# CPSC475/575
# Persistence

# Saving Data TEMPORARY

- Ephemeral storage.
  - System kills app – view object state saved
  - You kill app- gone forever
- Techniques
  - Widget has ID, system saves state
  - Bundle or Intent (mostly for sending data to new activities or processes)
  - Singleton pattern

# Saving Data PERMANENT

- Shared Preferences- private data stored in key-value pairs

- Internal Storage - private data on the device

- External Storage – public data on the device

- SQLite Database (we will not do)

- Cloud (we will probably not do)

# Shared Preferences - Examples

- See Mainactivity.java in app module of Serialization_preferences

# Shared Preferences

- Similar to a bundle

- SharedPreferences Class

- Store and retrieve key-value pairs of data
  - keys are Strings
  - values are Strings, Sets of Strings, boolean, float, int, or long (like a bundle)

- Can save any data this way as long as its Parcelable (Serializable)

# Writing to SharedPreferences Recipe

- Obtain SharedPreferences object:

- Call edit() method on object to get a SharedPreferences.Editor object

- Insert data by calling put methods on the SharedPreferences.Editor object (Int, Boolean,String char etc)

- Commit changes

# Writing to SharedPreferences

```java
private static final String PREF_FILE_NAME = "PrefFile";
private static final String PASSWORD      = "Password";
private static final String DEFAULT_PWD   = "Default";
```
**Defaults**

```java
public void savePref() {
    //SHAREDPREFERENCES - PERMANENT STORAGE
    // get a handle to "PrefFile", create if necessary, only this
    // process has access can have MODE_WORLD_READABLE and MODE_WORLD_WRITEABLE. :
    SharedPreferences settings = getSharedPreferences("PrefFile",MODE_PRIVATE);
```
**choose file**

```java
    // can only make changes with editor
    SharedPreferences.Editor editor = settings.edit();
```
**must edit()**

```java
    // slap something in it, strings, booleans ints, check the docs
    editor.putString(PASSWORD, "admin");
```
**save values**

```java
    //editor.clear();           //removes everthing
    //editor.remove(PASSWORD);  //dumps key value pair
```
**can clear all or delete one**

```java
    // Commit the edits! You dont call this it aint saved!
    editor.commit();
}
```
**must commit**

# Reading From Shared Preferences recipe

- Provide key (string) and default value if key is not present

- get Boolean, Float, Int, Long, String, StringSet

# Reading from SharedPreferences

```
private static final String PREF_FILE_NAME  = "PrefFile";

private static final String PASSWORD        = "Password";

private static final String DEFAULT_PWD      = "Default";

public void getPref(){

    //SHAREDPREFERENCES - PERMANENT STORAGE

    // Restore preferences

    SharedPreferences settings = getSharedPreferences(PREF_FILE_NAME,MODE_PRIVATE);

    String savedPwd = settings.getString(PASSWORD,  DEFAULT_PWD);

}
```

Defaults

choose file

get value

# Shared Preferences File

- Stored as XML

- Stored on emulated device data/.../<yourpackagename>...

- Can use Device Explorer in AS to access
  (View→ Tool Windows→Device Explorer)

```xml
<?xml version="1.0" encoding="UTF-8" standalone="
<map>
    <string name="Password">admin</string>
</map>
```

# Soon - Preference Activity

- An Activity framework to allow user to select and set preferences for your app
- Main Activity can start a preference activity to allow user to set preferences
- Much like the preferences we have done except calls

    getDefaultSharedPreferences(this)
- Boilerplate professional code
- We will do these after we do Fragments
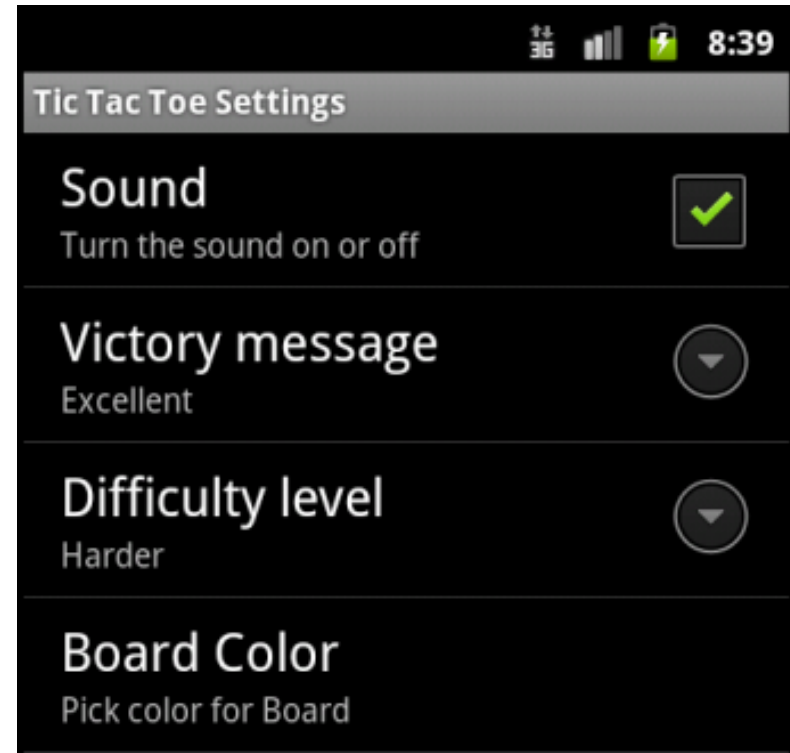
# Internal Storage - Examples

- See 5_Serialization

# 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 – Reading and Writing

- See 5_Serialization KP_fileIO class
- It's just standard Java file I/O

# Files - Useful Methods

- All of these are inherited from Context

- File getFilesDir()
  - 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)

# BTW, application specific 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 this file

# External Storage - Examples

- See 5_Serialization

# External Storage

- Public data stored on shared external storage
  - getExternalFilesDir()

# Checking Media Availability

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

# Checking Media Availability

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

```java
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;
}
```
available

Read

Cannot Use

# External File Directory
# (private to your app)

- Used only by your app (textures, sounds)
- External files associated with application are deleted when application uninstalled

```java
File file = new File(getExternalFilesDir(null), "DemoFile.jpg");
```

If any of the following
DIRECTORY_ALARMS,
DIRECTORY_MUSIC,
DIRECTORY_PICTURES,
Etc
specific subdirectory created

# External Shared Files

caveat see commonsware explanation on course website

- Files shared with other apps

- Use public directories on the external storage device

- **Not** deleted when app uninstalled

- getExternalStoragePublicDirectory( String type)

- Type is DIRECTORY_ALARMS,  DIRECTORY_DCIM (Digital Camera IMages), DIRECTORY_DOWNLOADS, DIRECTORY_MOVIES, DIRECTORY_MUSIC, DIRECTORY_NOTIFICATIONS, DIRECTORY_PICTURES, DIRECTORY_PODCASTS, DIRECTORY_RINGTONES

- System media scanner will categorize your files based on this type

# Summary

- SharedPreferences
- Internal and External Storage