BCS 371 Mobile Application Development I

Arthur Hoskey, Ph.D.
Farmingdale State College
Computer Systems Department

- File IO
- Device Explorer (send files to/from the emulator)

Today's Lecture

Read/Write to Files

- Use Kotlin I/O classes to read/write data to files.
- Files are saved on the device.
- For example…

Read/Write Data to a File

Open File

- Here is code to open files for input and output:

```
val FILE_NAME = "data.txt"

// Output Setup Code
val fos = openFileOutput(FILE_NAME, Context.MODE_PRIVATE)
val out = PrintStream(fos)
// Code to write to PrintStream goes here…

// Input Setup Code
val fis = openFileInput(FILE_NAME)
val scanner = Scanner(fis)
// Code to read from scanner goes here…
```

Open file for output that is private to the app (in app sandbox)

Connect file output stream to a PrintStream

Open file for input (will look in the private app directory)

Connect file input stream to a Scanner

Note: The method openFileOutput needs a context to compile correctly. For example, if you call openFileOutput inside of an anonymous listener it will require getting the context:

```
applicationContext.openFileOutput(FILE_NAME, Context.MODE_PRIVATE)
```

Open File

Read Data from a File Using a Scanner

- Use a FileInputStream and Scanner to read data from an input file (the Scanner class is similar to Java's Scanner class).

```
val FILE_NAME = "data.txt"

// Open file input stream.
val fis = openFileInput(FILE_NAME)

// Connecte file input stream to a Scanner
val scanner = Scanner(fis)

var line = ""
while ( scanner.hasNext() ) {
    line = scanner.nextLine()
    println(line)
}
```

hasNext returns true if there is more data to read. It will loop until it reaches the end of the file.

Read one line of data (as a string) from the file.

Read Data from a File Using a Scanner

Write Data to a File Using a PrintStream

- Use FileOutputStream and PrintStream to write data fo a file (PrintStream is similar to Java's PrintStream class).

```
try {
    val FILE_NAME = "data.txt"

    // Create a new output file stream that's private to this app
    val fos = openFileOutput(FILE_NAME, MODE_PRIVATE)

    // Create the PrintStream
    val out = PrintStream(fos)

    // Write data to the file
    var i = 100
    out.printf("i = %d\n", i)
} catch (e: FileNotFoundException) {
}
```

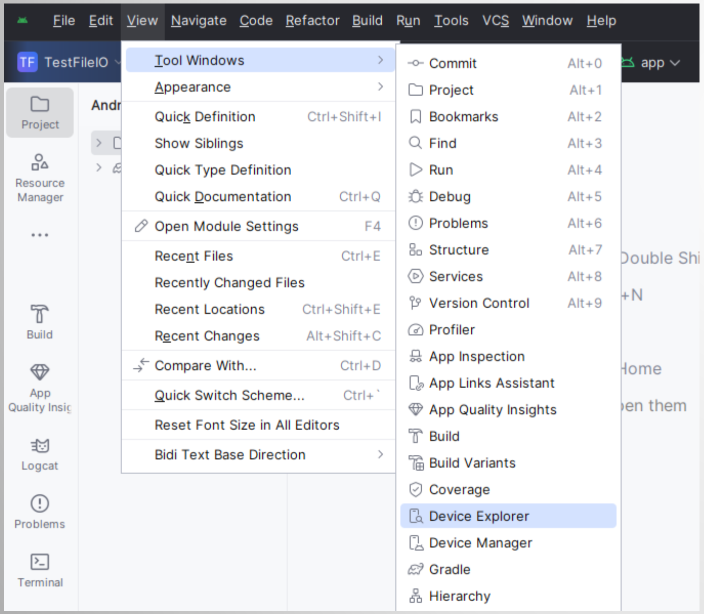
Format specifiers in the printf are similar to Java.

This will write i = 100 to the file.

