COMP08076
Programming Native App Interaction

- Module coordinator: John Nixon
  - john.nixon@uws.ac.uk, room E259, x3617

- Lecture 5
  - Objects and classes
  - Activities, a revisit
  - Some java
  - Saving data – shared preferences etc.
  - Permissions
  - Fragments

Saving Data

https://developer.android.com/training/basics/data-storage/index.html

- Most Android apps need to save data at some point
  - This might be to save information about app state when the app is paused (so user progress not lost)
  - Most non-trivial apps also need to save user settings
  - Some apps must manage large amounts of information in files and databases.

- Saving key-value pairs of simple data types in a shared preferences file
- Saving arbitrary files in Android's file system
- Using databases managed by SQLite

Storage options

- Shared Preferences
  - Store private primitive data in name/value (key-value) pairs.
- Internal Storage
  - Store private data on the device memory.
- External Storage
  - Store public data on the shared external storage.
- SQLite Databases
  - Store structured data in a private database.
- Network Connection
  - Store data on the web with your own network server

Shared Preferences

- For primitive data:
  - booleans, floats, ints, longs, and strings

- SharedPreferences mySharedPreferences = getSharedPreferences("myPrefsFile", SharedPreferencesActivity.*MODE_PRIVATE*);

- SharedPreferences.Editor editor = mySharedPreferences.edit();

- editor.putString(“clientName”,”john”);
- editor.putBoolean("isTrue", **true**);

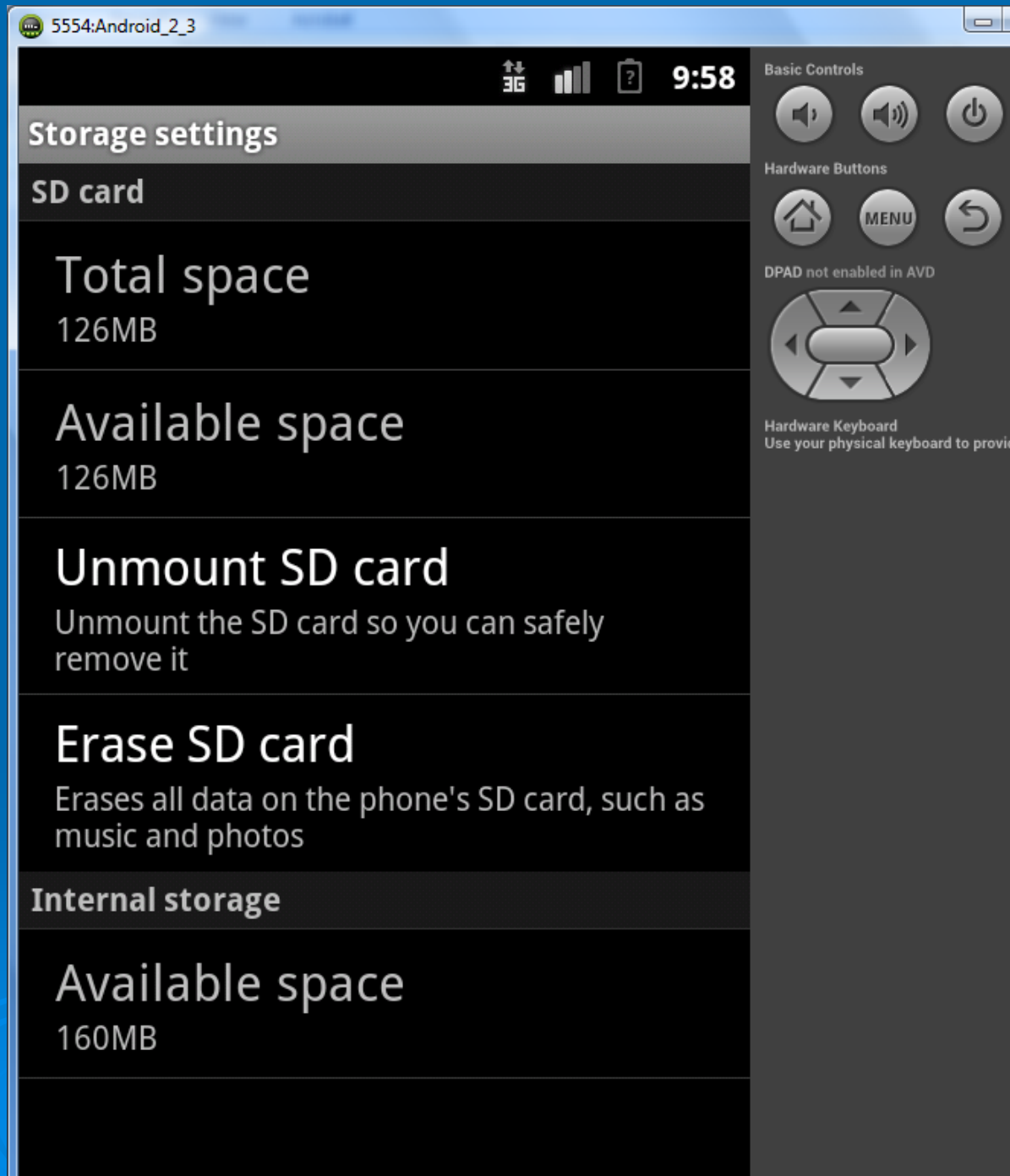
- editor.apply(); //or editor.commit();

