CS371m - Mobile Computing

Persistence
Storing Data

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

```java
@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);
}
```
SHARED PREFERENCES
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
  – So, somewhat limited options
• Not strictly for app preferences
SharedPreferences

- Several levels of preferences:
  - `getPreferences(int mode)` for the Activity's Preferences
    - name based on Activity
  - `getSharedPreferences(String name, int mode)` for an Application's shared preferences
    - multiple activities
- `PreferenceManager.getDefaultSharedPreferences()` for system wide preferences
Using SharedPreferences

• Obtain a SharedPreferences object for application using these methods:
  – getSharedPreferences(String name, int mode)
  – getPreferences(int mode)

```java
// 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.getInt("mDifficulty", 0)]);
```
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
Limited Data Types for SharedPreferences

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>putBoolean(String key, boolean value)</td>
<td>Set a boolean value in the preferences editor, to be written</td>
</tr>
<tr>
<td>putFloat(String key, float value)</td>
<td>Set a float value in the preferences editor, to be written</td>
</tr>
<tr>
<td>putInt(String key, int value)</td>
<td>Set an int value in the preferences editor, to be written</td>
</tr>
<tr>
<td>putLong(String key, long value)</td>
<td>Set a long value in the preferences editor, to be written</td>
</tr>
<tr>
<td>putString(String key, String value)</td>
<td>Set a String value in the preferences editor, to be written</td>
</tr>
<tr>
<td>putStringSet(String key, Set&lt;String&gt; values)</td>
<td>Set a set of String values in the preferences editor, to be written</td>
</tr>
</tbody>
</table>
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
  – blocking operation, so use sparingly or in thread off of the UI thread to avoid ANR
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
<?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
• Current standard is to use a PreferenceFragment instead
INTERNAL STORAGE
Internal Storage

• Private data stored on device memory
  – not part of apk
• More like traditional file i/o
  – in fact not that different from Java I/O
• by default files are private to your application
  – other apps cannot access directly
  – recall content providers to share data with other apps
• 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 inherited from Context
  – file created if does not already exist
  – returns FileOutputStream object
    • regular Java class
• Modes include: MODE_APPEND, MODE_PRIVATE
  deprecated: MODE_WORLD_READABLE, MODE_WORLD_WRITEABLE
Writing to Files

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

```java
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
Static Files

• If you need or have a file with a lot of data at compile time:
  – create and save file in project res/raw/ directory
  – 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, part of the apk
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
Internal Files - Other Useful Methods

• All of these are inherited from Context
• File getFileDir()
  – get absolute path to filesystem directory where 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 FILES
External Storage

• Public data stored on shared external storage
• may be removable SD (Secure Digital) card or internal, non-removable storage
• files saved to external storage are world-readable
• files may be unavailable when device mounts external storage to another system
• files may be modified by user when they enable USB mass storage for device
• request WRITE_EXTERNAL_STORAGE permission in manifest
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

```java
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_ALARM, 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 a directory
  • same types as getExternalFilesDir method
Sharing Data

• Example:
  – In the random art app
  – add a button to save images
  – if we want images to show up with other "images" save to the DIRECTORY_PICTURES directory
  – now, other apps can view / use these images via the media scanner
  – NOT deleted when app deleted
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 + "");
    }
}
<table>
<thead>
<tr>
<th>PTest</th>
<th>type: Alarms, dir: /mnt/sdcard/Alarms</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTest</td>
<td>type: DCIM, dir: /mnt/sdcard/DCIM</td>
</tr>
<tr>
<td>PTest</td>
<td>/mnt/sdcard/DCIM/.thumbnails</td>
</tr>
<tr>
<td>PTest</td>
<td>/mnt/sdcard/DCIM/100ANDRO</td>
</tr>
<tr>
<td>PTest</td>
<td>type: Download, dir: /mnt/sdcard/Download</td>
</tr>
<tr>
<td>PTest</td>
<td>type: Movies, dir: /mnt/sdcard/Movies</td>
</tr>
<tr>
<td>PTest</td>
<td>type: Music, dir: /mnt/sdcard/Music</td>
</tr>
<tr>
<td>PTest</td>
<td>/mnt/sdcard/Music/Susan Boyle - Amazing grace.mp3</td>
</tr>
<tr>
<td>PTest</td>
<td>/mnt/sdcard/Music/Rem - Losing My Religion.mp3</td>
</tr>
<tr>
<td>PTest</td>
<td>type: Notifications, dir: /mnt/sdcard/Notifications</td>
</tr>
<tr>
<td>PTest</td>
<td>type: Pictures, dir: /mnt/sdcard/Pictures</td>
</tr>
<tr>
<td>PTest</td>
<td>type: Podcasts, dir: /mnt/sdcard/Podcasts</td>
</tr>
<tr>
<td>PTest</td>
<td>type: Ringtones, dir: /mnt/sdcard/Ringtones</td>
</tr>
</tbody>
</table>
OBJECT SERIALIZATION
Clicker

