

1 AmazingRace Android Anwendung

Folgende Dokumentation dokumentiert die Entwicklung der mobilen Anwendung *AmazingRace*, die in Java für die Android-Plattform *API-Level 22*, *Android 5.1.1* entwickelt wurde.

1.1 Lösungsidee

Die Anwendung soll in auf drei Aktivitäten aufgeteilt werden, wobei die Login-Aktivität als Launcher-Aktivität definiert werden soll. Über diese Aktivität wird die Anwendung gestartet. Nach einem erfolgreichem Login soll diese Aktivität beendet werden, da es keinen Grund gibt warum eine Benutzer wieder auf diese Aktivität zurückkehren sollte.

Nach dem Login wird der Benutzer zu einer Übersicht aller Routen weitergeleitet, wo er eine auswählen kann. In dieser Ansicht sollen die einzelnen Routen mit einem Kontext-Menü versehen werden, die je nach Zustand der Route folgende Optionen bereitstellen soll.

1. *Play (Es gibt einen offenen Checkpoint)*
Wechselt zur Checkpoint-Aktivität.
2. *Open (Route bereits beendet)*
Wechselt zur Checkpoint-Aktivität
3. *Reset (Wenn zumindest ein Checkpoint besucht wurde)*
Setzt diese Route zurück und verbleibt auf dieser Aktivität

Ebenfalls soll eine Action-Bar implementiert werden, die folgende Optionen bereitstellen soll.

1. *Reset*
Setzt alle Routen zurück und verbleibt auf dieser Aktivität
2. *Reload*
Lädt alle Routen neu.
3. *Close*
Öffnet einen Dialog, den der Benutzer bestätigen muss und beendet die Anwendung

Für alle Ladeoperationen soll ein ProgressDialog angezeigt werden, der eine dementsprechende Meldung beinhaltet (Z.b.: Load routes ...). Der verwendete Service soll über ein Interface abgebildet werden, somit beinhaltet die benutzenden Klasse keine Referenzen auf die konkrete Implementierung. Die Implementierung soll über eine *Factory* bereitgestellt werden, die die Service Instanz als *Singleton* verwaltet, da hier nur eine Instanz benötigt wird. Des Weiteren sollen *ViewModels* implementiert werden, die die gesamte View-Logik beinhalten und keine Referenzen auf Android-Ressourcen beinhalten sollen.

Die Aktivitäten sollen die gesamte Interaktionslogik beinhalten und die möglichen *ServiceExceptions* der Serviceaufrufe, die über die View-Model erfolgen und von diesen weitergereicht werden, behandeln, da hierbei Referenzen auf Android-Ressourcen benötigt werden. Diese *ServiceException* soll in einer abstrakten Basisklasse behandelt werden, von der alle Aktivitäten ableiten. Somit ist diese Fehlerbehandlung (Anzeige für den Benutzer) in einer Klasse gekapselt und trotzdem können die Aktivitäten ihrerseits Fehlerbehandlungen durchführen.

Übung 3

1.1.1 Architektur

Folgend ist die Architektur der Anwendung *AmazingRace* dokumentiert.

Folgende Abbildung zeigt die Hierarchie der Klasse *AbstractActivity* die als Basisklasse aller Aktivitäten fungiert und gemeinsam genutzte Ressourcen und Konstanten bereitstellt.

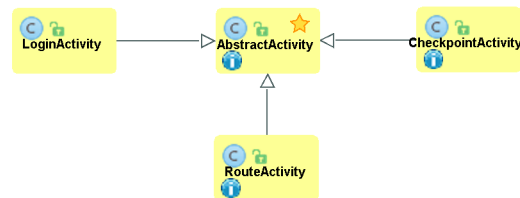


Abbildung 1: Klassenhierarchie der Klasse *AbstractActivity*

Folgende Abbildung zeigt die Paketstruktur der Anwendung *AmazingRace*. Als Wurzelnamensraum wurde *at.fh.ooe.moc5.amazingrace* gewählt unter dem sich alle Ressourcen befinden.

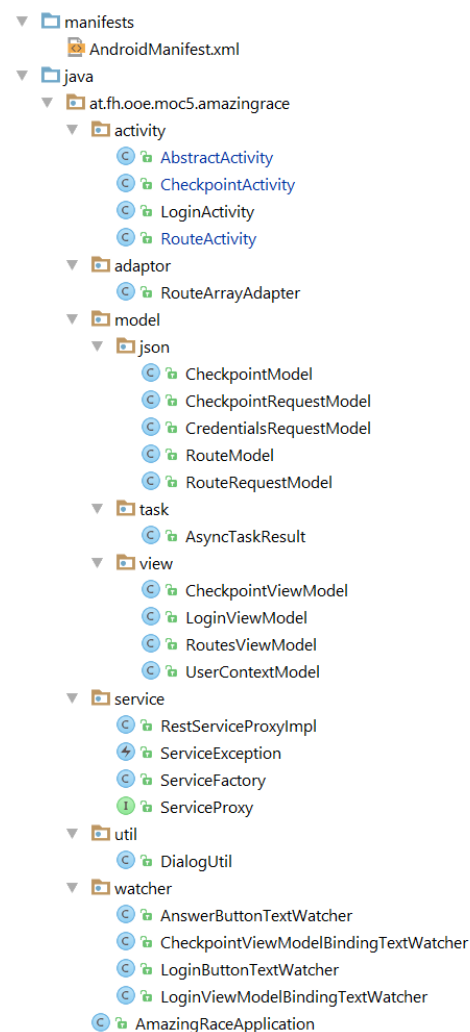


Abbildung 2: Paketstruktur der Anwendung

Übung 3

Folgende Abbildung zeigt die Ressourcenverzeichnisse in denen alle Ressourcen wie *Layout* usw. liegen. Es wurde hierbei darauf verzichtet, die Ressourcen für mehrere Endgeräte und Sprachen zu definieren.

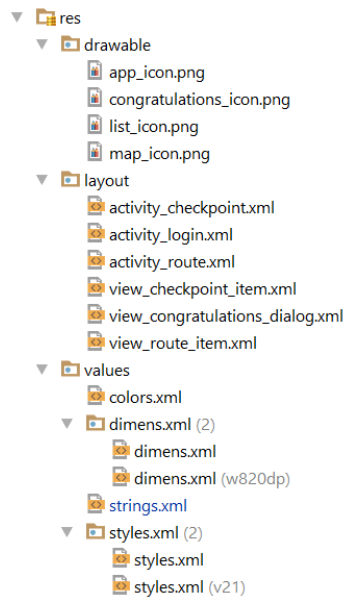


Abbildung 3: Ressourcen der Anwendung

Des Weiteren wurde eine Klasse *AmazingRaceApplication* eingeführt, die die unbehandelten Exceptions behandelt, den Benutzerkontext hält und gemeinsam verwendete Konstanten definiert.

Übung 3

1.2 Source

Im folgenden ist der gesamte Source der Anwendung aufgeführt.

1.2.1 Manifest

Listing 1: AndroidManifest.xml

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="at.fh.ooe.moc5.amazingrace">
4
5      <!-- Permissions -->
6      <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
7      <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
8      <uses-permission android:name="android.permission.INTERNET" />
9      <uses-permission android:name="android.permission.LOCATION_HARDWARE" />
10     <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
11     <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
12
13     <application
14         android:name=".AmazingRaceApplication"
15         android:allowBackup="true"
16         android:icon="@drawable/app_icon"
17         android:label="@string/application_title"
18         android:supportsRtl="true"
19         android:theme="@style/AppTheme">
20
21         <!-- Main Activity -->
22         <activity
23             android:name=".activity.LoginActivity"
24             android:label="@string/application_title">
25             <intent-filter>
26                 <action android:name="android.intent.action.MAIN" />
27                 <category android:name="android.intent.category.LAUNCHER" />
28             </intent-filter>
29         </activity>
30
31         <!-- explicit accessible activities -->
32         <activity
33             android:name=".activity.RouteActivity"
34             android:label="@string/activity_title_route" />
35         <activity
36             android:name=".activity.CheckpointActivity"
37             android:label="@string/activity_title_checkpoint" />
38
39         <!-- Meta-Data for google maps -->
40         <meta-data
41             android:name="com.google.android.gms.version"
42             android:value="@integer/google_play_services_version" />
43         <meta-data
44             android:name="com.google.android.maps.v2.API_KEY"
45             android:value="@string/google_maps_key" />
46     </application>
47 </manifest>

```

Übung 3

1.2.2 Aktivitäten

Listing 2: AmazingRaceApplication.java

```

1 package at.fh.ooe.moc5.amazingrace;
2
3 import android.app.Application;
4 import android.util.Log;
5 import android.widget.Toast;
6
7 import at.fh.ooe.moc5.amazingrace.model.view.UserContextModel;
8
9 /**
10  * Created by Thomas on 12/24/2015.
11  * This is the custom application implementation which handles uncaught exceptions.
12  * The application will get killed on a an uncaught exception.
13  */
14 public class AmazingRaceApplication extends Application {
15
16     public UserContextModel loggedUser;
17
18     public static final String INTENT_EXTRA_ROUTE = "INTENT_EXTRA_ROUTE";
19
20     /**
21      * Registers an exception handler for uncaught exceptions.
22      */
23     public void onCreate() {
24         super.onCreate();
25         // Setup handler for uncaught exceptions.
26         Thread.setDefaultUncaughtExceptionHandler(new Thread.UncaughtExceptionHandler() {
27             @Override
28             public void uncaughtException(Thread thread, Throwable e) {
29                 handleUncaughtException(thread, e);
30             }
31         });
32     }
33
34     /**
35      * Handles uncaught exceptions which are considered critical, therefore the application will
36      * exist with an error.
37      * @param thread the current thread the exception occurred
38      * @param e the occurred exception
39      */
40     public void handleUncaughtException(Thread thread, Throwable e) {
41         e.printStackTrace();
42         Log.e("error", "Uncaught exception caught", e);
43         System.exit(1);
44     }
45
46     /**
47      * gets the logged user context
48      *
49      * @return the user context representing the logged user
50      */
51     public UserContextModel getLoggedUser() {
52         return loggedUser;
53     }
54
55     /**
56      * Sets the logged user which can be accessed by all implemented activities.
57      *
58      * @param loggedUser the logged user

```

Übung 3

```
59     */  
60     public void setLoggedInUser(UserContextModel loggedInUser) {  
61         this.loggedInUser = loggedInUser;  
62     }  
63 }
```

