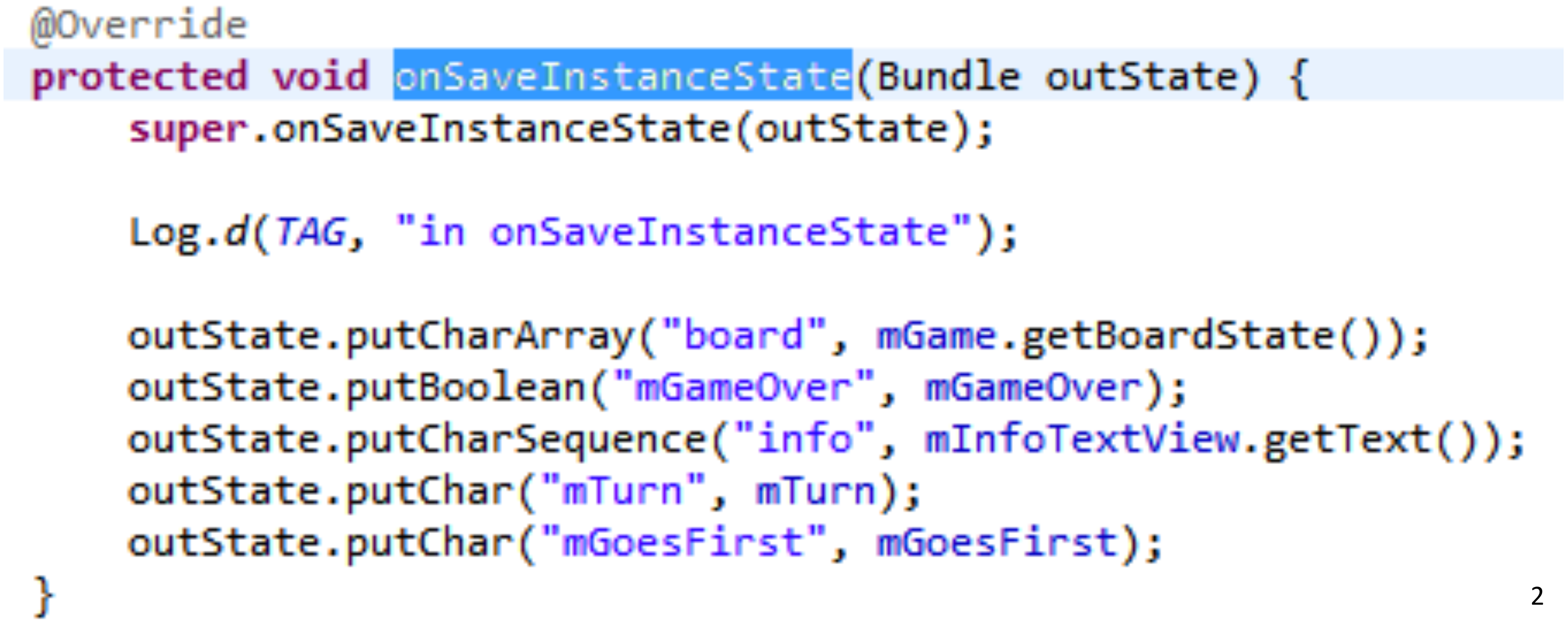
CMSC 628/491: Introduction to Mobile Computing
Local Storage