• What is Object Serialization?
A. Giving a number to object for sorting
B. Converting object to a byte stream
C. Searching for objects
D. Converting Object to a file
E. Reading Objects from files
Object Serialization

• Taking a runtime data structure or object and converting it to a form that can be stored and / or transmitted
  – converted to a byte stream
• store the object in between program runs
• transmit the object over a network
• store the data, **not the methods / ops**
  – **not the class definition**
Object Serialization

runtime Object ArrayList<Integer>

serialization

Secondary Storage / Network

deserialization

runtime Object ArrayList<Integer>
Serialization - Why?

• Could just do it by hand
  – write out fields and structure to file
  – read it back in
• Serialization provides an abstraction in place of the "by hand" approach
• Much less code to write
• Example, Java has a specification for serializing objects
  – little effort on your part
Serialization in Java

• java.io.Serializable interface
• Here are the methods in the Serializable interface:

• Really, that's it
• A TAG interface
• A way for a class to mark that is Serializable
Serialization in Java

java.util

**Class ArrayList<E>**

java.lang.Object
    java.util.AbstractCollection<E>
    java.util.AbstractList<E>
    java.util.ArrayList<E>

**All Implemented Interfaces:**

- Serializable
- Cloneable
- Iterable<E>
- Collection<E>
- List<E>
- RandomAccess

**Direct Known Subclasses:**

- AttributeList
- RoleList
- RoleUnresolvedList
Serialization in Java

• Data is serialized, not the class definition
• Program that deserializes must have the class definition
• Use an ObjectOutputStream object to write out Serializable objects
  – serialize, deflate, flatten, dehydrate, marshal
• Later, use an ObjectInputStream to read in Serializable objects
  – deserialize, inflate, unflatten, hydrate, unmarshal
ObjectOutputStream Example

• from CS307 / CS314
• Evil Hangman test cases
• play the game and test student results
• for each guess we want the patterns and the number of words in each pattern
  — Map<String, Integer>
ObjectOutputStream Example
Create tests

```java
private static final String DICTIONARY_FILE = "dictionary.txt";
private static final int NUM_TESTS = 13;
private static final String OUTPUT_FILE = "evilGraderTests.eht"

public static void main(String[] args) {
    try {
        ObjectOutputStream os
            = new ObjectOutputStream(new FileOutputStream(new File(OUTPUT_FILE)));

        // make guesses and write results
        for (int i = 0; i < guesses.length(); i++) {
            char guess = guesses.charAt(i);
            Map<String, Integer> result = hm.makeGuess(guess);

            os.writeObject(result);
            os.writeInt(hm.numWordsCurrent());
            os.writeObject(hm.getPattern());
        }
    }
}
```

- LATER FOR EACH GUESS
- data methods (writeInt, ...) for primitives
ObjectOutputStream writeObject

public final void writeObject(Object obj)
    throws IOException

Parameters:
    obj - the object to be written

Throws:
    InvalidClassException - Something is wrong with a class used by serialization.
    NotSerializableException - Some object to be serialized does not implement the 
        java.io.Serializable interface.
    IOException - Any exception thrown by the underlying OutputStream.
ObjectOutputStream Data Methods