```

1 package at.fh.ooe.moc5.amazingrace.activity;
2
3 import android.app.ProgressDialog;
4 import android.content.Context;
5 import android.content.DialogInterface;
6 import android.content.Intent;
7 import android.net.ConnectivityManager;
8 import android.net.NetworkInfo;
9 import android.os.Bundle;
10 import android.support.v7.app.AppCompatActivity;
11 import android.util.Log;
12 import android.widget.Toast;
13
14 import java.io.Serializable;
15 import java.util.Objects;
16
17 import at.fh.ooe.moc5.amazingrace.AmazingRaceApplication;
18 import at.fh.ooe.moc5.amazingrace.R;
19 import at.fh.ooe.moc5.amazingrace.service.ServiceException;
20 import at.fh.ooe.moc5.amazingrace.util.DialogUtil;
21
22 /**
23  * Created by Thomas on 12/28/2015.
24  * <p/>
25  * This is the base class for all activities which forces the activity to define one view model it
26  * ↪ uses.
27  * This class provides common functionality usable by all activities.
28  */
29 public class AbstractActivity<T extends Serializable> extends AppCompatActivity {
30
31     protected T viewModel;
32     protected boolean restored = Boolean.FALSE;
33     protected AmazingRaceApplication application;
34     protected Boolean validViewModel = Boolean.TRUE;
35
36     private ProgressDialog progress;
37
38     private static final String VIEW_MODEL = "AbstractActivity#VIEW_MODEL";
39
40     // region Constants
41
42     /**
43      * Enumeration for specifying the available menu group ids
44      */
45     public static enum MenuGroup {
46         ROUTE_ITEM(0),
47         OPTIONS(1);
48
49         public final int value;
50
51         private MenuGroup(int value) {
52             this.value = value;
53         }
54
55         public static MenuGroup getMenuGroupForValue(int value) {
56             for (MenuGroup group : MenuGroup.values()) {
57                 if (group.value == value) {
58                     return group;
59                 }
60             }
61         }
62     }
63
64     // endregion
65 }

```

Übung 3

```

60
61         throw new IllegalArgumentException("MenuGroup with value '" + value + "' not found");
62     }
63 }
64
65 /**
66  * Enumeration for specifying the available menu ids
67  */
68 public static enum MenuId {
69     PLAY(0),
70     RESET(1),
71     RESET_ALL_ROUTES(2),
72     CLOSE(3),
73     RELOAD(4);
74
75     public final int value;
76
77     private MenuId(int value) {
78         this.value = value;
79     }
80
81     public static MenuId getMenuIdForValue(int value) {
82         for (MenuId id : MenuId.values()) {
83             if (id.value == value) {
84                 return id;
85             }
86         }
87
88         throw new IllegalArgumentException("MenuGroup with value '" + value + "' not found");
89     }
90 }
91 //endregion
92
93 //region Lifecycle Methods
94
95 /**
96  * Initializes the backed application so that it can be accessed by the concrete activity
97  ↪ application.
98  * It also tries to get the saved view model from the savedInstanceState which can be null
99  ↪ here if haven't been saved before.
100  *
101  * @param savedInstanceState
102  */
103 @Override
104 protected void onCreate(Bundle savedInstanceState) {
105     super.onCreate(savedInstanceState);
106     application = (AmazingRaceApplication) getApplication();
107     if (savedInstanceState != null) {
108         viewModel = (T) savedInstanceState.get(VIEW_MODEL);
109         restored = (viewModel != null);
110     }
111 }
112
113 /**
114  * Re-initializes the application on resume of the activity.
115  */
116 @Override
117 protected void onResume() {
118     super.onResume();
119     application = (AmazingRaceApplication) getApplication();
120     if (viewModel == null) {
121         validViewModel = Boolean.FALSE;
122         openInvalidUserAccountDialog();
123     }
124 }

```


Übung 3

```

121     }
122 }
123
124 /**
125  * Saves the backed view model.
126  *
127  * @param outState the out coming state
128  */
129 @Override
130 protected void onSaveInstanceState(Bundle outState) {
131     super.onSaveInstanceState(outState);
132     outState.putSerializable(VIEW_MODEL, viewModel);
133 }
134 //endregion
135
136 /**
137  * Handles a service exception by displaying a toast with a proper message
138  *
139  * @param exception the occurred ServiceException
140  * @return true if handled here, false otherwise, which indicates that no error code or an
141  ↪ unknown error occurred was set.
142  */
143 protected boolean handleServiceException(ServiceException exception) {
144     Objects.requireNonNull(exception);
145     boolean handled = Boolean.FALSE;
146     ServiceException.ServiceErrorCode errorCode = exception.getErrorCode();
147     if (errorCode != null) {
148         switch (errorCode) {
149             case INVALID_REQUEST:
150                 Toast.makeText(AbstractActivity.this, R.string.error_request_invalid,
151                 ↪ Toast.LENGTH_LONG).show();
152                 handled = Boolean.TRUE;
153                 break;
154             case TIMEOUT:
155                 Toast.makeText(AbstractActivity.this, R.string.error_request_timeout,
156                 ↪ Toast.LENGTH_LONG).show();
157                 handled = Boolean.TRUE;
158                 break;
159             case UNKNOWN:
160                 Toast.makeText(AbstractActivity.this, R.string.error_unknown,
161                 ↪ Toast.LENGTH_LONG).show();
162                 handled = Boolean.TRUE;
163                 break;
164             case INVALID_CREDENTIALS:
165                 openInvalidUserAccountDialog();
166                 handled = Boolean.TRUE;
167         }
168     } else {
169         Toast.makeText(AbstractActivity.this, R.string.error_unknown,
170         ↪ Toast.LENGTH_LONG).show();
171     }
172     Log.e(this.getClass().getSimpleName(), "ServiceException occurred", exception);
173
174     return handled;
175 }
176
177 /**
178  * Opens the invalid user dialog which indicates that the user has become invalid during
179  ↪ application invocation.
180  * After the user clicked ok the user will be logged out.
181  */
182 protected void openInvalidUserAccountDialog() {
183     DialogUtil.createErrorDialog(AbstractActivity.this, R.string.error_no_network, new
184     ↪ DialogInterface.OnClickListener() {

```

Übung 3

```

178         @Override
179         public void onClick(DialogInterface dialog, int which) {
180             dialog.dismiss();
181             application.setLoggedUser(null);
182             finishAffinity();
183             startActivity(new Intent(AbstractActivity.this, LoginActivity.class));
184         }
185     }).show();
186 }
187
188 /**
189  * This method checks if an internet connection is available for the application.
190  * If a network connection is not available then a dialog is shown to notify the user.
191  *
192  * @return true if network is available false otherwise
193  * @see AbstractActivity#isOnline()
194  */
195 protected boolean checkAndDisplayAvailableNetwork() {
196     final boolean online = isOnline();
197     if (!online) {
198         DialogUtil.createErrorDialog(AbstractActivity.this, R.string.error_no_network, new
199             ↳ DialogInterface.OnClickListener() {
200                 @Override
201                 public void onClick(DialogInterface dialog, int which) {
202                     dialog.dismiss();
203                 }
204             }).show();
205     }
206     return online;
207 }
208
209 /**
210  * Opens an dialog to ask the user if he really wants to quit the application.
211  */
212 protected void openCloseApplicationDialog() {
213     DialogUtil.createYesNoAlertDialog(AbstractActivity.this, R.string.dialog_title_warning,
214         ↳ R.string.warning_want_quit, new DialogInterface.OnClickListener() {
215             @Override
216             public void onClick(DialogInterface dialog, int which) {
217                 dialog.dismiss();
218                 switch (which) {
219                     case DialogInterface.BUTTON_POSITIVE:
220                         application.setLoggedUser(null);
221                         finishAffinity();
222                         break;
223                 }
224             }
225         }).show();
226 }
227
228 /**
229  * Opens a progress dialog with an custom message.
230  *
231  * @param messageId the resource id of the message
232  */
233 protected void openProgressDialog(int messageId) {
234     progress = ProgressDialog.show(AbstractActivity.this, getString(R.string.progress_title),
235         ↳ getString(messageId), Boolean.TRUE);
236 }
237
238 /**
239  * Closes the progress dialog.
240  */

```

Übung 3

```
238     protected void closeProgressDialog() {
239         if (progress != null) {
240             progress.dismiss();
241             progress = null;
242         }
243     }
244
245     /**
246      * Helper method for checking if a network connection is available.
247      *
248      * @return true if a network connection is available, false otherwise
249      */
250     protected boolean isOnline() {
251         ConnectivityManager cm =
252             (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
253         NetworkInfo netInfo = cm.getActiveNetworkInfo();
254         return netInfo != null && netInfo.isConnected();
255     }
256 }
```

Listing 4: LoginActivity.java

```

1 package at.fh.ooe.moc5.amazingrace.activity;
2
3 import android.content.Intent;
4 import android.os.*;
5 import android.view.View;
6 import android.widget.Button;
7 import android.widget.EditText;
8 import android.widget.Toast;
9
10 import at.fh.ooe.moc5.amazingrace.R;
11 import at.fh.ooe.moc5.amazingrace.model.task.AsyncTaskResult;
12 import at.fh.ooe.moc5.amazingrace.model.view.LoginViewModel;
13 import at.fh.ooe.moc5.amazingrace.model.view.UserContextModel;
14 import at.fh.ooe.moc5.amazingrace.service.ServiceException;
15 import at.fh.ooe.moc5.amazingrace.watcher.LoginButtonTextWatcher;
16 import at.fh.ooe.moc5.amazingrace.watcher.LoginViewModelBindingTextWatcher;
17
18 /**
19  * This class represents the login activity where an user need to login.
20  */
21 public class LoginActivity extends AbstractActivity<LoginViewModel> implements
    ↪ View.OnClickListener {
22
23     // region Lifecycle Methods
24
25     /**
26      * Sets acitivites content layout and initializes the view model
27      *
28      * @param savedInstanceState
29      */
30     @Override
31     protected void onCreate(Bundle savedInstanceState) {
32         super.onCreate(savedInstanceState);
33         setContentView(R.layout.activity_login);
34         // Here we want always have an new view model
35         viewModel = new LoginViewModel();
36     }
37
38     /**
39      * Prepares the activity views if view model is valid.
40      */
41     @Override
42     protected void onResume() {
43         super.onResume();
44         // View model will always be valid here
45         prepareViews();
46     }
47
48     /**
49      * Open dialog when back button is pressed on this activity.
50      *
51      * @see AbstractActivity#openCloseApplicationDialog();
52      */
53     @Override
54     public void onBackPressed() {
55         super.onBackPressed();
56         openCloseApplicationDialog();
57     }
58     // endregion
59

```

Übung 3