- String TextEntry = mySharedPreferences.getString(“clientName", “no name");

Files

- Android file system
- java.io

Storage


5554:Android_2_3
9:58
Storage settings
SD card
Total space
126MB
Available space
126MB
Unmount SD card
Unmount the SD card so you can safely remove it
Erase SD card
Erases all data on the phone's SD card, such as music and photos
Internal storage
Available space
160MB
Basic Controls
Hardware Buttons
MENU
DPAD not enabled in AVD
Hardware Keyboard

- **Internal Storage**
- You can save files directly on the device's internal storage. By default, files saved to the internal storage are private to your application and other applications cannot access them (nor can the user). When the user uninstalls your application, these files are removed.
- http://developer.android.com/guide/topics/data/data-storage.html#filesInternal
- **External storage**
- Every Android-compatible device supports a shared "external storage" that you can use to save files. This can be a removable storage media (such as an SD card) or an internal (non-removable) storage. Files saved to the external storage are world-readable and can be modified by the user when they enable USB mass storage to transfer files on a computer.
- http://developer.android.com/guide/topics/data/data-storage.html#filesExternal

Internal versus External Storage

- Internal storage
  - It's always available.
  - Files saved here are accessible by only your app by default.
  - When the user uninstalls your app, the system removes all your app's files from internal storage.
  - Internal storage is best when you want to be sure that neither the user nor other apps can access your files
  - No permissions required
- External storage
  - It's not always available, because the user can mount the external storage as USB storage and in some cases remove it from the device.
  - It's world-readable, so files saved here may be read outside of your control.
  - When the user uninstalls your app, the system removes your app's files from here only if you save them in the directory from getExternalFilesDir().
  - External storage is the best place for files that don't require access restrictions and for files that you want to share with other apps or allow the user to access with a computer.
  - Permissions required

Objects and Classes

- A class is a template for a type of object
  - All objects are defined by a class
  - All objects of the same class have the same attributes
  - All objects of the same class have a defined set of methods

- Inheritance
  - creating a new class (subclass) with the characteristics of an existing class (superclass) plus characteristics unique to the new class. Can add additional variables, data and methods which override methods in the superclass. Java has *single inheritance*.

- An existing class can be used as the starting point for a new one
  - The new class **extends** the existing one
  - Methods in the new class **override** existing methods

Activity as a Class

- **A class**

- **Activity**
  - A application component that has a **screen** with which users can interact
    - To make a phone call, take a photo etc.
  - Each screen of an application is a different activity (i.e. an application may have several activities)
  - Activity has a life cycle

  - Sub class the activity class to make one
    - Or the AppCompatActivity or the ListActivity

- Discussion on activities
  - http://developer.android.com/guide/components/activities.html

MainActivity.java

```
package com.example.pnai.someuicomponents;
import android.app.Activity;  //plus some other imports

public class MainActivity extends Activity implements whatever here{
// put your declarations here
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
// put what you want to do on start up here
    }

//put your to do stuff method here

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {…}

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {…}
}
```

- Interfaces
  - An interface has some of the features of a class by describing method names and their parameters but does not say how the methods and data items are to be implemented.
- Use of interfaces requires programs to follow a particular design
  - Certain methods must be provided
- Use of interfaces can be recognized by the word implements in a class heading.

- Scope of a variable
  - Class variables, local variables

- *Method overriding*: when a method in a subclass is used instead of one in the superclass with the same name.

