Link: https://github.com/kimchi9199/Mobile_Practice/tree/Lab04?search=1
Question:

Shared preferences: In Android app development, SharedPreferences is a simple key-value data storage mechanism that allows you to store and retrieve data in a persistent manner. It is a lightweight way to store simple data types like booleans, floats, integers, and strings that can be easily accessed from different parts of the app. The data is stored in an XML file in the app's private storage space, and can be accessed even after the app is closed and reopened.

The SharedPreferences API provides an easy way to store configuration information, user preferences, and other small amounts of data. For example, you can use SharedPreferences to store a user's login status, language preference, sound settings, or any other settings that need to be remembered across app launches.

Database:

In Android app development, a database is a structured collection of data that is organized in a way that allows efficient retrieval and manipulation of data. A database is typically used to store and manage large amounts of data, such as user profiles, messages, images, videos, and other types of data that are too complex or too large to be stored in SharedPreferences.

Android provides several ways to work with databases, including SQLite, which is a lightweight, open-source, and built-in relational database engine that provides a fast and efficient way to store and retrieve data. SQLite databases are stored as files in the app's private storage space, and can be accessed using the Android APIs for database management.

To use a database in an Android app, you typically need to create a subclass of the SQLiteOpenHelper class, which provides methods for creating, upgrading, and managing the database schema. You then use this helper class to create and access the database, and use SQL commands to perform database operations such as inserting, updating, and deleting data.

File Storage

File storage is a way to store data as files on the device's file system. In Android app development, there are two types of file storage:

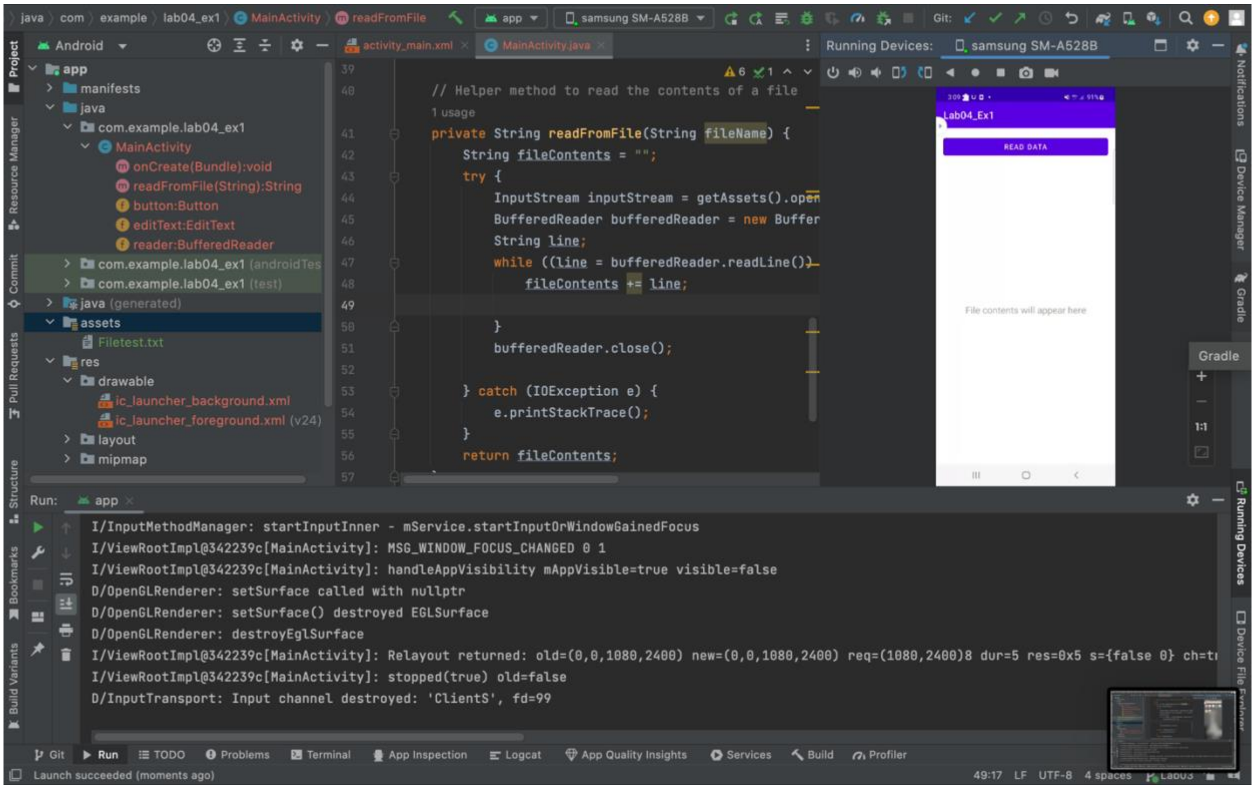
Internal storage: Internal storage is a private storage space that is only accessible to the app that created it. This storage space is not visible to other apps or the user, and is not accessible when the app is uninstalled. Internal storage is useful for storing private data that should not be accessible to other apps or the user, such as user preferences, cache files, or app-specific files.