```

60 //region Listeners
61
62 /**
63  * Onclick implementation which
64  *
65  * @param v
66  */
67 public void onClick(View v) {
68     switch (v.getId()) {
69         case R.id.loginBtn:
70             loginUser();
71             break;
72         case R.id.cancelBtn:
73             openCloseApplicationDialog();
74             break;
75     }
76 }
77 //endregion
78
79
80 //region Helper
81
82 /**
83  * Prepares the activity views.
84  */
85 private void prepareViews() {
86     viewModel.reset();
87     // prepare username view
88     EditText username = (EditText) findViewById(R.id.usernameEdTxt);
89     username.addTextChangedListener(new LoginViewModelBindingTextWatcher(viewModel,
90         ↪ username));
91     username.addTextChangedListener(new LoginButtonTextWatcher(this, viewModel));
92
93     // prepare password view
94     EditText password = (EditText) findViewById(R.id.passwordEdTxt);
95     password.addTextChangedListener(new LoginViewModelBindingTextWatcher(viewModel,
96         ↪ password));
97     password.addTextChangedListener(new LoginButtonTextWatcher(this, viewModel));
98
99     // Prepare login button
100     Button loginButton = (Button) findViewById(R.id.loginBtn);
101     loginButton.setEnabled(Boolean.FALSE);
102     loginButton.setOnClickListener(this);
103
104     // prepare cancel button
105     Button cancelButton = (Button) findViewById(R.id.cancelBtn);
106     cancelButton.setOnClickListener(this);
107 }
108
109 /**
110  * Resets the views by setting string value to empty string.
111  */
112 private void resetView() {
113     EditText username = (EditText) findViewById(R.id.usernameEdTxt);
114     EditText password = (EditText) findViewById(R.id.passwordEdTxt);
115
116     // Could be called on non visible activity
117     if ((username != null) && (password != null)) {
118         username.setText("");
119         password.setText("");
120     }
121 }

```

Übung 3

```

121  /**
122   * Performs asynchronous login service call.
123   */
124  private void loginUser() {
125      if (checkAndDisplayAvailableNetwork()) {
126          new AsyncTask<Object, Object, AsyncTaskResult<UserContextModel>>() {
127              @Override
128              protected AsyncTaskResult<UserContextModel> doInBackground(Object... params) {
129                  UserContextModel model = null;
130                  Exception exception = null;
131                  try {
132                      model = viewModel.loginAction();
133                  } catch (Exception e) {
134                      exception = e;
135                  }
136
137                  return new AsyncTaskResult<UserContextModel>(model, exception);
138              }
139
140              @Override
141              protected void onPreExecute() {
142                  super.onPreExecute();
143                  openProgressDialog(R.string.progress_login);
144              }
145
146              @Override
147              protected void onPostExecute(AsyncTaskResult<UserContextModel> result) {
148                  super.onPostExecute(result);
149                  closeProgressDialog();
150                  // Exception occurred
151                  if (result.exception != null) {
152                      resetView();
153                      if (result.exception instanceof ServiceException) {
154                          handleServiceException(((ServiceException) result.exception));
155                      } else {
156                          Toast.makeText(LoginActivity.this, R.string.error_unknown,
157                              ↪ Toast.LENGTH_SHORT).show();
158                      }
159                  }
160                  // Login failed due to invalid credentials
161                  else if (result.result == null) {
162                      resetView();
163                      Toast.makeText(LoginActivity.this, R.string.error_login_failed,
164                          ↪ Toast.LENGTH_SHORT).show();
165                  }
166                  // Login ok
167                  else {
168                      application.setLoggedUser(result.result);
169                      final Intent intent = new Intent(LoginActivity.this, RouteActivity.class);
170                      startActivity(intent);
171                      // Finish this activity because no need to get back to this one
172                      finish();
173                  }
174              }.execute();
175          }
176      }
177  }

```

Listing 5: RouteActivity.java

```

1 package at.fh.ooe.moc5.amazingrace.activity;
2
3 import android.content.Intent;
4 import android.os.AsyncTask;
5 import android.os.Bundle;
6 import android.view.ContextMenu;
7 import android.view.Menu;
8 import android.view.MenuItem;
9 import android.view.View;
10 import android.widget.AdapterView;
11 import android.widget.ListView;
12 import android.widget.Toast;
13
14 import java.util.List;
15
16 import at.fh.ooe.moc5.amazingrace.AmazingRaceApplication;
17 import at.fh.ooe.moc5.amazingrace.R;
18 import at.fh.ooe.moc5.amazingrace.adapter.RouteArrayAdapter;
19 import at.fh.ooe.moc5.amazingrace.model.json.RouteModel;
20 import at.fh.ooe.moc5.amazingrace.model.task.AsyncTaskResult;
21 import at.fh.ooe.moc5.amazingrace.model.view.RoutesViewModel;
22 import at.fh.ooe.moc5.amazingrace.service.ServiceException;
23
24 /**
25  * This class represents the route activity which lists all available routes to the user.
26  */
27 public class RouteActivity extends AbstractActivity<RoutesViewModel> implements
↳ AdapterView.OnItemClickListener {
28
29     //region Lifecycle Methods
30
31     /**
32      * Initializes this activity.
33      *
34      * @param savedInstanceState
35      */
36     @Override
37     protected void onCreate(Bundle savedInstanceState) {
38         super.onCreate(savedInstanceState);
39         setContentView(R.layout.activity_route);
40         // Only if user is logged and view model hasn't been restored from bundle
41         if ((application.getLoggedUser() != null) && (!restored)) {
42             viewModel = new RoutesViewModel(application.getLoggedUser());
43         }
44         // Register route list for context menu
45         registerForContextMenu(findViewById(R.id.listRoute));
46     }
47
48     /**
49      * Prepares the activity views if view model is valid
50      */
51     @Override
52     protected void onResume() {
53         super.onResume();
54         if (validViewModel) {
55             prepareView();
56             loadRoutes();
57         }
58     }
59

```

Übung 3

```

60  /**
61   * Opens application close dialog if back button is pressed on this activity.
62   *
63   * @see AbstractActivity#openCloseApplicationDialog();
64   */
65  @Override
66  public void onBackPressed() {
67      super.onBackPressed();
68      openCloseApplicationDialog();
69  }
70
71  /**
72   * Creates the context menu items for the list view.
73   *
74   * @param menu    the menu to add options too.
75   * @param v        the view to add options for
76   * @param menuInfo the menuinfo related to the view
77   */
78  @Override
79  public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo
    ↪ menuInfo) {
80      super.onCreateContextMenu(menu, v, menuInfo);
81      // Creates options for list view
82      if (R.id.listRoute == v.getId()) {
83          onCreateRouteListContextMenu(menu, (ListView) v, (AdapterView.AdapterContextMenuInfo)
    ↪ menuInfo);
84      }
85  }
86
87  /**
88   * Creates the menu items for the application action bar
89   *
90   * @param menu the menu to add options too
91   * @return true
92   */
93  @Override
94  public boolean onCreateOptionsMenu(Menu menu) {
95      menu.add(MenuGroup.OPTIONS.value, MenuId.RESET_ALL_ROUTES.value, 0,
    ↪ R.string.action_reset_all);
96      menu.add(MenuGroup.OPTIONS.value, MenuId.RELOAD.value, 0, R.string.action_reload);
97      menu.add(MenuGroup.OPTIONS.value, MenuId.CLOSE.value, 0, R.string.action_close);
98      return Boolean.TRUE;
99  }
100
101  /**
102   * Handles the menu options select and delegates to the proper method.
103   *
104   * @param item the menu item which has been selected
105   * @return
106   */
107  @Override
108  public boolean onOptionsItemSelected(MenuItem item) {
109      switch (MenuId.getMenuIdForValue(item.getItemId())) {
110          case RESET_ALL_ROUTES:
111              resetRoute(null);
112              return Boolean.TRUE;
113          case RELOAD:
114              loadRoutes();
115              return Boolean.TRUE;
116          case CLOSE:
117              openCloseApplicationDialog();
118              return Boolean.TRUE;
119          default:

```


Übung 3

```

120         return super.onOptionsItemSelected(item);
121     }
122 }
123
124 //endregion
125
126 //region Utilities
127
128 /**
129  * Creates the menu items for the list view
130  *
131  * @param menu    the context menu to add items too
132  * @param v        the list view to create items for
133  * @param menuInfo the list view related menu info
134  */
135 private void onCreateRouteListContextMenu(ContextMenu menu, ListView v,
136     AdapterView.AdapterContextMenuInfo menuInfo) {
137     RouteModel route = (RouteModel) v.getItemAtPosition(menuInfo.position);
138     if (route.getNextCheckpoint() != null) {
139         menu.add(MenuGroup.ROUTE_ITEM.value, MenuId.PLAY.value, 0, R.string.action_play);
140         if (!route.getVisitedCheckpoints().isEmpty()) {
141             menu.add(MenuGroup.ROUTE_ITEM.value, MenuId.RESET.value, 1,
142                 ↪ R.string.action_reset);
143         }
144     } else {
145         menu.add(MenuGroup.ROUTE_ITEM.value, MenuId.PLAY.value, 0, R.string.action_open);
146         menu.add(MenuGroup.ROUTE_ITEM.value, MenuId.RESET.value, 0, R.string.action_reset);
147     }
148 }
149
150 /**
151  * Handles the selected context item
152  *
153  * @param item the selected menu item
154  * @return true if item has been handled
155  */
156 @Override
157 public boolean onContextItemSelected(MenuItem item) {
158     switch (MenuGroup.getMenuGroupForValue(item.getGroupId())) {
159         case ROUTE_ITEM:
160             return handleRouteItemSelected(item);
161         default:
162             return super.onContextItemSelected(item);
163     }
164 }
165
166 /**
167  * Handles the selected route menu item.
168  *
169  * @param item the selected menu item
170  * @return true if item has been handled
171  */
172 private boolean handleRouteItemSelected(MenuItem item) {
173     final RouteModel model = (RouteModel) ((ListView)
174         ↪ findViewById(R.id.listRoute)).getItemAtPosition(((AdapterView.AdapterContextMenuInfo)
175         ↪ item.getMenuInfo()).position);
176     switch (MenuId.getMenuIdForValue(item.getItemId())) {
177         case PLAY:
178             goToCheckpointActivity(model);
179             return Boolean.TRUE;
180         case RESET:
181             resetRoute(model);
182             return Boolean.TRUE;
183     }
184 }

```

Übung 3