Nilanjan Banerjee

University of Maryland
Baltimore County
nilanb@umbc.edu

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);
}
```

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
  - images
  - video
  - contact information

SHARED PREFERENCES

Shared Preferences

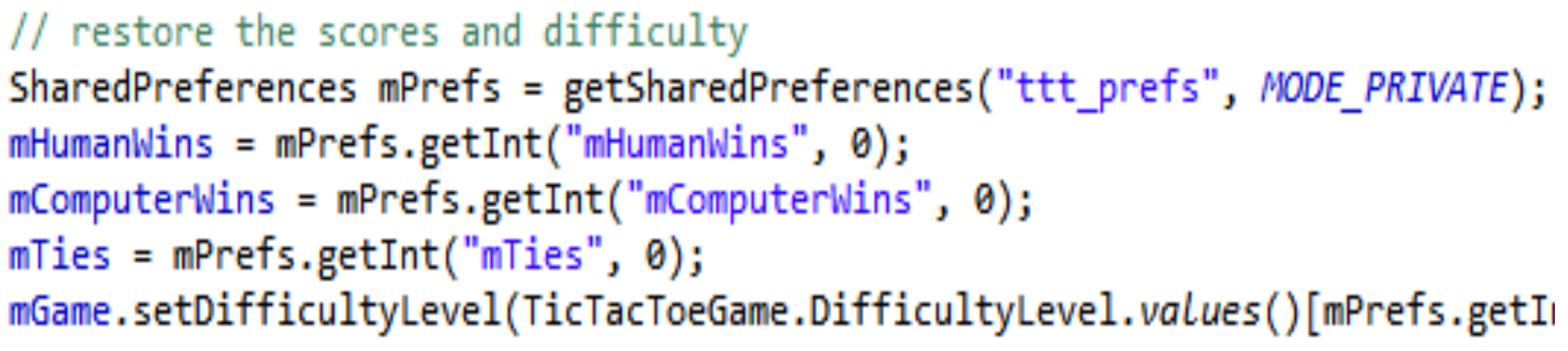
- Private primitive data stored in key-value pairs
- SharedPreferences Class
- Store and retrieve key-value pairs of data
  - keys are Strings
  - values are Strings, Sets of Strings, boolean, float, int, or long
- Not strictly for *preferences*

SharedPreferences

- Several levels of preferences:
- getPreferences(int mode) for the Activities Preferences
- getSharedPreferences(String name, int mode) for a 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)
    - if you want multiple files
  - getPreferences(int mode)

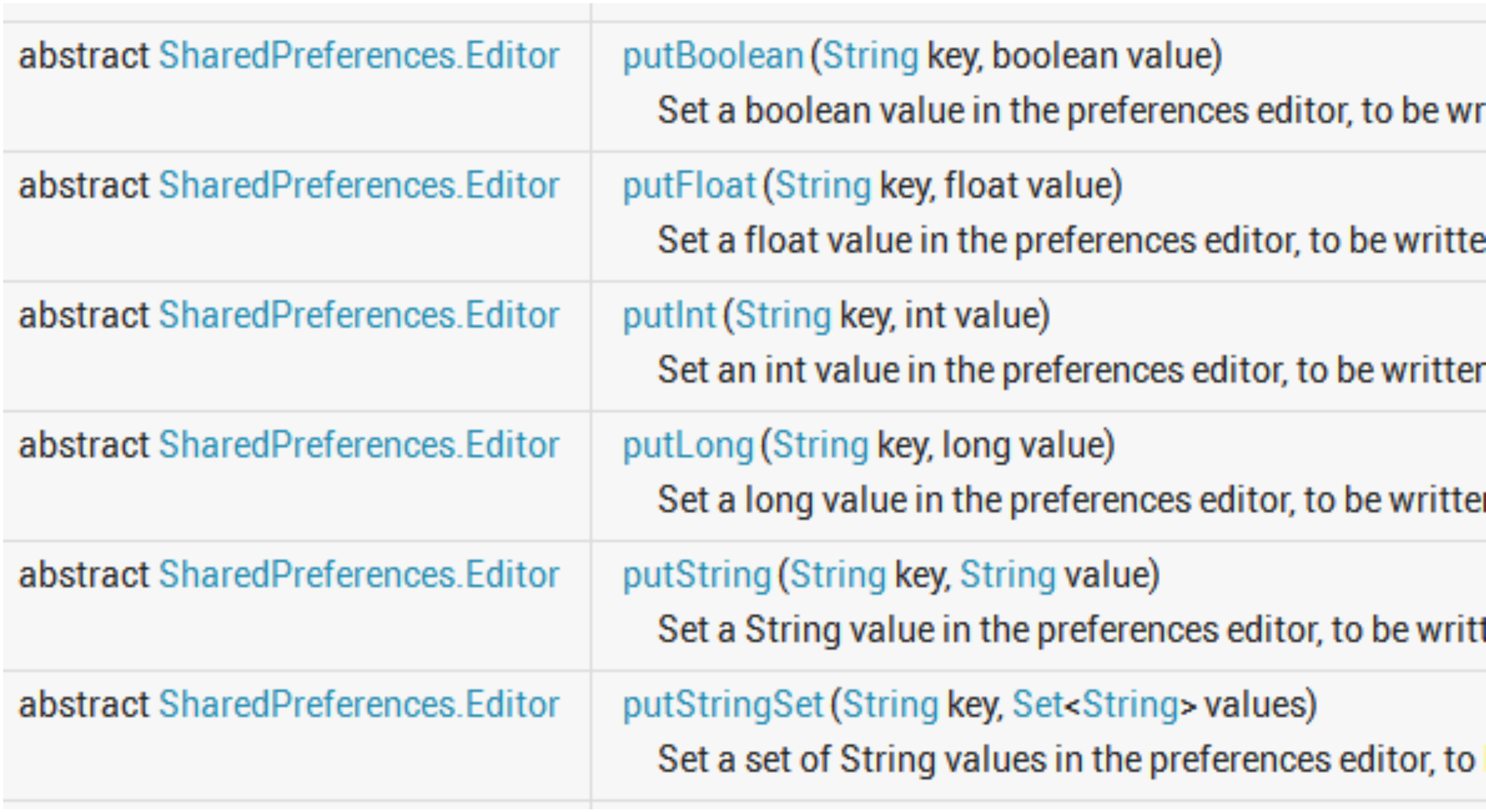


```
// restore the scores and difficulty
SharedPreferences mPrefs = getSharedPreferences("ttt_prefs", MODE_PRIVATE);
mHumanWins = mPrefs.getInt("mHumanWins", 0);
mComputerWins = mPrefs.getInt("mComputerWins", 0);
mTies = mPrefs.getInt("mTies", 0);
mGame.setDifficultyLevel(TicTacToeGame.DifficultyLevel.values()[mPrefs.getI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

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

Data for SharedPreferences



abstract SharedPreferences.Editor	putBoolean (String key, boolean value) Set a boolean value in the preferences editor, to be wr
abstract SharedPreferences.Editor	putFloat (String key, float value) Set a float value in the preferences editor, to be writte
abstract SharedPreferences.Editor	putInt (String key, int value) Set an int value in the preferences editor, to be writter
abstract SharedPreferences.Editor	putLong (String key, long value) Set a long value in the preferences editor, to be writter
abstract SharedPreferences.Editor	putString (String key, String value) Set a String value in the preferences editor, to be writt
abstract SharedPreferences.Editor	putStringSet (String key, Set<String> values) Set a set of String values in the preferences editor, to