File storage is a way to store data as files on the device's file system. In Android app development, there are two types of file storage:

1. Internal storage: Internal storage is a private storage space that is only accessible to the app that created it. This storage space is not visible to other apps or the user, and is not accessible when the app is uninstalled. Internal storage is useful for storing private data that should not be accessible to other apps or the user, such as user preferences, cache files, or app-specific files.
2. External storage: External storage is a public storage space that can be accessed by any app on the device, as well as the user through a file manager app. External storage is typically used for storing media files, such as images, videos, or audio files, that the user can access and share across apps.

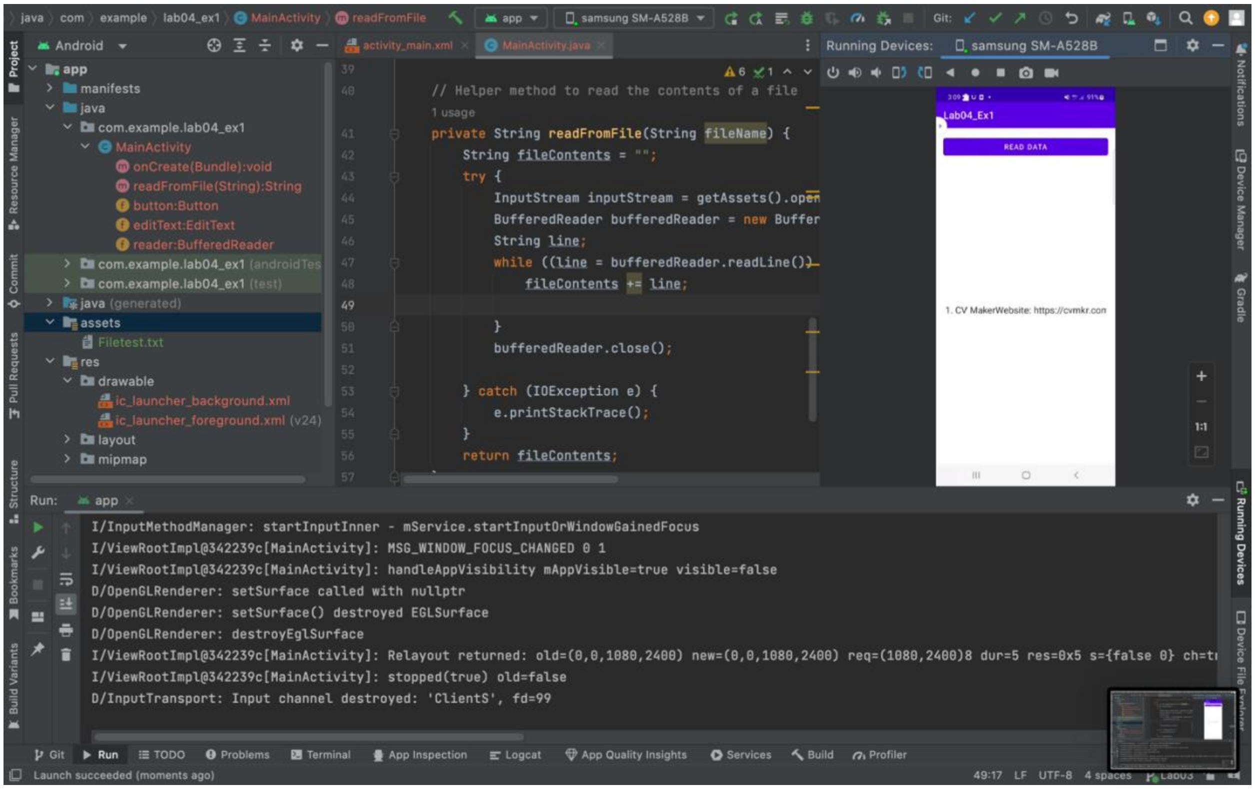
Overall, file storage is a flexible way to store and manage data in an Android app. By using internal storage, you can keep private data secure and out of reach of other apps and the user, while external storage allows you to store media files that can be accessed and shared across apps.