```

180         default:
181             return Boolean.FALSE;
182     }
183 }
184
185 /**
186  * Prepares this activity views.
187  */
188 private void prepareView() {
189     // prepare route list view
190     ListView listView = (ListView) findViewById(R.id.listRoute);
191     if (listView.getAdapter() == null) {
192         final RouteArrayAdapter adapter = new RouteArrayAdapter(RouteActivity.this);
193         listView.setAdapter(adapter);
194         listView.setOnItemClickListener(this);
195     }
196 }
197
198 /**
199  * Loads the routes into the adapter
200  */
201 private void loadRoutes() {
202     if (checkAndDisplayAvailableNetwork()) {
203         new AsyncTask<Object, Object, AsyncTaskResult<List<RouteModel>>>() {
204             @Override
205             protected AsyncTaskResult<List<RouteModel>> doInBackground(Object... params) {
206                 List<RouteModel> routes = null;
207                 Exception exception = null;
208                 try {
209                     routes = viewModel.loadRoutes();
210                 } catch (Exception e) {
211                     exception = e;
212                 }
213
214                 return new AsyncTaskResult<List<RouteModel>>(routes, exception);
215             }
216
217             @Override
218             protected void onPreExecute() {
219                 super.onPreExecute();
220                 openProgressDialog(R.string.progress_loading_routes);
221             }
222
223             @Override
224             protected void onPostExecute(AsyncTaskResult<List<RouteModel>> result) {
225                 super.onPostExecute(result);
226                 closeProgressDialog();
227                 // Error occurred
228                 if (result.exception != null) {
229                     // ServiceException occurred
230                     if (result.exception instanceof ServiceException) {
231                         handleServiceException(((ServiceException) result.exception));
232                     } else {
233                         Toast.makeText(RouteActivity.this, R.string.error_unknown,
234                                     ↳ Toast.LENGTH_LONG).show();
235                     }
236                 }
237                 // Lists where loaded
238                 else {
239                     RouteArrayAdapter adapter = (RouteArrayAdapter) ((ListView)
240                                     ↳ findViewById(R.id.listRoute)).getAdapter();
241                     adapter.clear();
242                     adapter.addAll(result.result);
243                 }
244             }
245         }.execute();
246     }
247 }
248
249 /**
250  * ...
251  */
252 }
253
254 /**
255  * ...
256  */
257 }
258
259 /**
260  * ...
261  */
262 }
263
264 /**
265  * ...
266  */
267 }
268
269 /**
270  * ...
271  */
272 }
273
274 /**
275  * ...
276  */
277 }
278
279 /**
280  * ...
281  */
282 }
283
284 /**
285  * ...
286  */
287 }
288
289 /**
290  * ...
291  */
292 }
293
294 /**
295  * ...
296  */
297 }
298
299 /**
300  * ...
301  */
302 }
303
304 /**
305  * ...
306  */
307 }
308
309 /**
310  * ...
311  */
312 }
313
314 /**
315  * ...
316  */
317 }
318
319 /**
320  * ...
321  */
322 }
323
324 /**
325  * ...
326  */
327 }
328
329 /**
330  * ...
331  */
332 }
333
334 /**
335  * ...
336  */
337 }
338
339 /**
340  * ...
341  */
342 }
343
344 /**
345  * ...
346  */
347 }
348
349 /**
350  * ...
351  */
352 }
353
354 /**
355  * ...
356  */
357 }
358
359 /**
360  * ...
361  */
362 }
363
364 /**
365  * ...
366  */
367 }
368
369 /**
370  * ...
371  */
372 }
373
374 /**
375  * ...
376  */
377 }
378
379 /**
380  * ...
381  */
382 }
383
384 /**
385  * ...
386  */
387 }
388
389 /**
390  * ...
391  */
392 }
393
394 /**
395  * ...
396  */
397 }
398
399 /**
400  * ...
401  */
402 }
403
404 /**
405  * ...
406  */
407 }
408
409 /**
410  * ...
411  */
412 }
413
414 /**
415  * ...
416  */
417 }
418
419 /**
420  * ...
421  */
422 }
423
424 /**
425  * ...
426  */
427 }
428
429 /**
430  * ...
431  */
432 }
433
434 /**
435  * ...
436  */
437 }
438
439 /**
440  * ...
441  */
442 }
443
444 /**
445  * ...
446  */
447 }
448
449 /**
450  * ...
451  */
452 }
453
454 /**
455  * ...
456  */
457 }
458
459 /**
460  * ...
461  */
462 }
463
464 /**
465  * ...
466  */
467 }
468
469 /**
470  * ...
471  */
472 }
473
474 /**
475  * ...
476  */
477 }
478
479 /**
480  * ...
481  */
482 }
483
484 /**
485  * ...
486  */
487 }
488
489 /**
490  * ...
491  */
492 }
493
494 /**
495  * ...
496  */
497 }
498
499 /**
500  * ...
501  */
502 }
503
504 /**
505  * ...
506  */
507 }
508
509 /**
510  * ...
511  */
512 }
513
514 /**
515  * ...
516  */
517 }
518
519 /**
520  * ...
521  */
522 }
523
524 /**
525  * ...
526  */
527 }
528
529 /**
530  * ...
531  */
532 }
533
534 /**
535  * ...
536  */
537 }
538
539 /**
540  * ...
541  */
542 }
543
544 /**
545  * ...
546  */
547 }
548
549 /**
550  * ...
551  */
552 }
553
554 /**
555  * ...
556  */
557 }
558
559 /**
560  * ...
561  */
562 }
563
564 /**
565  * ...
566  */
567 }
568
569 /**
570  * ...
571  */
572 }
573
574 /**
575  * ...
576  */
577 }
578
579 /**
580  * ...
581  */
582 }
583
584 /**
585  * ...
586  */
587 }
588
589 /**
590  * ...
591  */
592 }
593
594 /**
595  * ...
596  */
597 }
598
599 /**
600  * ...
601  */
602 }
603
604 /**
605  * ...
606  */
607 }
608
609 /**
610  * ...
611  */
612 }
613
614 /**
615  * ...
616  */
617 }
618
619 /**
620  * ...
621  */
622 }
623
624 /**
625  * ...
626  */
627 }
628
629 /**
630  * ...
631  */
632 }
633
634 /**
635  * ...
636  */
637 }
638
639 /**
640  * ...
641  */
642 }
643
644 /**
645  * ...
646  */
647 }
648
649 /**
650  * ...
651  */
652 }
653
654 /**
655  * ...
656  */
657 }
658
659 /**
660  * ...
661  */
662 }
663
664 /**
665  * ...
666  */
667 }
668
669 /**
670  * ...
671  */
672 }
673
674 /**
675  * ...
676  */
677 }
678
679 /**
680  * ...
681  */
682 }
683
684 /**
685  * ...
686  */
687 }
688
689 /**
690  * ...
691  */
692 }
693
694 /**
695  * ...
696  */
697 }
698
699 /**
700  * ...
701  */
702 }
703
704 /**
705  * ...
706  */
707 }
708
709 /**
710  * ...
711  */
712 }
713
714 /**
715  * ...
716  */
717 }
718
719 /**
720  * ...
721  */
722 }
723
724 /**
725  * ...
726  */
727 }
728
729 /**
730  * ...
731  */
732 }
733
734 /**
735  * ...
736  */
737 }
738
739 /**
740  * ...
741  */
742 }
743
744 /**
745  * ...
746  */
747 }
748
749 /**
750  * ...
751  */
752 }
753
754 /**
755  * ...
756  */
757 }
758
759 /**
760  * ...
761  */
762 }
763
764 /**
765  * ...
766  */
767 }
768
769 /**
770  * ...
771  */
772 }
773
774 /**
775  * ...
776  */
777 }
778
779 /**
780  * ...
781  */
782 }
783
784 /**
785  * ...
786  */
787 }
788
789 /**
790  * ...
791  */
792 }
793
794 /**
795  * ...
796  */
797 }
798
799 /**
800  * ...
801  */
802 }
803
804 /**
805  * ...
806  */
807 }
808
809 /**
810  * ...
811  */
812 }
813
814 /**
815  * ...
816  */
817 }
818
819 /**
820  * ...
821  */
822 }
823
824 /**
825  * ...
826  */
827 }
828
829 /**
830  * ...
831  */
832 }
833
834 /**
835  * ...
836  */
837 }
838
839 /**
840  * ...
841  */
842 }
843
844 /**
845  * ...
846  */
847 }
848
849 /**
850  * ...
851  */
852 }
853
854 /**
855  * ...
856  */
857 }
858
859 /**
860  * ...
861  */
862 }
863
864 /**
865  * ...
866  */
867 }
868
869 /**
870  * ...
871  */
872 }
873
874 /**
875  * ...
876  */
877 }
878
879 /**
880  * ...
881  */
882 }
883
884 /**
885  * ...
886  */
887 }
888
889 /**
890  * ...
891  */
892 }
893
894 /**
895  * ...
896  */
897 }
898
899 /**
900  * ...
901  */
902 }
903
904 /**
905  * ...
906  */
907 }
908
909 /**
910  * ...
911  */
912 }
913
914 /**
915  * ...
916  */
917 }
918
919 /**
920  * ...
921  */
922 }
923
924 /**
925  * ...
926  */
927 }
928
929 /**
930  * ...
931  */
932 }
933
934 /**
935  * ...
936  */
937 }
938
939 /**
940  * ...
941  */
942 }
943
944 /**
945  * ...
946  */
947 }
948
949 /**
950  * ...
951  */
952 }
953
954 /**
955  * ...
956  */
957 }
958
959 /**
960  * ...
961  */
962 }
963
964 /**
965  * ...
966  */
967 }
968
969 /**
970  * ...
971  */
972 }
973
974 /**
975  * ...
976  */
977 }
978
979 /**
980  * ...
981  */
982 }
983
984 /**
985  * ...
986  */
987 }
988
989 /**
990  * ...
991  */
992 }
993
994 /**
995  * ...
996  */
997 }
998
999 /**
1000  * ...
1001  */
1002 }
1003
1004 /**
1005  * ...
1006  */
1007 }
1008
1009 /**
1010  * ...
1011  */
1012 }
1013
1014 /**
1015  * ...
1016  */
1017 }
1018
1019 /**
1020  * ...
1021  */
1022 }
1023
1024 /**
1025  * ...
1026  */
1027 }
1028
1029 /**
1030  * ...
1031  */
1032 }
1033
1034 /**
1035  * ...
1036  */
1037 }
1038
1039 /**
1040  * ...
1041  */
1042 }
1043
1044 /**
1045  * ...
1046  */
1047 }
1048
1049 /**
1050  * ...
1051  */
1052 }
1053
1054 /**
1055  * ...
1056  */
1057 }
1058
1059 /**
1060  * ...
1061  */
1062 }
1063
1064 /**
1065  * ...
1066  */
1067 }
1068
1069 /**
1070  * ...
1071  */
1072 }
1073
1074 /**
1075  * ...
1076  */
1077 }
1078
1079 /**
1080  * ...
1081  */
1082 }
1083
1084 /**
1085  * ...
1086  */
1087 }
1088
1089 /**
1090  * ...
1091  */
1092 }
1093
1094 /**
1095  * ...
