

# CS378 - Mobile Computing

## Persistence

# Saving State

- We have already seen saving app state into a Bundle on orientation changes or when an app is killed to reclaim resources but may be recreated later

```
@Override
protected void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);

    Log.d(TAG, "in onSaveInstanceState");

    outState.putCharArray("board", mGame.getBoardState());
    outState.putBoolean("mGameOver", mGameOver);
    outState.putCharSequence("info", mInfoTextView.getText());
    outState.putChar("mTurn", mTurn);
    outState.putChar("mGoesFirst", mGoesFirst);
}
```

# Storing Data

- Multiple options for storing data associated with apps
- Shared Preferences
- Internal Storage
  - device memory
- External Storage
- SQLite Database
- Network Connection

# Sharing Data

- Private data can be shared by creating a Content Provider
- Android has many built in Content Providers for things such as
  - audio (random song from last time)
  - images
  - video
  - contact information

# Shared Preferences

- Private primitive data stored in key-value pairs
- SharedPreferences Class
- Store and retrieve key-value pairs of data
  - keys are Strings
  - values are Strings, Sets of Strings, boolean, float, int, or long
- Not strictly for *preferences*

# Using SharedPreferences

- Obtain a SharedPreferences object for application using these methods:
  - getSharedPreferences(String name, int mode)
    - if you want multiple files
  - getPreferences(int mode)

```
// restore the scores and difficulty
SharedPreferences mPrefs = getSharedPreferences("ttt_prefs", MODE_PRIVATE);
mHumanWins = mPrefs.getInt("mHumanWins", 0);
mComputerWins = mPrefs.getInt("mComputerWins", 0);
mTies = mPrefs.getInt("mTies", 0);
mGame.setDifficultyLevel(TicTacToeGame.DifficultyLevel.values()[mPrefs.getI
```

# SharedPreferences Modes

- File creation modes
- Constants from the Context class
  - Activity is a descendant of Context
- `MODE_PRIVATE`
  - accessed only by calling application
- `MODE_WORLD_READABLE`
  - other applications have read access
- `MODE_WORLD_WRITEABLE`
  - other applications have write access
- `MODE_MULTI_PROCESS`
  - file on disk checked for modification even if shared preferences instance loaded. (Multiple threads using the same file)

# Writing to SharedPreferences

- After obtaining SharedPreferences object:
  - call `edit()` method on object to get a `SharedPreferences.Editor` object
  - place data by calling `put` methods on the `SharedPreferences.Editor` object
  - also possible to clear all data or remove a particular key



# Writing to SharedPreferences

- When done writing data via the editor call either `apply()` or `commit()`
- `apply()` is the simpler method
  - used when only one process expected to write to the preferences object
- `commit()` returns a boolean if write was successful
  - for when multiple process may be writing to preferences

# Reading From Shared Preferences

- After obtaining SharedPreferences object use various get methods to retrieve data
- Provide key (string) and default value if key is not present
- get Boolean, Float, Int, Long, String, StringSet
- getAll() returns Map<String, ?> with all of the key/value pairs in the preferences