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>
```

INTERNAL STORAGE

Internal Storage

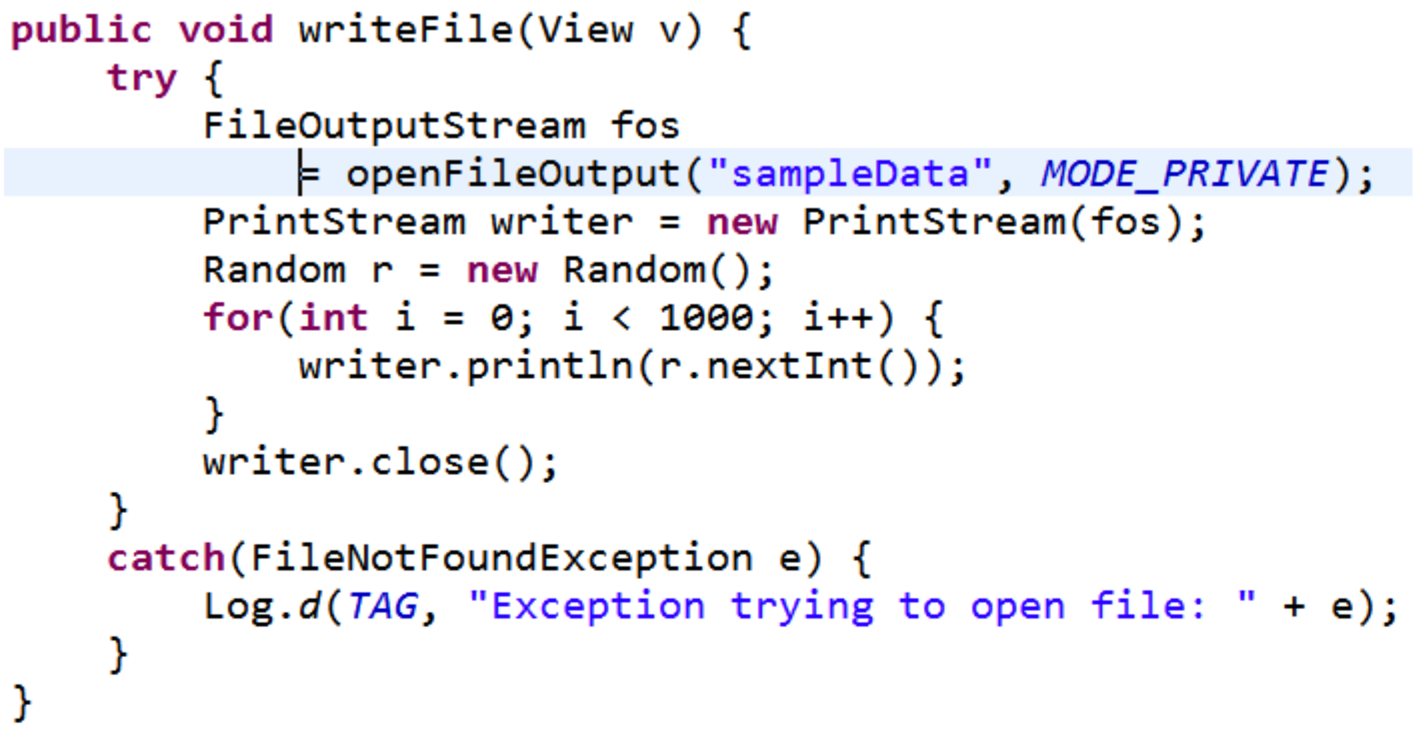
- Private data stored on device memory
- More like traditional file i/o
  - in fact not that different from Java I/O
- by default files are private to your application
  - other apps cannot access
- files removed when app is uninstalled

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

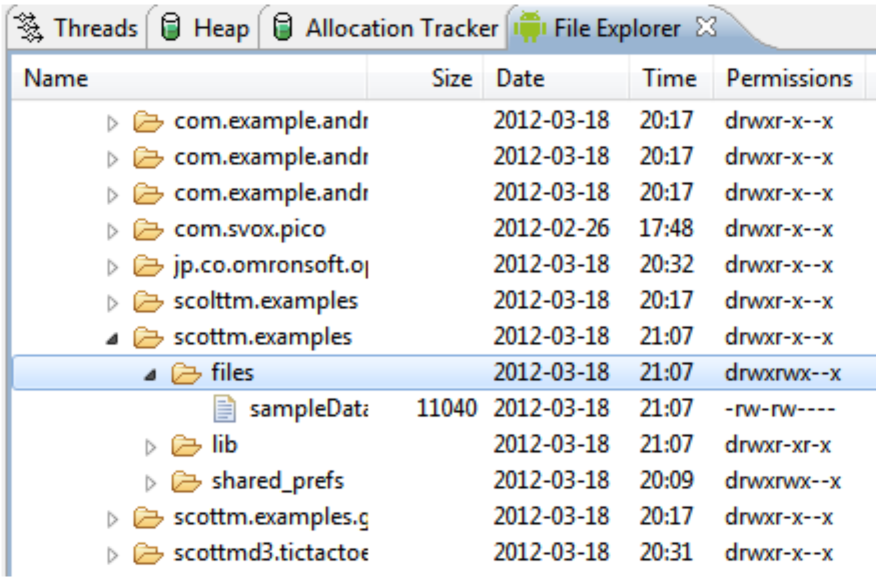
- FileOutputStream writes raw bytes
  - arrays of bytes or single bytes
- Much easier to wrap the FileOutputStream in PrintStream object



```
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.o		2012-03-18	20:32	drwxr-x--x
scolttm.examples		2012-03-18	20:17	drwxr-x--x
scottm.examples		2012-03-18	21:07	drwxr-x--x
files		2012-03-18	21:07	drwxrwx--x
sampleDat	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
scottm.examples.g		2012-03-18	20:17	drwxr-x--x
scottmd3.tictactoe		2012-03-18	20:31	drwxr-x--x

