**STEFAN SHKENDEROV**

**Places to Stay**

Report

Contents

[Overview: 3](#_Toc40706375)

[Layout: 3](#_Toc40706376)

[Use of findviewbyId(): 4](#_Toc40706377)

[Use of intents and Bundles: 4](#_Toc40706378)

[Use of Preferences 5](#_Toc40706379)

[File I/O 5](#_Toc40706380)

[Writetofile(): 5](#_Toc40706381)

[ReadFile(): 6](#_Toc40706382)

[Network communication: 6](#_Toc40706383)

[GET request 6](#_Toc40706384)

[POST request 7](#_Toc40706385)

# Overview:

For this assignment, we are required to create an Android Application, consisting of a Map interface, using Mapnik, with which the user is going to interact. In it the user can pinpoint places in its current location, entering information like a name, type and price. These places can be synched to a file or a web service, with auto save features available using Preferences, depending of the user’s choice.

# Layout:

In this application there are two XML layout files located in the “layout” folder.

The first one is used for the main activity. It consists of a full screen map activity. For that reason, the height and with of both the map view and linear layout tags are set to match\_parent.

The second xml file in the layout folder is used for the addplace activity. This is a form, in which the user enters the details of a place at his current location that he wants to save. This layout contains:  
A TextView, which contains the string “Add a place to stay”.

Three EditText fields, which are used for entering the details of the place – Name, Type and price. The first two are of inputType=”text” and the third one is using inputType: numberDecimal, not letting the user input letters where numbers are expected. All of these edit text fields are using with attribute of match\_parent, making them as wide as possible, and a height of wrap\_content. Corresponding placeholders are added to each field via the android:hint attribute, telling the user what is to be entered in each text box, removing the need of having additional text views for each input.

On the bottom, one submit button is added, with height and width attributes of wrap\_content, making it no bigger than it needs to be.

The place, where the user is going to explore the most functionality of the app is the menu. It is placed on the top right of each view. It consists of six items, linking to the add place, preferences, loading places from web or file and saving places to the web or file.

When pressing the preferences link in the menu, the user is taken to the preference activity. Its layout is located in the folder named “xml” and consists of two checkBoxPreference items, with default values of false, with descriptions and IDs added accordingly.

# Use of findviewbyId():

In this application, the findviewbyId method is used in a number of places, when an element either has to be modified or when extracting data from it.

This method is used on the map view in the main activity, which gets it and stores it in a variable. This variable is then used for setting multi touch controls (true), centering the map, setting the zoom level and adding overlay items.

This method is used in the addPlace activity as well. It is called on each text field for collecting the data entered by the user, as well as setting up an onClick listener on the submit button.

# Use of intents and Bundles:

Intents in this application are used to link between activities.

When the user presses the “addPlace” link in the menu, in the main activity a new Intent is created, linking to the addPlace class, and then startActivityForResult function is called on this new intent. This is done, because the application expects data to be received from the addPlace activity. When the user enters the data of the new place to stay and presses the submit button, a new intent and bundle are created. The data from the text fields is put in the bundle, it is then added to the intent via putExtras() and then it is parsed back to the main activity. Then the onActivityResult method is invoked, which gets the bundle from the intent, puts it in a new temporary bundle object, and then gets the data, which is the entered by user values from the text fields via the get functions of the bundle with an argument of the unique key for each value, which were set in the addPlace activity.

The other place where intents are used is when the user opens the preferences activity. However, no data is parsed back like in the case above, because the standard preferences methods are used. This means that in the onOptionsSelected method, when the user selects preferences, the startactivity() method is used instead of startActivityForResult. When the user returns back to the main activity, onActivityResult is not invoked.

# Use of Preferences

In this application, the user can set preferences for:

Saving the places to stay, added by him to a file automatically

Saving the places to stay, added by him to a web service automatically

This is done by creating a new activity, containing two check boxes for each case. This new activity extends PreferenceActivity and in it there is only the onCreate method, containing the addPreferencesFromResource method, which adds what is entered to the check boxes to the preferences. In the main activity, two Boolean variables are created, one for each preference. The onResume method does all the handling of the preferences, meaning creating a SharedPreferences object via preferenceManager.getDefaultSharedPreferences(getApplicationContext()). This will make possible for the values of the preferences to be collected from that object via the getBoolean method. This is done by the key, which is set in the preferences xml file, and both default values are false. When the values are set into the Boolean variables, via if statements the appropriate methods for saving to the web service or to a file are called.