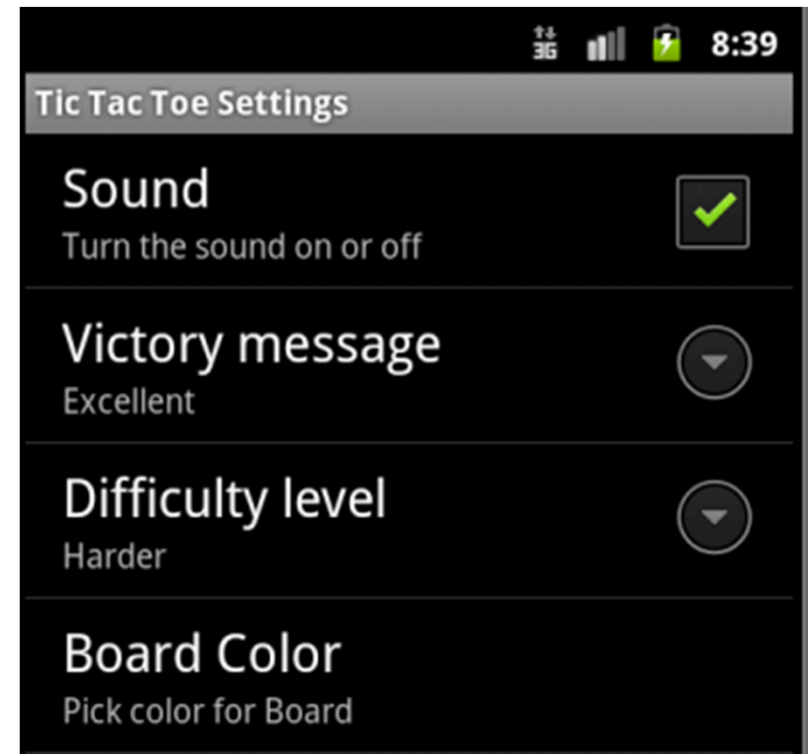
# Shared Preferences File

- Stored as XML

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<string name="victory_message">Excellent</string>
<int name="board_color" value="-65528" />
<int name="mTies" value="6" />
<string name="difficulty_level">Harder</string>
<int name="mComputerWins" value="1" />
<int name="mDifficulty" value="1" />
<int name="mHumanWins" value="9" />
</map>
```

# Preference Activity

- An Activity framework to allow user to select and set preferences for your app
- tutorial 6 has an example
  - difficulty, sound, color, victory message
- Main Activity can start a preference activity to allow user to set preferences



# Internal Storage

- Private data stored on device memory
- More like traditional file i/o
- by default files are private to your application
  - other apps cannot access
- files removed when app is uninstalled

# Internal Storage

- To create and write a private file to the device internal storage:
- call `openFileOutput(String name, int mode)`
  - method from `Context`
  - file created if does not already exist
  - returns `FileOutputStream` object (regular Java class)
- Modes same as `SharedPreferences` minus `MODE_MULTI_PROCESS` and addition of `MODE_APPEND`

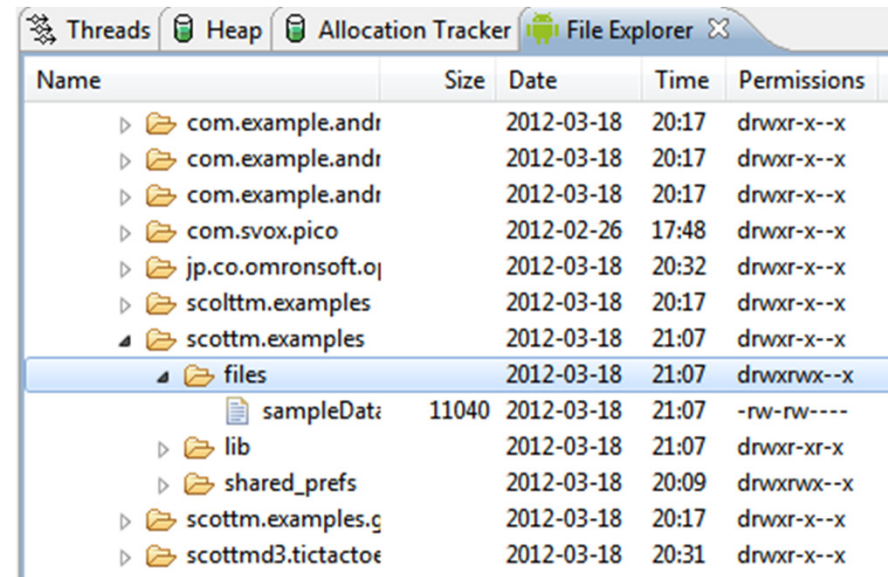
# Writing to Files

- FileOutputStream writes raw bytes
  - arrays of bytes or single bytes
- Much easier to wrap the FileOutputStream in PrintStream object

```
public void writeFile(View v) {  
    try {  
        FileOutputStream fos  
            = openFileOutput("sampleData", MODE_PRIVATE);  
        PrintStream writer = new PrintStream(fos);  
        Random r = new Random();  
        for(int i = 0; i < 1000; i++) {  
            writer.println(r.nextInt());  
        }  
        writer.close();  
    }  
    catch(FileNotFoundException e) {  
        Log.d(TAG, "Exception trying to open file: " + e);  
    }  
}
```

# Reading from Files

- files saved to device
  - data directory for app
- call `openFileInput(String name)` method to obtain a `FileInputStream`
- `FileInputStream` reads bytes
  - for convenience may connect to `Scanner` object or wrap in a `DataInputStream` object



| Name               | Size  | Date       | Time  | Permissions |
|--------------------|-------|------------|-------|-------------|
| com.example.andr   |       | 2012-03-18 | 20:17 | drwxr-x--x  |
| com.example.andr   |       | 2012-03-18 | 20:17 | drwxr-x--x  |
| com.example.andr   |       | 2012-03-18 | 20:17 | drwxr-x--x  |
| com.svox.pico      |       | 2012-02-26 | 17:48 | drwxr-x--x  |
| jp.co.omronsoft.oj |       | 2012-03-18 | 20:32 | drwxr-x--x  |
| scolttm.examples   |       | 2012-03-18 | 20:17 | drwxr-x--x  |
| scolttm.examples   |       | 2012-03-18 | 21:07 | drwxr-x--x  |
| files              |       | 2012-03-18 | 21:07 | drwxrwx--x  |
| sampleData         | 11040 | 2012-03-18 | 21:07 | -rw-rw----  |
| lib                |       | 2012-03-18 | 21:07 | drwxr-xr-x  |
| shared_prefs       |       | 2012-03-18 | 20:09 | drwxrwx--x  |
| scolttm.examples.g |       | 2012-03-18 | 20:17 | drwxr-x--x  |
| scottd3.tictactoe  |       | 2012-03-18 | 20:31 | drwxr-x--x  |



# Static Files

- If you need / have a file with a lot of data at compile time:
  - save file in project res/raw / directory
  - can open file using the `openRawResource(int id)` method and pass the `R.raw.id` of file
  - returns an `InputStream` to read from file
  - cannot write to the file

# Cache Files

- If need to cache data for application instead of storing persistently:
  - call `getCacheDir()` method to obtain a `File` object that is a directory where you can create and save temporary cache files
  - files may be deleted by Android later if space needed but you should clean them up on your own
  - recommended to keep under 1 MB

# External Files - Other Useful Methods

- All of these are inherited from Context
- File getFileDir()
  - get absolute path to filesystem directory when app files are saved
- File getDir(String name, int mode)
  - get and create if necessary a directory for files
- boolean deleteFile(String name)
  - get rid of files, especially cache files
- String[] fileList()
  - get an array of Strings with files associated with Context (application)

# External Storage

- Public data stored on shared external storage
- may be SD (Secure Digital) card on non removable
- files saved to external storage are world-readable
- files may be modified by user when they enable USB mass storage for device

# Checking Media Availability

- Call `Environment.getExternalStorageState()` method to determine if media available
  - may be mounted to computer, missing, read-only or in some other state that prevents accessing

# Checking Media State