Static Files

- If you need or 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

Internal 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 FILES

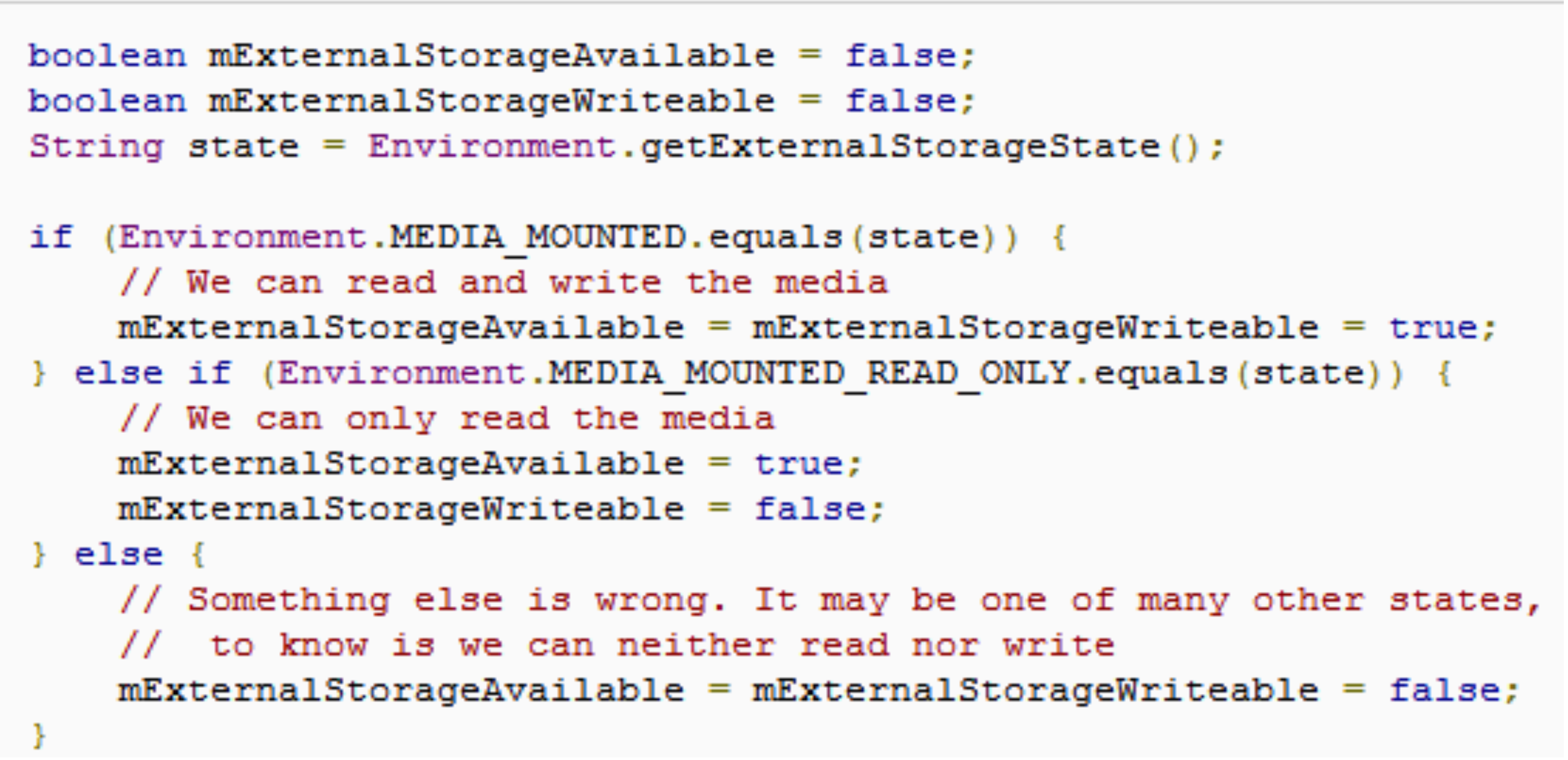
External Storage

- 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

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_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

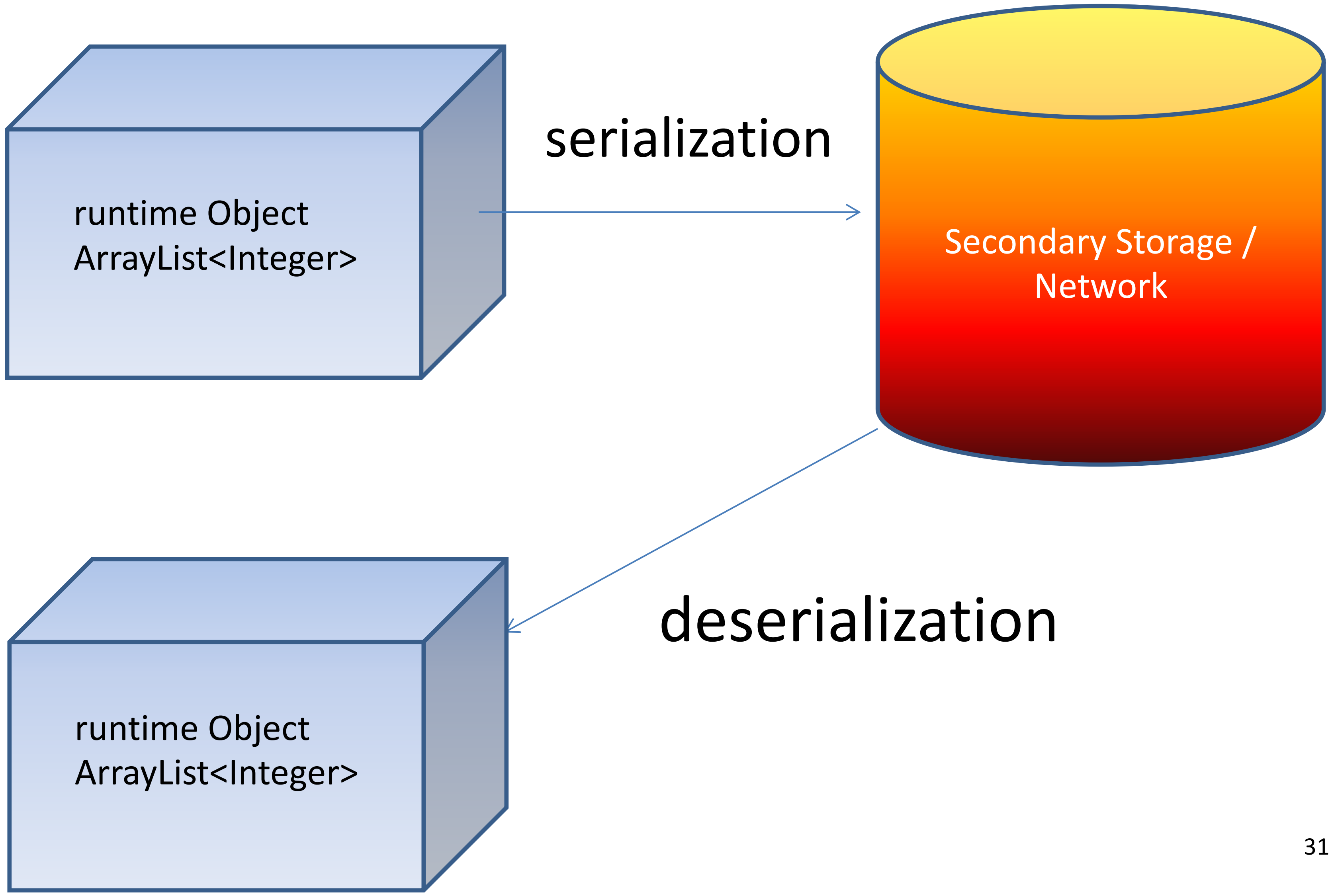
- If you want to save files to be shared with other apps:
- save the files (audio, images, video, etc.) to one of the public directories on the external storage device
- Environment.getExternalStoragePublicDirectory(String type) method returns a File object which is a directory
- same types as getExternalFilesDir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

OBJECT SERIALIZATION

Object Serialization

- Who's done this?
- Taking a runtime data structure or object and converting it to a form that can be stored
  - 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

Object Serialization



Serialization - Why?