Exercise 1:



```
    // Helper method to read the contents of a file
    1 usage
    private String readFromFile(String fileName) {
        String fileContents = "";
        try {
            InputStream inputStream = getAssets().open
            BufferedReader bufferedReader = new Buffer
            String line;
            while ((line = bufferedReader.readLine())
                fileContents += line;

            }
            bufferedReader.close();

        } catch (IOException e) {
            e.printStackTrace();
        }
        return fileContents;
```

```
I/InputMethodManager: startInputInner - mService.startInputOrWindowGainedFocus
I/ViewRootImpl@342239c[MainActivity]: MSG_WINDOW_FOCUS_CHANGED 0 1
I/ViewRootImpl@342239c[MainActivity]: handleAppVisibility mAppVisible=true visible=false
D/OpenGLRenderer: setSurface called with nullptr
D/OpenGLRenderer: setSurface() destroyed EGLSurface
D/OpenGLRenderer: destroyEglSurface
I/ViewRootImpl@342239c[MainActivity]: Relayout returned: old=(0,0,1080,2400) new=(0,0,1080,2400) req=(1080,2400)8 dur=5 res=0x5 s={false 0} ch=tr
I/ViewRootImpl@342239c[MainActivity]: stopped(true) old=false
D/InputTransport: Input channel destroyed: 'ClientS', fd=99
```