1096  */
1097 }
1098
1099 /**
1100  * ...
1101  */
1102 }
1103
1104 /**
1105  * ...
1106  */
1107 }
1108
1109 /**
1110  * ...
1111  */
1112 }
1113
1114 /**
1115  * ...
1116  */
1117 }
1118
1119 /**
1120  * ...
1121  */
1122 }
1123
1124 /**
1125  * ...
1126  */
1127 }
1128
1129 /**
1130  * ...
1131  */
1132 }
1133
1134 /**
1135  * ...
1136  */
1137 }
1138
1139 /**
1140  * ...
1141  */
1142 }
1143
1144 /**
1145  * ...
1146  */
1147 }
1148
1149 /**
1150  * ...
1151  */
1152 }
1153
1154 /**
1155  * ...
1156  */
1157 }
1158
1159 /**
1160  * ...
1161  */
1162 }
1163
1164 /**
1165  * ...
1166  */
1167 }
1168
1169 /**
1170  * ...
1171  */
1172 }
1173
1174 /**
1175  * ...
1176  */
1177 }
1178
1179 /**
1180  * ...
1181  */
1182 }
1183
1184 /**
1185  * ...
1186  */
1187 }
1188
1189 /**
1190  * ...
1191  */
1192 }
1193
1194 /**
1195  * ...
1196  */
1197 }
1198
1199 /**
1200  * ...
1201  */
1202 }
1203
1204 /**
1205  * ...
1206  */
1207 }
1208
1209 /**
1210  * ...
1211  */
1212 }
1213
1214 /**
1215  * ...
1216  */
1217 }
1218
1219 /**
1220  * ...
1221  */
1222 }
1223
1224 /**
1225  * ...
1226  */
1227 }
1228
1229 /**
1230  * ...
1231  */
1232 }
1233
1234 /**
1235  * ...
1236  */
1237 }
1238
1239 /**
1240  * ...
1241  */
1242 }
1243
1244 /**
1245  * ...
1246  */
1247 }
1248
1249 /**
1250  * ...
1251  */
1252 }
1253
1254 /**
1255  * ...
1256  */
1257 }
1258
1259 /**
1260  * ...
1261  */
1262 }
1263
1264 /**
1265  * ...
1266  */
1267 }
1268
1269 /**
1270  * ...
1271  */
1272 }
1273
1274 /**
1275  * ...
1276  */
1277 }
1278
1279 /**
1280  * ...
1281  */
1282 }
1283
1284 /**
1285  * ...
1286  */
1287 }
1288
1289 /**
1290  * ...
1291  */
1292 }
1293
1294 /**
1295  * ...
1296  */
1297 }
1298
1299 /**
1300  * ...
1301  */
1302 }
1303
1304 /**
1305  * ...
1306  */
1307 }
1308
1309 /**
1310  * ...
1311  */
1312 }
1313
1314 /**
1315  * ...
1316  */
1317 }
1318
1319 /**
1320  * ...
1321  */
1322 }
1323
1324 /**
1325  * ...
1326  */
1327 }
1328
1329 /**
1330  * ...
1331  */
1332 }
1333
1334 /**
1335  * ...
1336  */
1337 }
1338
1339 /**
1340  * ...
1341  */
1342 }
1343
1344 /**
1345  * ...
1346  */
1347 }
1348
1349 /**
1350  * ...
1351  */
1352 }
1353
1354 /**
1355  * ...
1356  */
1357 }
1358
1359 /**
1360  * ...
1361  */
1362 }
1363
1364 /**
1365  * ...
1366  */
1367 }
1368
1369 /**
1370  * ...
1371  */
1372 }
1373
1374 /**
1375  * ...
1376  */
1377 }
1378
1379 /**
1380  * ...
1381  */
1382 }
1383
1384 /**
1385  * ...
1386  */
1387 }
1388
1389 /**
1390  * ...
1391  */
1392 }
1393
1394 /**
1395  * ...
1396  */
1397 }
1398
1399 /**
1400  * ...
1401  */
1402 }
1403
1404 /**
1405  * ...
1406  */
1407 }
1408
1409 /**
1410  * ...
1411  */
1412 }
1413
1414 /**
1415  * ...
1416  */
1417 }
1418
1419 /**
1420  * ...
1421  */
1422 }
1423
1424 /**
1425  * ...
1426  */
1427 }
1428
1429 /**
1430  * ...
1431  */
1432 }
1433
1434 /**
1435  * ...
1436  */
1437 }
1438
1439 /**
1440  * ...
1441  */
1442 }
1443
1444 /**
1445  * ...
1446  */
1447 }
1448
1449 /**
1450  * ...
1451  */
1452 }
1453
1454 /**
1455  * ...
1456  */
1457 }
1458
1459 /**
1460  * ...
1461  */
1462 }
1463
1464 /**
1465  * ...
1466  */
1467 }
1468
1469 /**
1470  * ...
1471  */
1472 }
1473
1474 /**
1475  * ...
1476  */
1477 }
1478
1479 /**
1480  * ...
1481  */
1482 }
1483
1484 /**
1485  * ...
1486  */
1487 }
1488
1489 /**
1490  * ...
1491  */
1492 }
1493
1494 /**
1495  * ...
1496  */
1497 }
1498
1499 /**
1500  * ...
1501  */
1502 }
1503
1504 /**
1505  * ...
1506  */
1507 }
1508
1509 /**
1510  * ...
1511  */
1512 }
1513
1514 /**
1515  * ...
1516  */
1517 }
1518
1519 /**
1520  * ...
1521  */
1522 }
1523
1524 /**
1525  * ...
1526  */
1527 }
1528
1529 /**
1530  * ...
1531  */
1532 }
1533
1534 /**
1535  * ...
1536  */
1537 }
1538
1539 /**
1540  * ...
1541  */
1542 }
1543
1544 /**
1545  * ...
1546  */
1547 }
1548
1549 /**
1550  * ...
1551  */
1552 }
1553
1554 /**
1555  * ...
1556  */
1557 }
1558
1559 /**
1560  * ...
1561  */
1562 }
1563
1564 /**
1565  * ...
1566  */
1567 }
1568
1569 /**
1570  * ...
1571  */
1572 }
1573
1574 /**
1575  * ...
1576  */
1577 }
1578
1579 /**
1580  * ...
1581  */
1582 }
1583
1584 /**
1585  * ...
1586  */
1587 }
1588
1589 /**
1590  * ...
1591  */
1592 }
1593
1594 /**
1595  * ...
1596  */
1597 }
1598
1599 /**
1600  * ...
1601  */
1602 }
1603
1604 /**
1605  * ...
1606  */
1607 }
1608
1609 /**
1610  * ...
1611  */
1612 }
1613
1614 /**
1615  * ...
1616  */
1617 }
1618
1619 /**
1620  * ...
1621  */
1622 }
1623
1624 /**
1625  * ...
1626  */
1627 }
1628
1629 /**
1630  * ...
1631  */
1632 }
1633
1634 /**
1635  * ...
1636  */
1637 }
1638
1639 /**
1640  * ...
1641  */
1642 }
1643
1644 /**
1645  * ...
1646  */
1647 }
1648
1649 /**
1650  * ...
1651  */
1652 }
1653
1654 /**
1655  * ...
1656  */
1657 }
1658
1659 /**
1660  * ...
1661  */
1662 }
1663
1664 /**
1665  * ...
1666  */
1667 }
1668
1669 /**
1670  * ...
1671  */
1672 }
1673
1674 /**
1675  * ...
1676  */
1677 }
1678
1679 /**
1680  * ...
1681  */
1682 }
1683
1684 /**
1685  * ...
1686  */
1687 }
1688
1689 /**
1690  * ...
1691  */
1692 }
1693
1694 /**
1695  * ...
1696  */
1697 }
1698
1699 /**
1700  * ...
1701  */
1702 }
1703
1704 /**
1705  * ...
1706  */
1707 }
1708
1709 /**
1710  * ...
1711  */
1712 }
1713
1714 /**
1715  * ...
1716  */
1717 }
1718
1719 /**
1720  * ...
1721  */
1722 }
1723
1724 /**
1725  * ...
1726  */
1727 }
1728
1729 /**
1730  * ...
1731  */
1732 }
1733
1734 /**
1735  * ...
1736  */
1737 }
1738
1739 /**
1740  * ...
1741  */
1742 }
1743
1744 /**
1745  * ...
1746  */
1747 }
1748
1749 /**
1750  * ...
1751  */
1752 }
1753
1754 /**
1755  * ...
1756  */
1757 }
1758
1759 /**
1760  * ...
1761  */
1762 }
1763
1764 /**
1765  * ...
1766  */
1767 }
1768
1769 /**
1770  * ...
1771  */
1772 }
1773
1774 /**
1775  * ...
1776  */
1777 }
1778
1779 /**
1780  * ...
1781  */
1782 }
1783
1784 /**
1785  * ...
1786  */
1787 }
1788
1789 /**
1790  * ...
1791  */
1792 }
1793
1794 /**
1795  * ...
1796  */
1797 }
1798
1799 /**
1800  * ...
1801  */
1802 }
1803
1804 /**
1805  * ...
1806  */
1807 }
1808
1809 /**
1810  * ...
1811  */
1812 }
1813
1814 /**
1815  * ...
1816  */
1817 }
1818
1819 /**
1820  * ...
1821  */
1822 }
1823
1824 /**
1825  * ...
1826  */
1827 }
1828
1829 /**
1830  * ...
1831  */
1832 }
1833
1834 /**
1835  * ...
1836  */
1837 }
1838
1839 /**
1840  * ...
1841  */
1842 }
1843
1844 /**
1845  * ...
1846  */
1847 }
1848
1849 /**
1850  * ...
1851  */
1852 }
1853
1854 /**
1855  * ...
1856  */
1857 }
1858
1859 /**
1860  * ...
1861  */
1862 }
1863
1864 /**
1865  * ...
1866  */
1867 }
1868
1869 /**
1870  * ...
1871  */
1872 }
1873
1874 /**
1875  * ...
1876  */
1877 }
1878
1879 /**
1880  * ...
1881  */
1882 }
1883
1884 /**
1885  * ...
1886  */
1887 }
1888
1889 /**
1890  * ...
1891  */
1892 }
1893
1894 /**
1895  * ...
1896  */
1897 }
1898
1899 /**
1900  * ...
1901  */
1902 }
1903
1904 /**
1905  * ...
1906  */
1907 }
1908
1909 /**
1910  * ...
1911  */
1912 }
1913
1914 /**
1915  * ...
1916  */
1917 }
1918
1919 /**
1920  * ...
1921  */
1922 }
1923
1924 /**
1925  * ...
1926  */
1927 }
1928
1929 /**
1930  * ...
1931  */
1932 }
1933
1934 /**
1935  * ...
1936  */
1937 }
1938
1939 /**
1940  * ...
1941  */
1942 }
1943
1944 /**
1945  * ...
1946  */
1947 }
1948
1949 /**
1950  * ...
1951  */
1952 }
1953
1954 /**
1955  * ...
1956  */
1957 }
1958
1959 /**
1960  * ...
1961  */
1962 }
1963
1964 /**
1965  * ...
1966  */
1967 }
1968
1969 /**
1970  * ...
1971  */
1972 }
1973
1974 /**
1975  * ...
1976  */
1977 }
1978
1979 /**
1980  * ...
1981  */
1982 }
1983
1984 /**
1985  * ...
1986  */
1987 }
1988
1989 /**
1990  * ...
1991  */
1992 }
1993
1994 /**
1995  * ...
1996  */
1997 }
1998
1999 /**
2000  * ...
2001  */
2002 }
2003
2004 /**
2005  * ...
2006  */
2007 }
2008
2009 /**
2010  * ...
2011  */
2012 }
2013
2014 /**
2015  * ...
2016  */
2017 }
2018
2019 /**
2020  * ...
2021  */
2022 }
2023
2024 /**
2025  * ...
2026  */
2027 }
2028
2029 /**
2030  * ...
2031  */
2032 }
2033
2034 /**
2035  * ...
2036  */
2037 }
2038
2039 /**
2040  * ...
2041  */
2042 }
2043
2044 /**
2045  * ...
2046  */
2047 }
2048
2049 /**
2050  * ...
2051  */
2052 }
2053
2054 /**
2055  * ...
2056  */
2057 }
2058
2059 /**
2060  * ...
2061  */
2062 }
2063
2064 /**
2065  * ...
2066  */
2067 }
2068
2069 /**
2070  * ...
2071  */
2072 }
2073
2074 /**
2075  * ...
2076  */
2077 }
2078
2079 /**
2080  * ...
2081  */
2082 }
2083
2084 /**
2085  * ...
2086  */
2087 }
2088
2089 /**
2090  * ...
2091  */
2092 }
2093
2094 /**
2095  * ...
2096  */
2097 }
2098
2099 /**
2100  * ...
2101  */
2102 }
2103
2104 /**
2105  * ...
2106  */
2107 }
2108
2109 /**
2110  * ...
2111  */
2112 }
2113
2114 /**
2115  * ...
2116  */
2117 }
2118
2119 /**
2120  * ...
2121  */
2122 }
2123
2124 /**
2125  * ...
2126  */
2127 }
2128
2129 /**
2130  * ...
2131  */
2132 }
2133
2134 /**
2135  * ...
2136  */
2137 }
2138
2139 /**
2140  * ...
2141  */
2142 }
2143
2144 /**
2145  * ...
2146  */
2147 }
2148
2149 /**
2150  * ...
2151  */
2152 }
2153
2154 /**
2155  * ...
2156  */
2157 }
2158
2159 /**
2160  * ...
2161  */
2162 }
2163
2164 /**
2165  * ...
2166  */
```

Übung 3