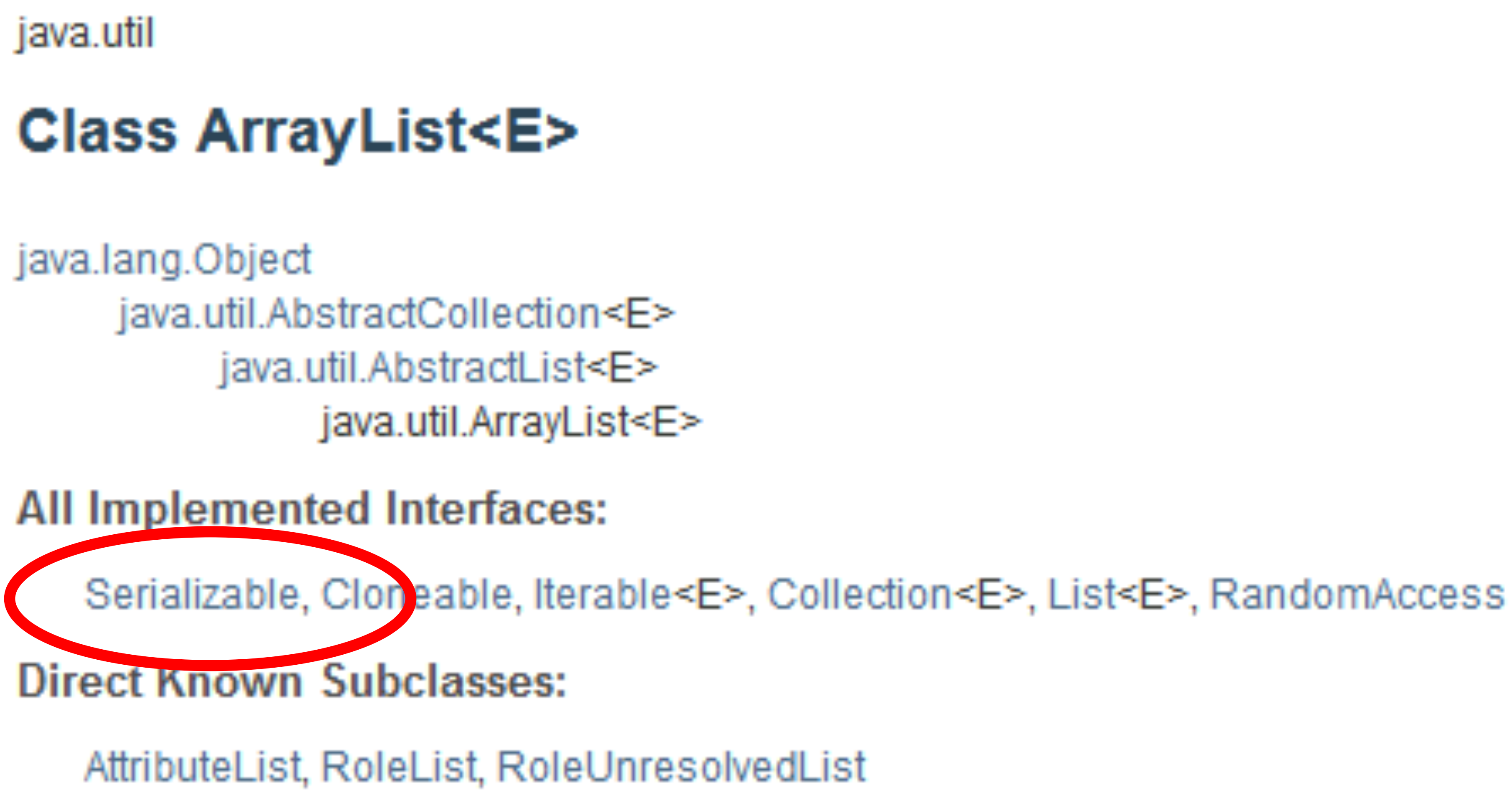
- Could just do it by hand
  - write out fields and structure
  - read it back in
- Serialization provides an abstraction for the "by hand" approach
- Much less code to write
- Example, Java has a specification for serializing objects
  - little effort on your part

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

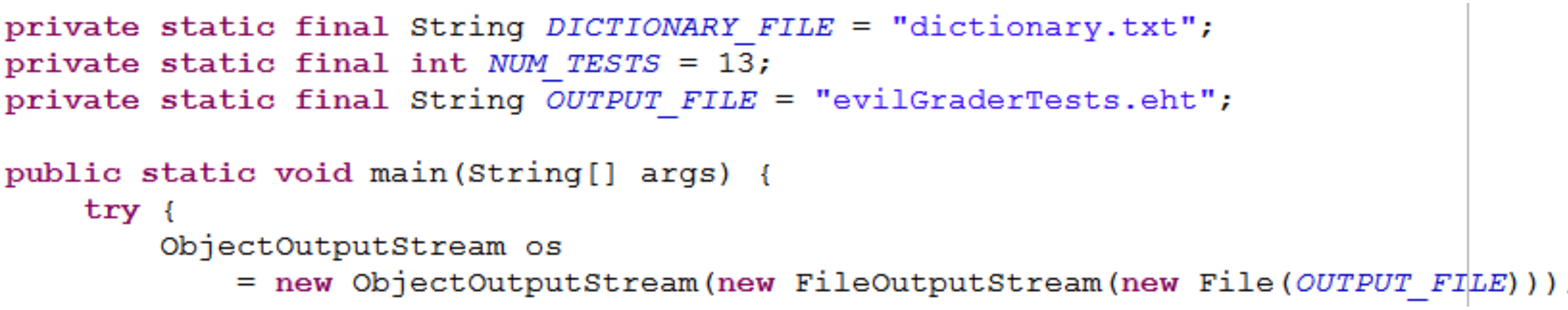
Direct Known Subclasses:

AttributeList, RoleList, RoleUnresolvedList

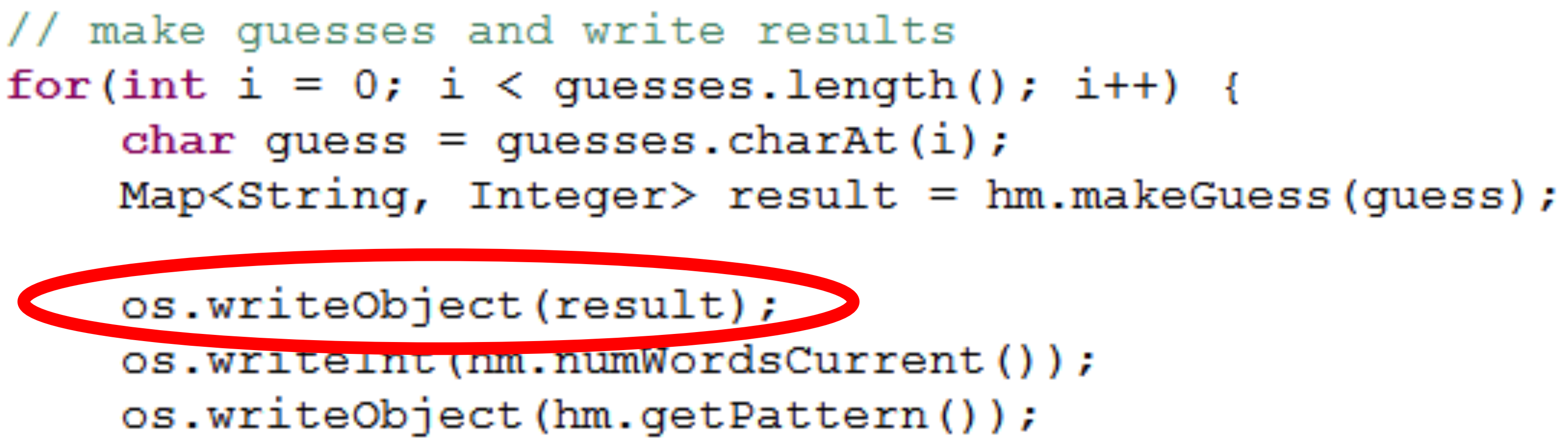
Serialization in Java

- Data is serialized, not class
- Program that deserializes must have the class definition
- Use an ObjectOutputStream object to write out the objects
- Later, use an ObjectInputStream to read in the objects

ObjectOutputStream Example

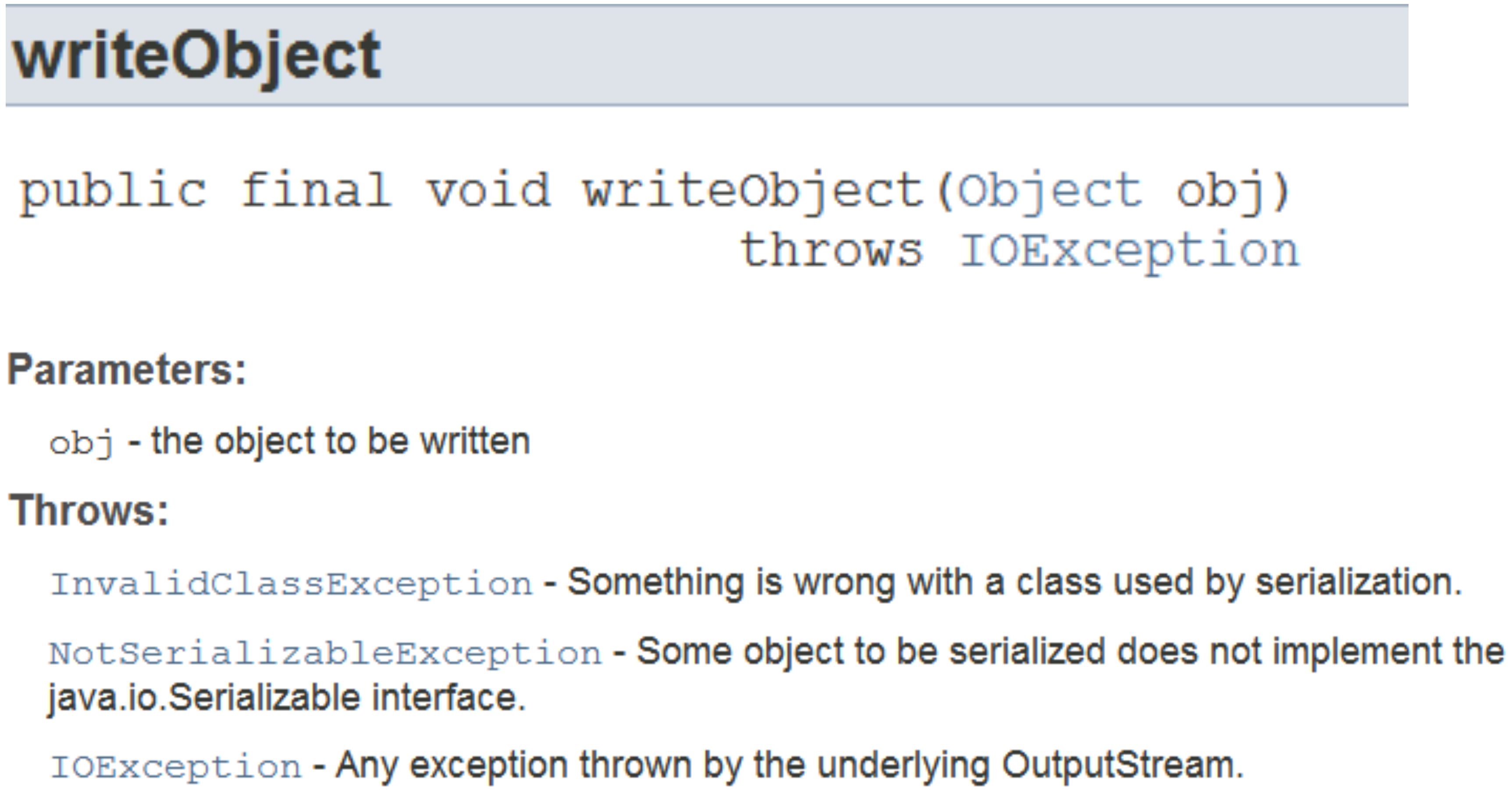


```
private static final String DICTIONARY_FILE = "dictionary.txt";
private static final int NUM_TESTS = 13;
private static final String OUTPUT_FILE = "evilGraderTests.eht";

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());
```