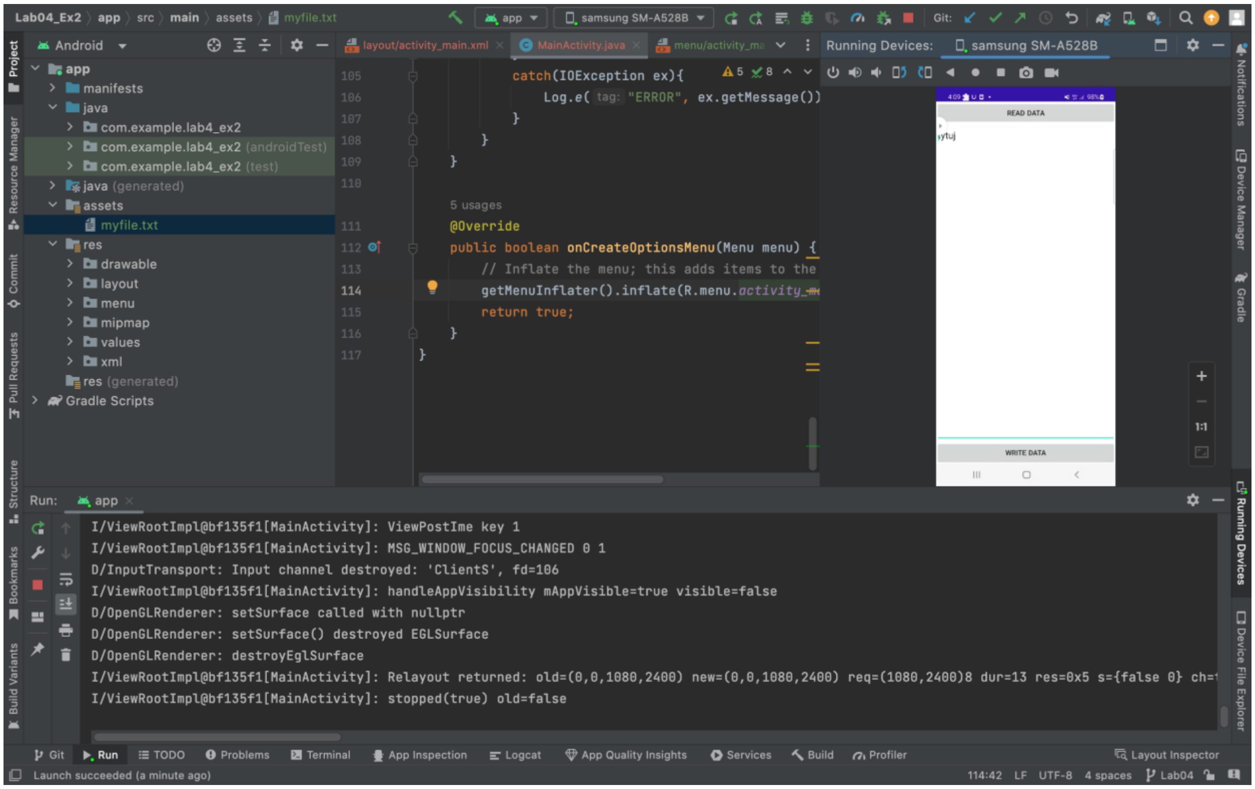
1. CV MakerWebsite: https://cvmkr.con

```
    // Helper method to read the contents of a file
    1 usage
    private String readFromFile(String fileName) {
        String fileContents = "";
        try {
            InputStream inputStream = getAssets().open
            BufferedReader bufferedReader = new Buffer
            String line;
            while ((line = bufferedReader.readLine())
                fileContents += line;

            }
            bufferedReader.close();

        } catch (IOException e) {
            e.printStackTrace();
        }
        return fileContents;
```

```
I/InputMethodManager: startInputInner - mService.startInputOrWindowGainedFocus
I/ViewRootImpl@342239c[MainActivity]: MSG_WINDOW_FOCUS_CHANGED 0 1
I/ViewRootImpl@342239c[MainActivity]: handleAppVisibility mAppVisible=true visible=false
D/OpenGLRenderer: setSurface called with nullptr
D/OpenGLRenderer: setSurface() destroyed EGLSurface
D/OpenGLRenderer: destroyEglSurface
I/ViewRootImpl@342239c[MainActivity]: Relayout returned: old=(0,0,1080,2400) new=(0,0,1080,2400) req=(1080,2400)8 dur=5 res=0x5 s={false 0} ch=tr
I/ViewRootImpl@342239c[MainActivity]: stopped(true) old=false
D/InputTransport: Input channel destroyed: 'ClientS', fd=99
```

Ex2:

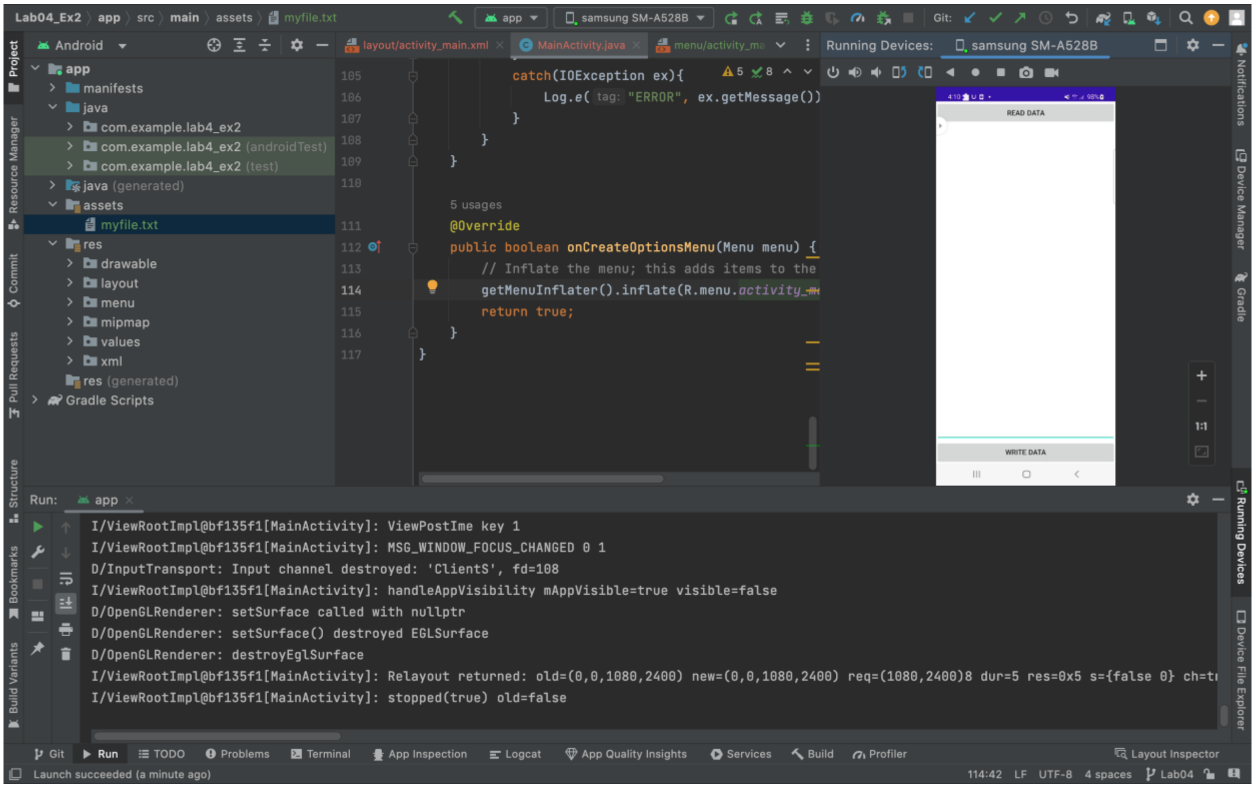
Write Data



```java
            catch(IOException ex){
                Log.e( tag: "ERROR", ex.getMessage())
            }
        }
    }

    5 usages
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the
        getMenuInflater().inflate(R.menu.activity_m
        return true;
    }
}
```

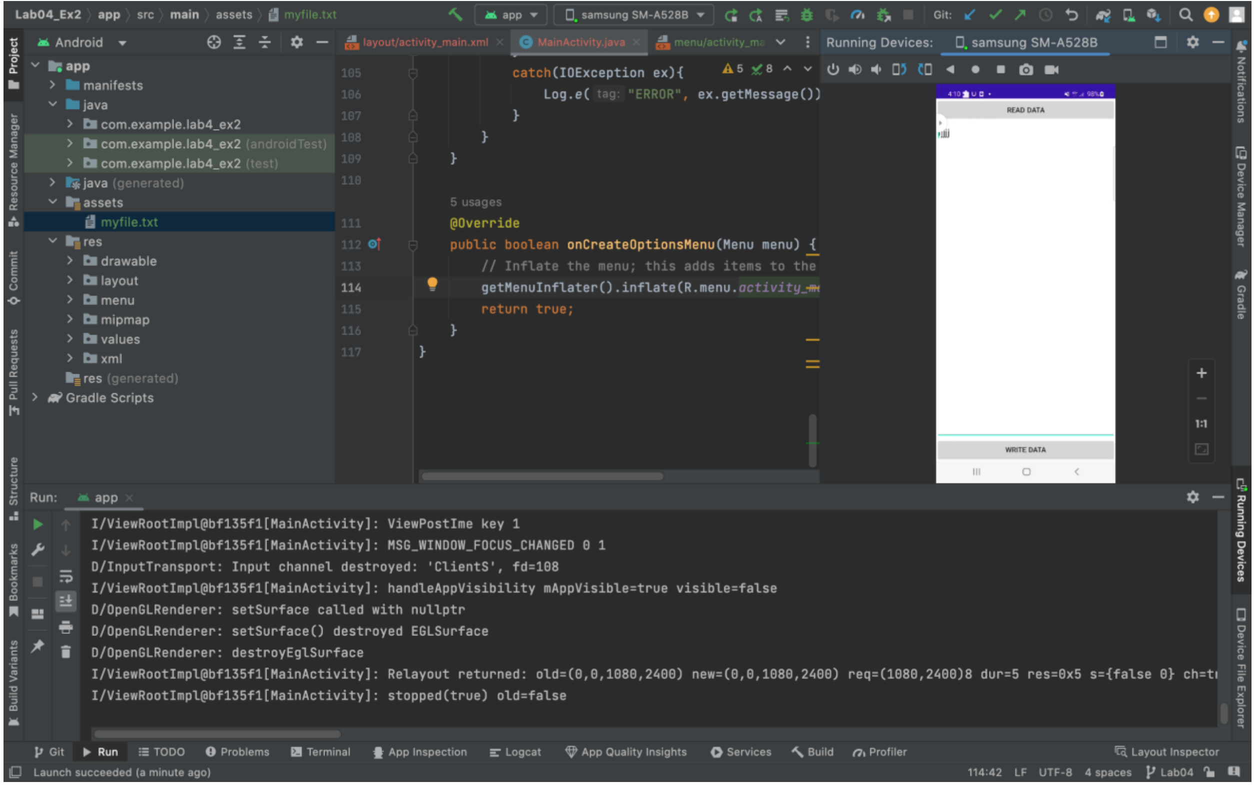
```
I/ViewRootImpl@bf135f1[MainActivity]: ViewPostIme key 1
I/ViewRootImpl@bf135f1[MainActivity]: MSG_WINDOW_FOCUS_CHANGED 0 1
D/InputTransport: Input channel destroyed: 'ClientS', fd=106
I/ViewRootImpl@bf135f1[MainActivity]: handleAppVisibility mAppVisible=true visible=false
D/OpenGLRenderer: setSurface called with nullptr
D/OpenGLRenderer: setSurface() destroyed EGLSurface
D/OpenGLRenderer: destroyEglSurface
I/ViewRootImpl@bf135f1[MainActivity]: Relayout returned: old=(0,0,1080,2400) new=(0,0,1080,2400) req=(1080,2400)8 dur=13 res=0x5 s={false 0} ch=
I/ViewRootImpl@bf135f1[MainActivity]: stopped(true) old=false
```

