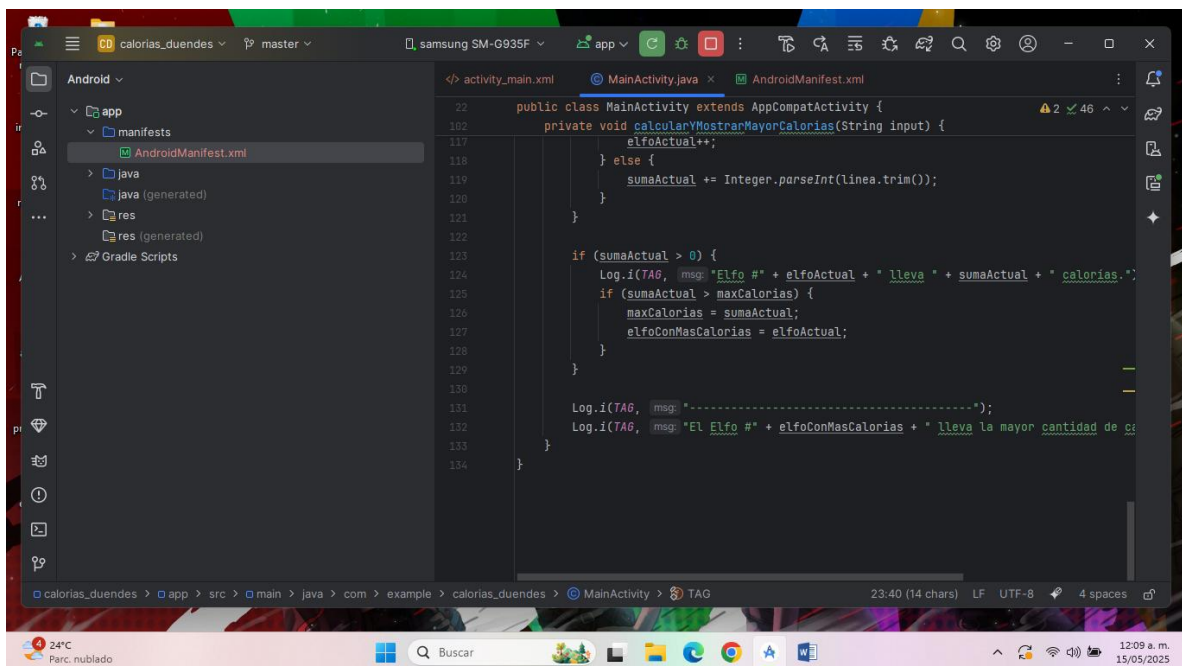
```
22 public class MainActivity extends AppCompatActivity {
77
78     public static String readFileAsString(String filePath) {
79         StringBuilder result = new StringBuilder();
80         File file = new File(filePath);
81         if (file.exists()) {
82             FileInputStream fis = null;
83             try {
84                 fis = new FileInputStream(file);
85                 int data;
86                 while ((data = fis.read()) != -1) {
87                     result.append((char) data);
88                 }
89             } catch (Exception e) {
90                 Log.d(TAG, msg: "Error leyendo archivo: " + e.toString());
91             } finally {
92                 try {
93                     if (fis != null) fis.close();
94                 } catch (IOException ignored) {}
95             }
96         } else {
97             Log.d(TAG, msg: "Archivo no encontrado en: " + filePath);
98         }
99
100         return result.toString();
101     }
102 }
```

Esta parte nos muestra que el archivo esta siendo leído, y del mismo modo nos mandara mensajes de error en caso de que el archivo no exista o no se pueda leer



```
22 public class MainActivity extends AppCompatActivity {
102     1 usage
103     private void calcularYMostrarMayorCalorias(String input) {
104         String[] lineas = input.split(regex: "\\n");
105         int maxCalorias = 0;
106         int sumaActual = 0;
107         int elfoActual = 1;
108         int elfoConMasCalorias = 1;
109
110         for (String linea : lineas) {
111             if (linea.trim().isEmpty()) {
112                 Log.i(TAG, msg: "Elfo #" + elfoActual + " lleva " + sumaActual + " calorías.");
113                 if (sumaActual > maxCalorias) {
114                     maxCalorias = sumaActual;
115                     elfoConMasCalorias = elfoActual;
116                 }
117                 sumaActual = 0;
118                 elfoActual++;
119             } else {
120                 sumaActual += Integer.parseInt(linea.trim());
121             }
122         }
123
124         if (sumaActual > 0) {
125             Log.i(TAG, msg: "Elfo #" + elfoActual + " lleva " + sumaActual + " calorías.");
126         }
127     }
128 }
129
130
131
132
133
134
```