ObjectOutputStream writeObject



writeObject

```
public final void writeObject(Object obj)
                       throws IOException
```

Parameters:

`obj` - the object to be written

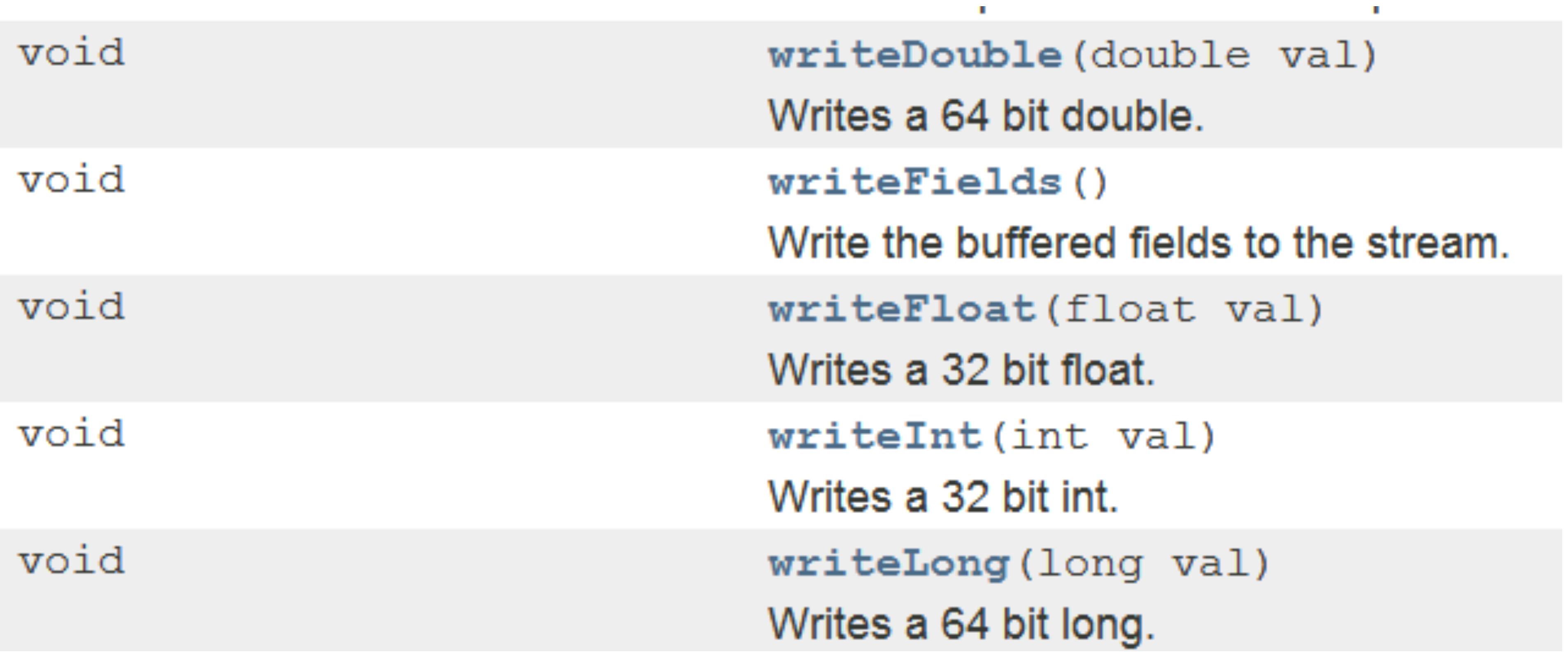
Throws:

`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.