Read Data:



```
        catch(IOException ex){
            Log.e( tag: "ERROR", ex.getMessage())
        }
    }
}

5 usages
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the
    getMenuInflater().inflate(R.menu.activity_m
    return true;
}
}
```

```
I/ViewRootImpl@bf135f1[MainActivity]: ViewPostIme key 1
I/ViewRootImpl@bf135f1[MainActivity]: MSG_WINDOW_FOCUS_CHANGED 0 1
D/InputTransport: Input channel destroyed: 'ClientS', fd=108
I/ViewRootImpl@bf135f1[MainActivity]: handleAppVisibility mAppVisible=true visible=false
D/OpenGLRenderer: setSurface called with nullptr
D/OpenGLRenderer: setSurface() destroyed EGLSurface
D/OpenGLRenderer: destroyEglSurface
I/ViewRootImpl@bf135f1[MainActivity]: Relayout returned: old=(0,0,1080,2400) new=(0,0,1080,2400) req=(1080,2400)8 dur=5 res=0x5 s={false 0} ch=t
I/ViewRootImpl@bf135f1[MainActivity]: stopped(true) old=false
```



```
        catch(IOException ex){
            Log.e( tag: "ERROR", ex.getMessage())
        }
    }
}

5 usages
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the
    getMenuInflater().inflate(R.menu.activity_m
    return true;
}
}
```

```
I/ViewRootImpl@bf135f1[MainActivity]: ViewPostIme key 1
I/ViewRootImpl@bf135f1[MainActivity]: MSG_WINDOW_FOCUS_CHANGED 0 1
D/InputTransport: Input channel destroyed: 'ClientS', fd=108
I/ViewRootImpl@bf135f1[MainActivity]: handleAppVisibility mAppVisible=true visible=false
D/OpenGLRenderer: setSurface called with nullptr
D/OpenGLRenderer: setSurface() destroyed EGLSurface
D/OpenGLRenderer: destroyEglSurface
I/ViewRootImpl@bf135f1[MainActivity]: Relayout returned: old=(0,0,1080,2400) new=(0,0,1080,2400) req=(1080,2400)8 dur=5 res=0x5 s={false 0} ch=t
I/ViewRootImpl@bf135f1[MainActivity]: stopped(true) old=false
```