aquí se puede ver que es lo que se realizara una vez que se lea el archivo, lo cual es, leerá línea por línea y en caso de encontrar una línea vacía tomara todos los datos antes de esa línea, los sumara y asignara el resultado a un duenda, indicando que ese es el total de calorías que ese duende está cargando, después segura leyendo el archivo y repetirá el proceso hasta que el archivo se termine.



```
124         if (sumaActual > 0) {
125             Log.i(TAG, msg: "Elfo #" + elfoActual + " lleva " + sumaActual + " calorías.");
126             if (sumaActual > maxCalorias) {
127                 maxCalorias = sumaActual;
128                 elfoConMasCalorias = elfoActual;
129             }
130         }
131
132         Log.i(TAG, msg: "-----");
133         Log.i(TAG, msg: "El Elfo #" + elfoConMasCalorias + " lleva la mayor cantidad de calorías.");
134     }
135 }
136
```

Por ultimo nos mostrara una lista en el log cat con el total de duendes y la cantidad de calorías que cada uno lleva y nos respodera la pregunta del punto 2


```
2025-05-15 00:01:34.646 19644-19644 ConteoCalorias com.example.calorias_duendes I Elfo #238 lleva 49450 calorías.
2025-05-15 00:01:34.646 19644-19644 ConteoCalorias com.example.calorias_duendes I Elfo #239 lleva 57870 calorías.
2025-05-15 00:01:34.646 19644-19644 ConteoCalorias com.example.calorias_duendes I Elfo #240 lleva 48314 calorías.
2025-05-15 00:01:34.646 19644-19644 ConteoCalorias com.example.calorias_duendes I Elfo #241 lleva 45664 calorías.
2025-05-15 00:01:34.646 19644-19644 ConteoCalorias com.example.calorias_duendes I Elfo #242 lleva 47993 calorías.
2025-05-15 00:01:34.646 19644-19644 ConteoCalorias com.example.calorias_duendes I Elfo #243 lleva 44725 calorías.
2025-05-15 00:01:34.646 19644-19644 ConteoCalorias com.example.calorias_duendes I Elfo #244 lleva 44270 calorías.
2025-05-15 00:01:34.647 19644-19644 ConteoCalorias com.example.calorias_duendes I Elfo #245 lleva 50483 calorías.
2025-05-15 00:01:34.647 19644-19644 ConteoCalorias com.example.calorias_duendes I Elfo #246 lleva 45775 calorías.
2025-05-15 00:01:34.647 19644-19644 ConteoCalorias com.example.calorias_duendes I Elfo #247 lleva 43034 calorías.
2025-05-15 00:01:34.647 19644-19644 ConteoCalorias com.example.calorias_duendes I Elfo #248 lleva 29041 calorías.
2025-05-15 00:01:34.647 19644-19644 ConteoCalorias com.example.calorias_duendes I Elfo #249 lleva 35165 calorías.
2025-05-15 00:01:34.647 19644-19644 ConteoCalorias com.example.calorias_duendes I Elfo #250 lleva 59527 calorías.
2025-05-15 00:01:34.647 19644-19644 ConteoCalorias com.example.calorias_duendes I Elfo #251 lleva 59462 calorías.
2025-05-15 00:01:34.647 19644-19644 ConteoCalorias com.example.calorias_duendes I Elfo #252 lleva 43684 calorías.
2025-05-15 00:01:34.647 19644-19644 ConteoCalorias com.example.calorias_duendes I Elfo #253 lleva 49306 calorías.
2025-05-15 00:01:34.647 19644-19644 ConteoCalorias com.example.calorias_duendes I Elfo #254 lleva 42287 calorías.
2025-05-15 00:01:34.647 19644-19644 ConteoCalorias com.example.calorias_duendes I Elfo #255 lleva 14518 calorías.
2025-05-15 00:01:34.647 19644-19644 ConteoCalorias com.example.calorias_duendes I -----
2025-05-15 00:01:34.647 19644-19644 ConteoCalorias com.example.calorias_duendes I El Elfo #84 lleva la mayor cantidad de calorías: 7
```