# File I/O

The requirements for this assignment include file input/output functionality. The user needs to have the option to save all current pinpointed locations to a file and load locations from a file. For ease of reading/editing the code the file input and output are separated into two functions in the main activity:

## Writetofile():

In the beginning of the function two file object are created – one for the file directory and one for the file itself. This is done so that the file has its separate directory. Then, an if statement checks if this directory exist, and if not – creates it via the mkdir() function. Same is done with the file object. Then, a printWriter object is created, called to the new file path and the data is saved from the array list of Accommodation objects to the file via the println method of the print writer. The file type is CSV and because of that a comma is added between each attribute of the current object of the array list while saving. Then, the close() function is called. All file saving code is surrounded by a try-catch block to catch any I/O exceptions. The function code itself is in an if statement with the condition of the Accommodation objects array list is not empty. This prevents overwriting an existing file full of places to stay, when the user has no places entered in the application and the preference for automatically saving to file selected. Without this if statement the entries in the files would be deleted. In case the array list is empty, via an else statement, a message is displayed to the user that there is no data to be saved.

## ReadFile():

Similar to writeToFile() , in this function two file objects are created with the same paths as the ones in the write function. Then, a try-catch block is written, catching any exceptions that might occur during reading. In it, a BufferedReader object is created, linking to the file, as well as a String variable, where each line during the iterations of the while loop will be saved. The while loop, which runs until there are no more lines in the file, i.e. while readline() is not null, loops trough the lines of the file. On each iteration the line is saved to the String file, and then the data in it is separated into a string array via the line.split method, which will recognize the comma as the symbol that separates the values. If the string array variable has the length of five, which means that it has read a whole line, a new Accommodation object is created, with the values of the current array, setting them to the appropriate places in the constructor. Then, this object is added to the arrayList of objects and after that the objectsToOverlayItems() function is called, which translates all Accommodation objects in the array list to Overlay items in the overlay items array list and then calls the method of the map view to show all overlay items.

# Network communication:

The requirements for this assignment are that the user must have the option to save and load places to stay from a web service. For this, two classes are created, one for uploading, using a POST request and one for downloading, using GET request.

The Get request class is done via the doInBackground and onPostExecute methods in the class. The first one is used to get the data from the web service and returns it in a String format. Once returned, the second method formats it and displays it to the user via overlay items. The first method first opens a connection to the web service, using get parameters in the URL, as required (username, year and format). The format that we are using, as in the File I/O is CSV. Then the inputstream is fetched and if the response code is 200 the method proceeds to creating a buffered reader, String variables for the return value and for each line in the while loop, which reads every line fetched and adds it to the result string. If the server does not return 200, via an else statement, an error is returned. All of this is surrounded by a try-catch-finally block, catching any possible I/O exceptions. In the “finally”, the connection is disconnected via the disconnect method. The onPostExcecute method is invoked after that, fetching the data returned from the previous method and via a for loop, similar to the file reading, it saves it in the array list of Acommodation objects. Then, a toast text is displayed to the user and the objectsToOverlayItems function is called in order for the new objects to be made into overlay items, as described in the file reading part.

For the POST request and saving to the web service, a new class is created. In a doInBackground method,a URL connection is created to the php script. Then, in an if statement, checking if the Accommodation objects array list is not empty, a for loop is created, iterating trough this array list. On each iteration, a String value is created with the POST data to be sent, which is fetched from the current accommodation object’s getter methods. Then, the connection is opened, the setDoOutput is set to true and a fixed length streaming mode is set to the length of the string, containing the data which will be sent. Then, the outputstream is fetched from the connection, and via the write method of it, the data saved in the string is sent. The script is expected to response with the data that was sent to it, as a confirmation and if the response code is 200 and a new input stream and buffered reader the data is returned. If the server returns anything else, a “no places uploaded” error is returned. Similar to the Get request class, all of this is surrounded by a try-catch-finally-block (all of the above is in the try part) for catching any possible I/O exceptions. In the Finally part, the connection is disconnected. Then, the onPostExecute method fetches what was returned by the DoInBackground method. If the returned value is not the error message specified above, then a toast message appears, notifying the user that the upload was successful.