<table>
<thead>
<tr>
<th>Method Type</th>
<th>Method Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td><code>writeDouble(double val)</code></td>
<td>Writes a 64 bit double.</td>
</tr>
<tr>
<td>void</td>
<td><code>writeFields()</code></td>
<td>Write the buffered fields to the stream.</td>
</tr>
<tr>
<td>void</td>
<td><code>writeFloat(float val)</code></td>
<td>Writes a 32 bit float.</td>
</tr>
<tr>
<td>void</td>
<td><code>writeInt(int val)</code></td>
<td>Writes a 32 bit int.</td>
</tr>
<tr>
<td>void</td>
<td><code>writeLong(long val)</code></td>
<td>Writes a 64 bit long.</td>
</tr>
</tbody>
</table>

- ... and others
ObjectInputStream

• When ready to run tests

```java
ObjectInputStream reader
    = new ObjectInputStream(new FileInputStream(new File("TEST_FILE_NAME")));
int numMines = reader.readInt();
```

• Make the guesses

```java
for(int i = 0; i < actualGuesses.length(); i++) {
    char ch = actualGuesses.charAt(i);
    System.out.println("\nRound Number: " + roundNumber);

    // read in expected results
    Map<String, Integer> expectedMap
        = (Map<String, Integer>) reader.readObject();
```
Externalizable

• A sub-interface of Serializable
• Gives more control over the format of the serialization to the class itself
• Two methods:

<table>
<thead>
<tr>
<th>Modifier and Type</th>
<th>Method and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void</td>
<td>readExternal(ObjectInput in)</td>
</tr>
<tr>
<td></td>
<td>The object implements the readExternal method to restore its contents by calling the</td>
</tr>
<tr>
<td></td>
<td>methods of DataInput for primitive types and readObject for objects, strings and arrays.</td>
</tr>
<tr>
<td>void</td>
<td>writeExternal(ObjectOutput out)</td>
</tr>
<tr>
<td></td>
<td>The object implements the writeExternal method to save its contents by calling the</td>
</tr>
<tr>
<td></td>
<td>methods of DataOutput for its primitive values or calling the writeObject method of</td>
</tr>
<tr>
<td></td>
<td>ObjectOutput for objects, strings, and arrays.</td>
</tr>
</tbody>
</table>
Externalizable

• ObjectOutputStream will test if object is Serializable
  — if not, throws an exception

• Then tests if Externalizable
  — if so calls the writeExternal method on the object
  — if not, uses default specification for serialization
PARCEL AND PARCELABLE
Bundles Again

• What can you add to Bundles?
• Recall Bundles sent to onCreate when restoring an Activity
• Bundles attached to Intents
• put
  – Bundle, byte, char, CharSequence (includes String), float, Parcelable, Serializable, short, Size (width and height)
  – arrays and ArrayLists of some of those types
Parcelable?

• **Parcel:**
  - Android class for sending data via IPC
  - Inter Process Communication
  - Send an object (data) from one process to another

• Generally faster (at run time) than **Serializable**
  - long term storage vs. short term storage
Parcelable

- interface
- have class implement interface
- implement writeToParcel method
  - not just a Tag interface
  - writes current state of object to Parcel
  - void writeToParcel (Parcel dest, int flags)
  - add a static field named CREATOR to class
    - object that implements Parcelable.Creator interface
public class MyParcelable implements Parcelable {
    private int mData;

    public int describeContents() {
        return 0;
    }

    public void writeToParcel(Parcel out, int flags) {
        out.writeInt(mData);
    }

    public static final Parcelable.Creator<MyParcelable> CREATOR
        = new Parcelable.Creator<MyParcelable>() {
            public MyParcelable createFromParcel(Parcel in) {
                return new MyParcelable(in);
            }

            public MyParcelable[] newArray(int size) {
                return new MyParcelable[size];
            }
        };

    private MyParcelable(Parcel in) {
        mData = in.readInt();
    }
}
OTHER STORAGE OPTIONS
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