```
boolean mExternalStorageAvailable = false;
boolean mExternalStorageWriteable = false;
String state = Environment.getExternalStorageState();

if (Environment.MEDIA_MOUNTED.equals(state)) {
    // We can read and write the media
    mExternalStorageAvailable = mExternalStorageWriteable = true;
} else if (Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
    // We can only read the media
    mExternalStorageAvailable = true;
    mExternalStorageWriteable = false;
} else {
    // Something else is wrong. It may be one of many other states,
    // to know is we can neither read nor write
    mExternalStorageAvailable = mExternalStorageWriteable = false;
}
```

- other states such as media being shared, missing, and others

# Accessing Files on External Storage

- call `getExternalFilesDir(String type)` to obtain a directory (File object) to get directory to save files
- type is String constant from Environment class
  - `DIRECTORY_ALARMS`, `DIRECTORY_DCIM` (Digital Camera Images),  
`DIRECTORY_DOWNLOADS`,  
`DIRECTORY_MOVIES`, `DIRECTORY_MUSIC`,  
`DIRECTORY_NOTIFICATIONS`,  
`DIRECTORY_PICTURES`, `DIRECTORY_PODCASTS`,  
`DIRECTORY_RINGTONES`

# External File Directory

- If not a media file then send **null** as parameter to `getExternalFilesDir()` method
- The `DIRECTORY_<TYPE>` constants allow Android's Media Scanner to categorize files in the system
- External files associated with application are deleted when application uninstalled



# External Data Shared Files

- If you want to save files to be shared with other apps:
- save the files (audio, images, video, etc.) to one of the public directories on the external storage device
- `Environment.getExternalStoragePublicDirectory(String type)` method returns a `File` object which is directory
- same types as `getExternalFilesDir` method

# Examining Shared Directories

- Not the same as the system media directories

```
private void showDirs() {  
    for(String type : types) {  
        File dir = Environment  
            .getExternalStoragePublicDirectory(type);  
        Log.d(TAG, "type: " + type + ", dir: " + dir);  
        File[] files = dir.listFiles();  
        if(files != null)  
            for(File f : dir.listFiles())  
                Log.d(TAG, f + "");  
    }  
}
```

# Result

|       |                                                     |
|-------|-----------------------------------------------------|
| PTest | type: Alarms, dir: /mnt/sdcard/Alarms               |
| PTest | type: DCIM, dir: /mnt/sdcard/DCIM                   |
| PTest | /mnt/sdcard/DCIM/.thumbnails                        |
| PTest | /mnt/sdcard/DCIM/100ANDRO                           |
| PTest | type: Download, dir: /mnt/sdcard/Download           |
| PTest | type: Movies, dir: /mnt/sdcard/Movies               |
| PTest | type: Music, dir: /mnt/sdcard/Music                 |
| PTest | /mnt/sdcard/Music/Susan Boyle - Amazing grace.mp3   |
| PTest | /mnt/sdcard/Music/Rem - Losing My Religion.mp3      |
| PTest | type: Notifications, dir: /mnt/sdcard/Notifications |
| PTest | type: Pictures, dir: /mnt/sdcard/Pictures           |
| PTest | type: Podcasts, dir: /mnt/sdcard/Podcasts           |
| PTest | type: Ringtones, dir: /mnt/sdcard/Ringtones         |

# SQLite Database

- Structured data stored in a private database
- More on this next lecture

# Network Connection

- Store data on web with your own network server
- Use wireless or carrier network to store and retrieve data on web based server
- classes from `java.net` and `android.net`