```

241         }
242     }
243     }.execute();
244 }
245 }
246
247 /**
248  * Resets the given route.
249  *
250  * @param model the route to reset
251  */
252 private void resetRoute(final RouteModel model) {
253     if (checkAndDisplayAvailableNetwork()) {
254         new AsyncTask<Object, Object, AsyncTaskResult<Boolean>>() {
255             @Override
256             protected AsyncTaskResult<Boolean> doInBackground(Object... params) {
257                 Boolean result = null;
258                 Exception exception = null;
259                 try {
260                     if (model != null) {
261                         result = viewModel.resetRoute(model);
262                     } else {
263                         result = viewModel.resetAllRoutes();
264                     }
265                 } catch (Exception e) {
266                     exception = e;
267                 }
268
269                 return new AsyncTaskResult<Boolean>(result, exception);
270             }
271
272             @Override
273             protected void onPreExecute() {
274                 super.onPreExecute();
275                 openProgressDialog(R.string.progress_resetting_routes);
276             }
277
278             @Override
279             protected void onPostExecute(AsyncTaskResult<Boolean> result) {
280                 super.onPostExecute(result);
281                 closeProgressDialog();
282                 // Error occurred
283                 if (result.exception != null) {
284                     // ServiceException occurred
285                     if (result.exception instanceof ServiceException) {
286                         handleServiceException(((ServiceException) result.exception));
287                     } else {
288                         Toast.makeText(RouteActivity.this, R.string.error_unknown,
289                                     ↳ Toast.LENGTH_LONG).show();
290                     }
291                 }
292                 // Reset failed on rest method
293                 else if (!result.result) {
294                     Toast.makeText(RouteActivity.this, R.string.error_route_reset_failed,
295                                 ↳ Toast.LENGTH_LONG).show();
296                 }
297                 // Reset ok
298                 else {
299                     prepareView();
300                 }
301             }
302         }.execute();
303     }
304 }

```

Übung 3

```

302     }
303
304     /**
305      * Fires an intent which redirects teh user to the checkpoint activity.
306      *
307      * @param model the route to serialize to the checkpoint activity
308      */
309     private void goToCheckpointActivity(RouteModel model) {
310         Intent intent = new Intent(RouteActivity.this, CheckpointActivity.class);
311         intent.putExtra(AmazingRaceApplication.INTENT_EXTRA_ROUTE, model);
312         startActivity(intent);
313     }
314     //endregion
315
316     //region Listener
317
318     /**
319      * Clcik listener which redirects to the checkpoint activity
320      *
321      * @param parent the adapter
322      * @param view the view
323      * @param position the item position
324      * @param id the view id
325      */
326     @Override
327     public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
328         goToCheckpointActivity(((RouteModel) parent.getItemAtPosition(position)));
329     }
330     //endregion
331 }

```

Listing 6: CheckpointActivity.java

```

1 package at.fh.ooe.moc5.amazingrace.activity;
2
3 import android.content.DialogInterface;
4 import android.content.Intent;
5 import android.graphics.Color;
6 import android.os.AsyncTask;
7 import android.os.Bundle;
8 import android.view.Menu;
9 import android.view.MenuItem;
10 import android.view.View;
11 import android.widget.Button;
12 import android.widget.EditText;
13 import android.widget.LinearLayout;
14 import android.widget.TextView;
15 import android.widget.Toast;
16
17 import com.google.android.gms.maps.CameraUpdateFactory;
18 import com.google.android.gms.maps.GoogleMap;
19 import com.google.android.gms.maps.MapFragment;
20 import com.google.android.gms.maps.OnMapReadyCallback;
21 import com.google.android.gms.maps.model.BitmapDescriptorFactory;
22 import com.google.android.gms.maps.model.LatLng;
23 import com.google.android.gms.maps.model.MarkerOptions;
24 import com.google.android.gms.maps.model.PolylineOptions;
25
26 import at.fh.ooe.moc5.amazingrace.AmazingRaceApplication;
27 import at.fh.ooe.moc5.amazingrace.R;
28 import at.fh.ooe.moc5.amazingrace.model.json.CheckpointModel;
29 import at.fh.ooe.moc5.amazingrace.model.json.RouteModel;
30 import at.fh.ooe.moc5.amazingrace.model.task.AsyncTaskResult;
31 import at.fh.ooe.moc5.amazingrace.model.view.CheckpointViewModel;
32 import at.fh.ooe.moc5.amazingrace.service.ServiceException;
33 import at.fh.ooe.moc5.amazingrace.util.DialogUtil;
34 import at.fh.ooe.moc5.amazingrace.watcher.AnswerButtonTextWatcher;
35 import at.fh.ooe.moc5.amazingrace.watcher.CheckpointViewModelBindingTextWatcher;
36
37 public class CheckpointActivity extends AppCompatActivity<CheckpointViewModel> implements
    ↳ View.OnClickListener, DialogInterface.OnClickListener {
38
39     //region Lifecycle Methods
40
41     /**
42      * Prepares the view model depending on the intent serialized route model
43      *
44      * @param savedInstanceState the saved bundle instance
45      */
46     @Override
47     protected void onCreate(Bundle savedInstanceState) {
48         super.onCreate(savedInstanceState);
49         setContentView(R.layout.activity_checkpoint);
50         if (application.getLoggedUser() != null) {
51             final Intent intent = getIntent();
52             if (intent.getExtras().containsKey(AmazingRaceApplication.INTENT_EXTRA_ROUTE)) {
53                 RouteModel route = (RouteModel)
                    ↳ intent.getExtras().get(AmazingRaceApplication.INTENT_EXTRA_ROUTE);
54                 viewModel = new CheckpointViewModel(application.getLoggedUser(), route);
55             }
56         }
57     }
58

```

Übung 3