- *Method overloading*: two or more methods with the same name but receiving different arguments. Often used to give alternative ways of instantiating objects with a number of constructor methods.

```
e.g.   void move(){
                    //move in default way
                    ……
       }

       void move(int x, int y){
                    // move to position (x,y)
                    ……
       }
```

- To move an object call its move() method

- avatar.move() or avatar.move(50,50)

- Variables and methods can be described as:

- *public* – accessible from any class and inherited by subclass
- *private* – accessible only from within class, not inherited by subclass
- *protected* – accessible only from within class and any subclass – inherited by subclass

- Prefix *final* means that a variable cannot be changed (i.e. it is constant) or a method cannot be overridden.
-
- Prefix *final static* is used for constants to be used in a number of different classes in a program (e.g. Math.PI).
-
- *casting* can be used to change the type of an object.
-
- ```
  float m = 2.5f;
  int j = (int) m;
  ```
  ```

System permissions

http://developer.android.com/guide/topics/security/permissions.html

- Security features are provided through a "permission" mechanism that enforces restrictions on the specific operations that a particular process can perform, and per-URI permissions for granting ad hoc access to specific pieces of data.
- A basic Android application has no permissions associated with it by default, meaning it cannot do anything that would adversely impact the user experience or any data on the device
- To make use of protected features you must include permissions in the manifest file
- E.g.
  - <uses-permission android:name="android.permission.RECEIVE_SMS" />
- Permissions granted to an app when it is installed
- List of permissions
  - http://developer.android.com/reference/android/Manifest.permission.html

Examples of permissions

- <uses-permission android:name=*"android.permission.INTERNET" />*
- <uses-permission android:name=*"android.permission.CAMERA"/>*
- <uses-permission android:name=*"android.permission.READ_CONTACTS"/>*
- <uses-permission android:name=*"android.permission.READ_EXTERNAL_STORAGE"/>*
- <uses-permission android:name=*"android.permission.SEND_SMS"/>*
- <uses-permission android:name=*"android.permission.WRITE_EXTERNAL_STORAGE"/>*

Fragments

- Introduced with Android 3 (api 11) in 2011
- Designed to help reconfigure UI for different screen sizes
  - Phone, tablet, … tv

- Optional layer between activities and widgets
- Extra layer of complexity

- Essentially a way of “chunking” UI

Fragments

- http://developer.android.com/training/basics/fragments/index.html

Assessment 1

- http://moodle.uws.ac.uk/pluginfile.php/301244/course/section/89726/PNAI%20assignment%201%202014_15%20%281%29.doc

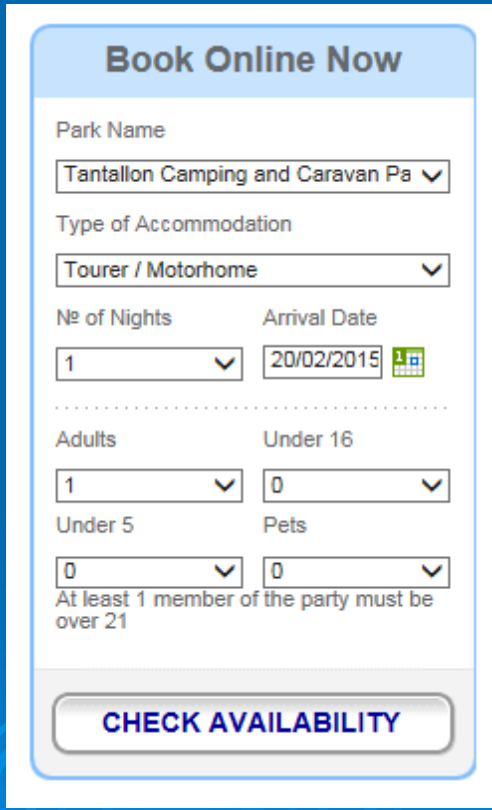
- Minimum to show competency

- Three screens – so three activities
  - Splash screen
  - Data selection screen
  - Data confirmation screen

- Splash screen
- Navigation to other two screens
  - i.e. can start up each of the other two activities
  - Can use buttons
  - Start of today's lab

- Data selection screen
  - Where most of the work is done

- Use of layouts
- Allows users to enter some data according to the brief
- Does some calculation based on the input data – perhaps when a button is pressed
- Outputs the result of the calculation to the screen



- Data confirmation screen
- Can be navigated to from the splash screen

Comments

- // B00******
- // Declaration: I declare that the work submitted is my own unless otherwise stated.

- // use comments to explain code
- // to say if something does not work properly

- // to acknowledge where ideas have come from
- // such as tutorials
- /* block comment over several lines
- */

Enhancements

- Using shared preferences to store data in data selection screen and display on data confirmation screen
- Dealing with all the data in the brief
- Well designed splash screen using, for example, ImageView
- Improved navigation between screens
  - Back navigation, navigation triggered by different components…
- Increased range of components
  - Text fields, radio buttons, spinners, checkbox….
- Dynamic calculation of cost
  - Triggered by appropriate component events
- Good use of comments
- ….