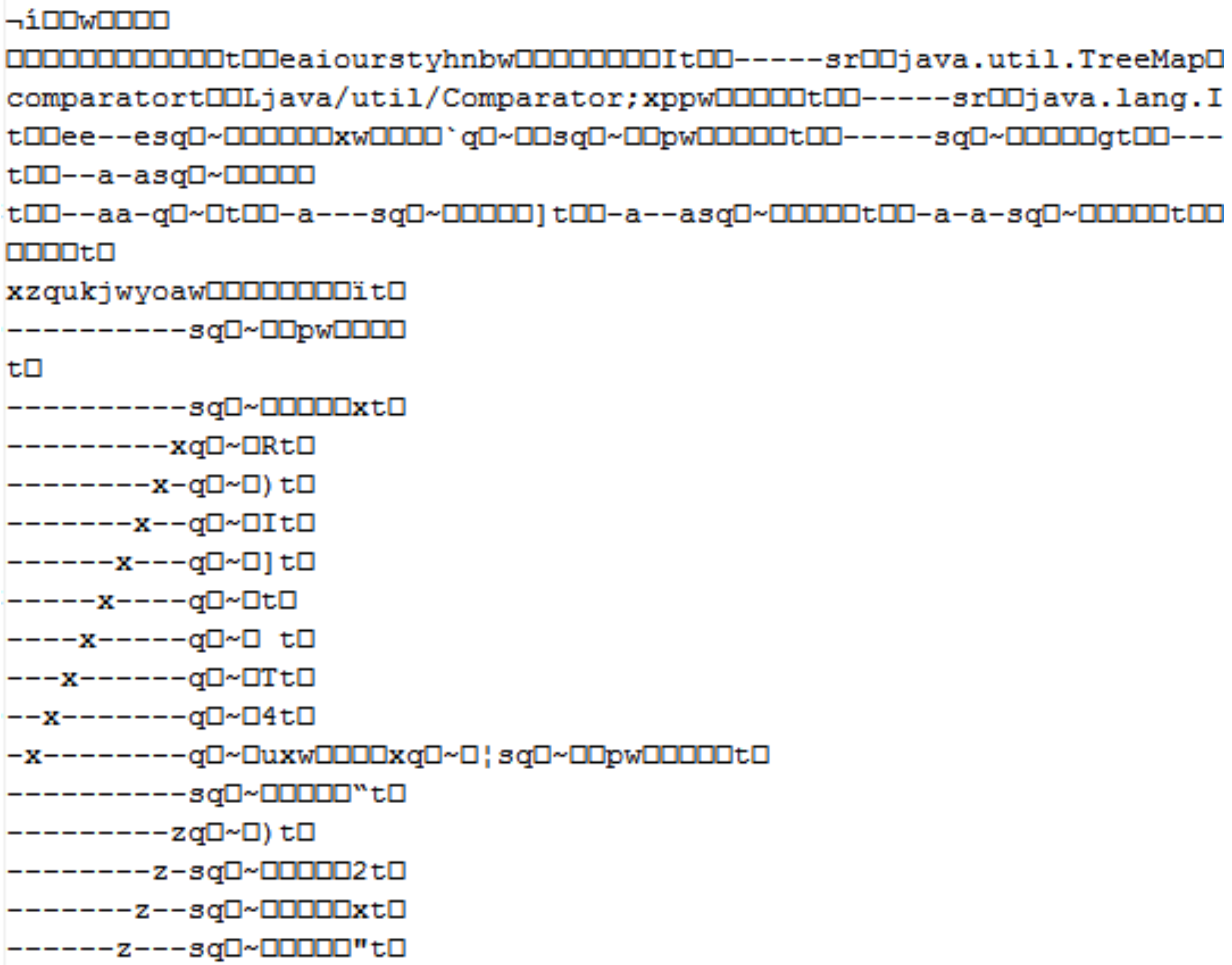
ObjectOutputStream Data Methods



`void`	`writeDouble(double val)` Writes a 64 bit double.
`void`	`writeFields()` Write the buffered fields to the stream.
`void`	`writeFloat(float val)` Writes a 32 bit float.
`void`	`writeInt(int val)` Writes a 32 bit int.
`void`	`writeLong(long val)` Writes a 64 bit long.

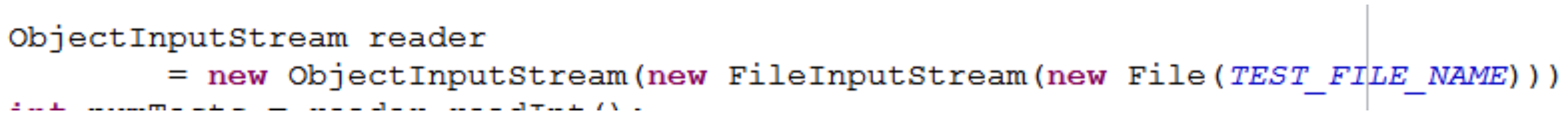
- … and others

Output File - not human readable

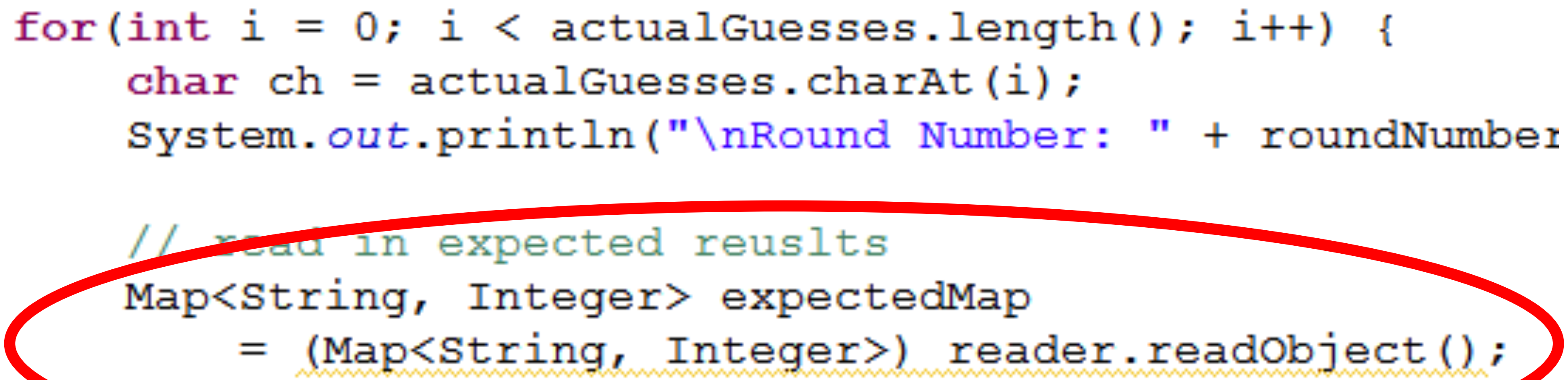


```
¬i□□w□□□□
□□□□□□□□□□□□t□□eaiourstyhnbw□□□□□□□□It□□-----sr□□java.util.TreeMap□
comparatort□□Ljava/util/Comparator;xppw□□□□□t□□-----sr□□java.lang.I
t□□ee--esq□~□□□□□□xw□□□□`q□~□□sq□~□□pw□□□□□t□□-----sq□~□□□□□gt□□---
t□□--a-asq□~□□□□□
t□□--aa-q□~□t□□-a---sq□~□□□□□]t□□-a--asq□~□□□□□t□□-a-a-sq□~□□□□□t□□
□□□□t□
xzqukjwyoaw□□□□□□□□ït□
----------sq□~□□pw□□□□
t□
----------sq□~□□□□□xt□
---------xq□~□Rt□
--------x-q□~□)t□
-------x--q□~□It□
------x---q□~□]t□
-----x----q□~□t□
----x-----q□~□ t□
---x------q□~□Tt□
--x-------q□~□4t□
-x--------q□~□uxw□□□□xq□~□¦sq□~□□pw□□□□□t□
----------sq□~□□□□□“t□
---------zq□~□)t□
--------z-sq□~□□□□□2t□
-------z--sq□~□□□□□xt□
------z---sq□~□□□□□"t□
```

ObjectInputStream



```
ObjectInputStream reader
        = new ObjectInputStream(new FileInputStream(new File(TEST_FILE_NAME)))
```



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

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

OTHER STORAGE OPTIONS

SQLite Database

- Structured data stored in a private database

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