```

59  /**
60   * Prepares this activity views if view mdoel is valid.
61   */
62  @Override
63  protected void onResume() {
64      super.onResume();
65      if (validViewModel) {
66          prepareView();
67      }
68  }
69
70  /**
71   * Creates the options menu items
72   *
73   * @param menu the menu to add items too
74   * @return true
75   */
76  @Override
77  public boolean onCreateOptionsMenu(Menu menu) {
78      menu.add(MenuGroup.OPTIONS.value, MenuId.RESET.value, 0, R.string.action_reset);
79      menu.add(MenuGroup.OPTIONS.value, MenuId.CLOSE.value, 0, R.string.action_close);
80      return Boolean.TRUE;
81  }
82
83  /**
84   * Handles an options menu item select
85   *
86   * @param item the selected menu item
87   * @return true if items has been handled
88   */
89  @Override
90  public boolean onOptionsItemSelected(MenuItem item) {
91      switch (MenuId.getMenuIdForValue(item.getItemId())) {
92          case RESET:
93              resetRoute();
94              return Boolean.TRUE;
95          case CLOSE:
96              openCloseApplicationDialog();
97              return Boolean.TRUE;
98          default:
99              return super.onOptionsItemSelected(item);
100      }
101  }
102  //endregion
103
104  //region Helper
105
106  /**
107   * Prepares this acitivity views.
108   */
109  public void prepareView() {
110      // set default container visibility
111      findViewById(R.id.routeCheckpointContainer).setVisibility(View.VISIBLE);
112      findViewById(R.id.routeCheckpointMapContainer).setVisibility(View.VISIBLE);
113      findViewById(R.id.routeCheckpointListContainer).setVisibility(View.GONE);
114
115      // prepare list view of visited checkpoints
116      // ListView is not used because of outer wrapping ScrollView
117      LinearLayout listViewReplacement = (LinearLayout)
118          ↳ findViewById(R.id.routeCheckpointListView);
119      listViewReplacement.removeAllViews();
120      listViewReplacement.setVisibility(View.VISIBLE);
121      for (int i = 0; i < viewModel.getVisitedCheckpoints().size(); i++) {

```

Übung 3

```

121     CheckpointModel model = viewModel.getRoute().getVisitedCheckpoints().get(i);
122     View view = View.inflate(CheckpointActivity.this, R.layout.view_checkpoint_item,
        ↪ null);
123     ((TextView) view.findViewById(R.id.visitedCheckpointNameLabel)).setText((i + 1) + ". "
        ↪ + model.getName());
124     listViewReplacement.addView(view);
125 }
126 // Add next checkpoint to list with open marker
127 if (viewModel.getNextCheckpoint() != null) {
128     final CheckpointModel model = viewModel.getNextCheckpoint();
129     View view = View.inflate(CheckpointActivity.this, R.layout.view_checkpoint_item,
        ↪ null);
130     ((TextView) view.findViewById(R.id.visitedCheckpointNameLabel)).setText(
131         new StringBuilder(model.getName()).append("
        ↪ (").append(getText(R.string.open)).append(")").toString()
132     );
133     listViewReplacement.addView(view);
134 }
135
136 // route already finished
137 if (viewModel.getNextCheckpoint() == null) {
138     findViewById(R.id.routeCheckpointContainerLabel).setVisibility(View.GONE);
139     findViewById(R.id.routeCheckpointContainer).setVisibility(View.GONE);
140     findViewById(R.id.routeCheckpointListContainer).setVisibility(View.VISIBLE);
141 }
142 // checkpoints still open
143 else {
144     Button answerButton = (Button) findViewById(R.id.answerBtn);
145     answerButton.setEnabled(Boolean.FALSE);
146     answerButton.setOnClickListener(this);
147
148     ((TextView) findViewById(R.id.nextCheckpointTxt))
149         .setText(viewModel.getNextCheckpoint().getName());
150     ((TextView) findViewById(R.id.nextCheckpointHintTxt))
151         .setText(viewModel.getNextCheckpoint().getHint());
152
153     EditText answer = (EditText) findViewById(R.id.hintAnswerEdTxt);
154     answer.setText("");
155     answer.addTextChangedListener(new CheckpointViewModelBindingTextWatcher(viewModel));
156     answer.addTextChangedListener(new AnswerButtonTextWatcher(this, viewModel));
157 }
158
159 // Prepare maps
160 MapFragment mapFragment = (MapFragment)
161     ↪ getFragmentManager().findFragmentById(R.id.routeCheckpointMap);
162 mapFragment.getMapAsync(new OnMapReadyCallback() {
163     @Override
164     public void onMapReady(GoogleMap map) {
165         // clear former set map properties
166         map.clear();
167         // enable ui functions
168         map.getUiSettings().setZoomControlsEnabled(Boolean.TRUE);
169         map.getUiSettings().setAllGesturesEnabled(Boolean.TRUE);
170         map.getUiSettings().setCompassEnabled(Boolean.TRUE);
171         LatLng zoomLocation = null;
172         PolylineOptions lineOptions = null;
173         // Get visited points
174         if (!viewModel.getRoute().getVisitedCheckpoints().isEmpty()) {
175             lineOptions = new PolylineOptions();
176             for (int i = 0; i < viewModel.getRoute().getVisitedCheckpoints().size(); i++)
177                 ↪ {
178                     CheckpointModel model =
179                         ↪ viewModel.getRoute().getVisitedCheckpoints().get(i);

```

Übung 3

```

177         LatLng location = new LatLng(model.getLatitude(), model.getLongitude());
178         map.addMarker(new MarkerOptions().position(location)
179             .title((i + 1) + ". " + model.getName())
180             .icon(
181                 BitmapDescriptorFactory.defaultMarker(
182                     BitmapDescriptorFactory.HUE_GREEN
183                 )
184             )
185             .showInfoWindow();
186         lineOptions.add(location);
187         // Remember last for focusing on it
188         if (i == (viewModel.getVisitedCheckpoints().size() - 1)) {
189             zoomLocation = location;
190         }
191     }
192     map.addPolyline(lineOptions.color(Color.GREEN));
193 }
194 // Add next checkpoint which is not part of list and gets red line from it to the
195 //   ↪ former marker
196 if (viewModel.getRoute().getNextCheckpoint() != null) {
197     zoomLocation = new
198         ↪ LatLng(viewModel.getRoute().getNextCheckpoint().getLatitude(),
199         ↪ viewModel.getRoute().getNextCheckpoint().getLongitude());
200     map.addMarker(
201         new MarkerOptions()
202             .position(zoomLocation)
203             .title(viewModel.getNextCheckpoint().getName())
204             .icon(BitmapDescriptorFactory
205                 .defaultMarker(BitmapDescriptorFactory.HUE_RED))
206     ).showInfoWindow();
207     // If visited checkpoints exist
208     if (!viewModel.getRoute().getVisitedCheckpoints().isEmpty()) {
209         map.addPolyline(new PolylineOptions().add(
210             lineOptions.getPoints().get(lineOptions.getPoints().size()
211                 ↪ - 1), zoomLocation).color(Color.RED)
212         );
213     }
214 }
215 // move camera to last or next checkpoint
216 map.moveCamera(CameraUpdateFactory.newLatLngZoom(zoomLocation, 17));
217 }
218 });
219 //endregion
220 //region Listeners
221 /**
222  * Toggles the visibility if one container header gets clicked
223  *
224  * @param view the clicked view
225  */
226 public void toggleVisibility(View view) {
227     View checkpointListContainer = findViewById(R.id.routeCheckpointListContainer);
228     View checkpointMapContainer = findViewById(R.id.routeCheckpointMapContainer);
229
230     // Always references to parent LinearLayout so not all clickable children need to be
231     //   ↪ handled here
232     switch (((View) view.getParent()).getId()) {
233         case R.id.routeCheckpointListContainerLabel:
234             checkpointListContainer.setVisibility((View.VISIBLE ==
235                 ↪ checkpointListContainer.getVisibility()) ? View.GONE : View.VISIBLE);
236             break;

```


Übung 3

```

234         case R.id.routeCheckpointMapContainerLabel:
235             checkpointMapContainer.setVisibility((View.VISIBLE ==
                ↳ checkpointMapContainer.getVisibility()) ? View.GONE : View.VISIBLE);
236             break;
237     }
238 }
239
240 /**
241  * Handles the button clicks by redirecting to proper method.
242  *
243  * @param v the clicked view
244  */
245 @Override
246 public void onClick(View v) {
247     switch (v.getId()) {
248         case R.id.answerBtn:
249             onAnswerButtonClick((Button) v);
250             break;
251     }
252 }
253
254 @Override
255 public void onClick(DialogInterface dialog, int which) {
256     dialog.dismiss();
257 }
258 //endregion
259
260 /**
261  * Handles the answer button click
262  *
263  * @param button the clicked answer button
264  */
265 private void onAnswerButtonClick(Button button) {
266     if (checkAndDisplayAvailableNetwork()) {
267         new AsyncTask<Object, Object, AsyncTaskResult<Boolean>>() {
268             @Override
269             protected AsyncTaskResult<Boolean> doInBackground(Object... params) {
270                 Boolean result = null;
271                 Exception exception = null;
272                 try {
273                     result = viewModel.validateSecret();
274                 } catch (Exception e) {
275                     exception = e;
276                 }
277
278                 return new AsyncTaskResult<Boolean>(result, exception);
279             }
280
281             @Override
282             protected void onPreExecute() {
283                 super.onPreExecute();
284                 openProgressDialog(R.string.progress_validating_secret);
285             }
286
287             @Override
288             protected void onPostExecute(AsyncTaskResult<Boolean> result) {
289                 super.onPostExecute(result);
290                 closeProgressDialog();
291                 if (result.exception != null) {
292                     // ServiceException occurred
293                     if (result.exception instanceof ServiceException) {
294                         handleServiceException(((ServiceException) result.exception));
295                     } else {

```

Übung 3

```

326         Toast.makeText(CheckpointActivity.this, R.string.error_unknown,
327             ↪ Toast.LENGTH_LONG).show();
328     }
329     } else if (result.result) {
330         reloadRoute(Boolean.TRUE);
331     } else {
332         Toast.makeText(CheckpointActivity.this, R.string.info_secret_wrong,
333             ↪ Toast.LENGTH_LONG).show();
334     }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }