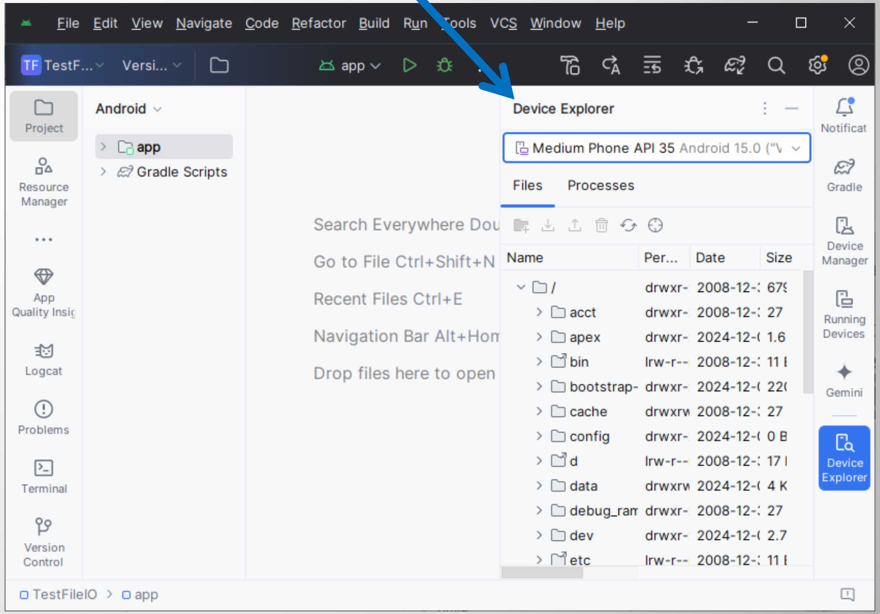
Write Data to a File Using a PrintStream

- Device File Explorer allows you to get/send files between the emulator and computer. You can also create files on the emulator.
- Go to: View|Tool Windows|Device Explorer.
- A window containing Device Explorer will open on the right side.

View | Tool Windows | Device Explorer



Device Explorer Window



Name	Per...	Date	Size
/	drwxr-	2008-12-:	67(
acct	drwxr-	2008-12-:	27
apex	drwxr-	2024-12-(	1.6
bin	lrw-r--	2008-12-:	11 I
bootstrap-	drwxr-	2024-12-(	22(
cache	drwxrw	2008-12-:	27
config	drwxr-	2024-12-(	0 B
d	lrw-r--	2008-12-:	17 I
data	drwxrw	2024-12-(	4 K
debug_ran	drwxr-	2008-12-:	27
dev	drwxr-	2024-12-(	2.7
etc	lrw-r--	2008-12-:	11 I

Device Explorer

Copying Files To/From the Emulator

- Use the **Device Explorer** to do this.

- **Save File to Computer (from Emulator).** Right-click file in Device Explorer to get a context menu. There will be a Save As… menu option.
- **Send File to Emulator (from Computer).** Right-click directory in Device Explorer and you will get a context menu. There will be an Upload… menu option.

- Files are stored in in the following directory:

/data/data/<your package name>/files

Copy File from AVD Private Area to Computer

- **Default App-Specific File Directory**. Inside Device Explorer navigate to the following directory:

/data/data/<your package name>/files

Note: This directory will not appear in Device Explorer until you run your app the first time (running the first time will install your app on the emulator and create this directory for you). If you ran your app and you do not see this directory click the synchronize icon (double arrow icon) in the Device Explorer toolbar.

- **Create New File (in Emulator)**. Right click the directory and a context menu will appear (use the files directory detailed above). Choose New|File from the context menu.
- **Open/Edit New File (in Emulator)**. Double click the file in the emulator to open it (the file will open in a tab in Android Studio). When you open it the Clear Read-Only Status dialog will appear. Click OK in this dialog and you will be able to edit the file. The file being edited is now on the computer. You to send this file back to the emulator (as detailed in the previous slide) to see the edits in the emulator. Hover on the file tab in Android Studio to get the full location of the file (helps when sending file to emulator).

Create/Edit New File in Emulator

- End of Slides

End of Slides