Aquí podemos ver el resultado del logcat

A continuación, coloco el código que se uso en la aplicación

```
package com.example.calorias_duendes;

import android.Manifest;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.os.Environment;
import android.util.Log;

import androidx.annotation.NonNull;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;

public class MainActivity extends AppCompatActivity {
    private static final String TAG = "ConteoCalorias";
    private static final int REQUEST_PERMISSION_READ_STORAGE = 100;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);

        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars =
            insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top,
            systemBars.right, systemBars.bottom);
            return insets;
        });

        // Verificar permiso de lectura
        if (ContextCompat.checkSelfPermission(this,
        Manifest.permission.READ_EXTERNAL_STORAGE)
        != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this,
            new
            String[]{Manifest.permission.READ_EXTERNAL_STORAGE},
            REQUEST_PERMISSION_READ_STORAGE);
        } else {
            leerArchivoYProcesar();
        }
    }

    // Manejo de resultado del permiso
    @Override
```

```

    public void onRequestPermissionsResult(int requestCode, @NonNull
String[] permissions,
                                           @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
        if (requestCode == REQUEST_PERMISSION_READ_STORAGE) {
            if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                leerArchivoYProcesar();
            } else {
                Log.e(TAG, "Permiso denegado para leer almacenamiento
externo");
            }
        }
    }

    private void leerArchivoYProcesar() {
        // Ruta a la carpeta Documents
        File documentsDir =
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOCUM
ENTS);
        File archivo = new File(documentsDir, "input datos duendes.txt");

        String contenido = readFileAsString(archivo.getAbsolutePath());

        if (!contenido.isEmpty()) {
            calcularYMostrarMayorCalorias(contenido);
        } else {
            Log.e(TAG, "El archivo está vacío o no se pudo leer.");
        }
    }

    // Método para leer archivo como string
    public static String readFileAsString(String filePath) {
        StringBuilder result = new StringBuilder();
        File file = new File(filePath);
        if (file.exists()) {
            FileInputStream fis = null;
            try {
                fis = new FileInputStream(file);
                int data;
                while ((data = fis.read()) != -1) {
                    result.append((char) data);
                }
            } catch (Exception e) {
                Log.d(TAG, "Error leyendo archivo: " + e.toString());
            } finally {
                try {
                    if (fis != null) fis.close();
                } catch (IOException ignored) {
                }
            }
        } else {
            Log.d(TAG, "Archivo no encontrado en: " + filePath);
        }
        return result.toString();
    }
}

```

```

private void calcularYMostrarMayorCalorias(String input) {
    String[] lineas = input.split("\n");
    int maxCalorias = 0;
    int sumaActual = 0;
    int elfoActual = 1;
    int elfoConMasCalorias = 1;

    for (String linea : lineas) {
        if (linea.trim().isEmpty()) {
            Log.i(TAG, "Elfo #" + elfoActual + " lleva " + sumaActual
+ " calorías.");
            if (sumaActual > maxCalorias) {
                maxCalorias = sumaActual;
                elfoConMasCalorias = elfoActual;
            }
            sumaActual = 0;
            elfoActual++;
        } else {
            sumaActual += Integer.parseInt(linea.trim());
        }
    }

    if (sumaActual > 0) {
        Log.i(TAG, "Elfo #" + elfoActual + " lleva " + sumaActual + "
calorías.");
        if (sumaActual > maxCalorias) {
            maxCalorias = sumaActual;
            elfoConMasCalorias = elfoActual;
        }
    }

    Log.i(TAG, "-----");
    Log.i(TAG, "El Elfo #" + elfoConMasCalorias + " lleva la mayor
cantidad de calorías: " + maxCalorias);
    }
}

```