/**
 * Reloads the route
 */
private void reloadRoute(final boolean checkIfFinished) {
    if (checkAndDisplayAvailableNetwork()) {
        new AsyncTask<Object, Object, AsyncTaskResult<Boolean>>() {
            @Override
            protected AsyncTaskResult<Boolean> doInBackground(Object... params) {
                Boolean found = null;
                Exception exception = null;
                try {
                    found = viewModel.reloadRoute();
                } catch (Exception e) {
                    exception = e;
                }
                return new AsyncTaskResult<Boolean>(found, exception);
            }

            @Override
            protected void onPostExecute(AsyncTaskResult<Boolean> result) {
                super.onPostExecute(result);
                closeProgressDialog();
                // Exception occurred
                if (result.exception != null) {
                    if (result.exception instanceof ServiceException) {
                        handleServiceException(((ServiceException) result.exception));
                    } else {
                        Toast.makeText(CheckpointActivity.this, R.string.error_unknown,
                            ↪ Toast.LENGTH_SHORT).show();
                    }
                }
                // Route found
                else if (result.result) {
                    prepareView();
                    if (checkIfFinished) {
                        if (viewModel.getNextCheckpoint() != null) {
                            Toast.makeText(CheckpointActivity.this, R.string.info_secret_ok,
                                ↪ Toast.LENGTH_LONG).show();
                        } else {
                            DialogUtil.createCustomContentDialog(CheckpointActivity.this,
                                ↪ R.layout.view_congratulations_dialog,
                                ↪ R.string.dialog_title_info, CheckpointActivity.this).show();
                        }
                    }
                }
                // Route not found, go back to routes activity
                else {
                    startActivity(new Intent(CheckpointActivity.this, RouteActivity.class));
                    finish();
                }
            }
        }
    }
}

```

Übung 3

```

353         }
354     }
355
356     @Override
357     protected void onPreExecute() {
358         super.onPreExecute();
359         openProgressDialog(R.string.progress_updating_route);
360     }
361     }.execute();
362 }
363 }
364
365 /**
366  * Resets the current route.
367  */
368 private void resetRoute() {
369     if (checkAndDisplayAvailableNetwork()) {
370         new AsyncTask<Object, Object, AsyncTaskResult<Boolean>>() {
371             @Override
372             protected AsyncTaskResult<Boolean> doInBackground(Object... params) {
373                 Boolean result = null;
374                 Exception exception = null;
375                 try {
376                     result = viewModel.resetRoute();
377                 } catch (Exception e) {
378                     exception = e;
379                 }
380
381                 return new AsyncTaskResult<Boolean>(result, exception);
382             }
383
384             @Override
385             protected void onPreExecute() {
386                 super.onPreExecute();
387                 openProgressDialog(R.string.progress_resetting_routes);
388             }
389
390             @Override
391             protected void onPostExecute(AsyncTaskResult<Boolean> result) {
392                 super.onPostExecute(result);
393                 closeProgressDialog();
394                 // Error occurred
395                 if (result.exception != null) {
396                     // ServiceException occurred
397                     if (result.exception instanceof ServiceException) {
398                         handleServiceException(((ServiceException) result.exception));
399                     } else {
400                         Toast.makeText(CheckpointActivity.this, R.string.error_unknown,
401                                     ↵ Toast.LENGTH_LONG).show();
402                     }
403                 }
404                 // Reset failed on rest method
405                 else if (!result.result) {
406                     Toast.makeText(CheckpointActivity.this, R.string.error_route_reset_failed,
407                                 ↵ Toast.LENGTH_LONG).show();
408                 }
409                 // Reset ok
410                 else {
411                     reloadRoute(Boolean.FALSE);
412                 }
413             }
414         }.execute();
415     }

```

Übung 3

```
414     }  
415     //endregion  
416 }
```

Übung 3

1.2.3 Adapter

Listing 7: RouteArrayAdapter.java

```

1  package at.fh.ooe.moc5.amazingrace.adapter;
2
3  import android.content.Context;
4  import android.view.View;
5  import android.view.ViewGroup;
6  import android.widget.AdapterView;
7  import android.widget.TextView;
8
9  import at.fh.ooe.moc5.amazingrace.R;
10 import at.fh.ooe.moc5.amazingrace.model.json.RouteModel;
11
12 /**
13  * Created by Thomas on 12/25/2015.
14  * The adapter for the route items
15  */
16 public class RouteArrayAdapter extends ArrayAdapter<RouteModel> {
17
18     public RouteArrayAdapter(Context ctx) {
19         super(ctx, 0);
20     }
21
22     @Override
23     public View getView(int position, View convertView, ViewGroup parent) {
24         if (convertView == null) {
25             convertView = View.inflate(getContext(), R.layout.view_route_item, null);
26         }
27
28         RouteModel route = getItem(position);
29         ((TextView)
30             ↪ convertView.findViewById(R.id.routeNameLabel)).setText(route.toItemString(getContext().getString(R.
31             ↪ getContext().getString(R.string.done)));
32
33         return convertView;
34     }
35 }

```

Übung 3

1.2.4 JSON-Models

Listing 8: CheckpointModel.java

```

1 package at.fh.ooe.moc5.amazingrace.model.json;
2
3 import com.google.gson.annotations.SerializedName;
4
5 import java.io.Serializable;
6 import java.util.Objects;
7
8 /**
9  * Created by Thomas on 12/25/2015.
10  */
11 public class CheckpointModel implements Serializable {
12
13     @SerializedName("Id")
14     private String id;
15     @SerializedName("Number")
16     private int number;
17     @SerializedName("Name")
18     private String name;
19     @SerializedName("Hint")
20     private String hint;
21     @SerializedName("Latitude")
22     private double latitude;
23     @SerializedName("Longitude")
24     private double longitude;
25     private transient boolean unvisited = Boolean.FALSE;
26
27     public CheckpointModel() {
28     }
29
30     //region Getter and Setter
31     public String getId() {
32         return id;
33     }
34
35     public void setId(String id) {
36         this.id = id;
37     }
38
39     public int getNumber() {
40         return number;
41     }
42
43     public void setNumber(int number) {
44         this.number = number;
45     }
46
47     public String getName() {
48         return name;
49     }
50
51     public void setName(String name) {
52         this.name = name;
53     }
54
55     public String getHint() {
56         return hint;
57     }
58
59     public void setHint(String hint) {

```

Übung 3

```

60         this.hint = hint;
61     }
62
63     public double getLatitude() {
64         return latitude;
65     }
66
67     public void setLatitude(double latitude) {
68         this.latitude = latitude;
69     }
70
71     public double getLongitude() {
72         return longitude;
73     }
74
75     public void setLongitude(double longitude) {
76         this.longitude = longitude;
77     }
78
79     public boolean isUnvisited() {
80         return unvisited;
81     }
82
83     public void setUnvisited(boolean unvisited) {
84         this.unvisited = unvisited;
85     }
86     //endregion
87
88     @Override
89     public boolean equals(Object o) {
90         if (this == o) return true;
91         if (o == null || getClass() != o.getClass()) return false;
92         CheckpointModel that = (CheckpointModel) o;
93         return Objects.equals(id, that.id);
94     }
95
96     @Override
97     public int hashCode() {
98         return Objects.hash(id);
99     }
100 }

```

Listing 9: CheckpointRequestModel.java

```
1 package at.fh.ooe.moc5.amazingrace.model.json;
2
3 import com.google.gson.annotations.SerializedName;
4
5 import java.io.Serializable;
6 import java.util.Objects;
7
8 /**
9  * Created by Thomas on 12/25/2015.
10 */
11 public class CheckpointRequestModel extends CredentialsRequestModel implements Serializable {
12
13     @SerializedName("CheckpointId")
14     public final String checkpointId;
15     @SerializedName("Secret")
16     public final String secret;
17
18     public CheckpointRequestModel(String username, String password, String checkpointId, String
19         ↪ secret) {
20         super(username, password);
21         this.checkpointId = checkpointId;
22         this.secret = secret;
23     }
24
25     @Override
26     public boolean equals(Object o) {
27         if (this == o) return true;
28         if (o == null || getClass() != o.getClass()) return false;
29         if (!super.equals(o)) return false;
30         CheckpointRequestModel that = (CheckpointRequestModel) o;
31         return Objects.equals(checkpointId, that.checkpointId);
32     }
33
34     @Override
35     public int hashCode() {
36         return Objects.hash(super.hashCode(), checkpointId);
37     }
38 }
```


Listing 10: CredentialsRequestModel.java

```

1 package at.fh.ooe.moc5.amazingrace.model.json;
2
3 import com.google.gson.annotations.SerializedName;
4
5 import java.io.Serializable;
6 import java.util.Objects;
7
8 /**
9  * Created by Thomas on 12/25/2015.
10 */
11 public class CredentialsRequestModel implements Serializable {
12
13     @SerializedName("UserName")
14     public final String userName;
15     @SerializedName("Password")
16     public final String password;
17
18     public CredentialsRequestModel(String userName, String password) {
19         this.userName = userName;
20         this.password = password;
21     }
22
23     public String toQueryString() {
24         return String.format("userName=%s&password=%s", userName, password);
25     }
26
27     @Override
28     public boolean equals(Object o) {
29         if (this == o) return true;
30         if (o == null || getClass() != o.getClass()) return false;
31         CredentialsRequestModel that = (CredentialsRequestModel) o;
32         return Objects.equals(userName, that.userName);
33     }
34
35     @Override
36     public int hashCode() {
37         return Objects.hash(userName);
38     }
39 }

```

Listing 11: RouteModel.java

```

1  package at.fh.ooe.moc5.amazingrace.model.json;
2
3  import com.google.gson.annotations.SerializedName;
4
5  import java.io.Serializable;
6  import java.util.Comparator;
7  import java.util.List;
8  import java.util.Objects;
9
10 /**
11  * Created by Thomas on 12/25/2015.
12  */
13 public class RouteModel implements Serializable {
14
15     @SerializedName("Id")
16     private String id;
17     @SerializedName("Name")
18     private String name;
19     @SerializedName("VisitedCheckpoints")
20     private List<CheckpointModel> visitedCheckpoints;
21     @SerializedName("NextCheckpoint")
22     private CheckpointModel nextCheckpoint;
23
24     public RouteModel() {
25     }
26
27     //region Getter and Setter
28     public String getId() {
29         return id;
30     }
31
32     public void setId(String id) {
33         this.id = id;
34     }
35
36     public String getName() {
37         return name;
38     }
39
40     public void setName(String name) {
41         this.name = name;
42     }
43
44     public List<CheckpointModel> getVisitedCheckpoints() {
45         return visitedCheckpoints;
46     }
47
48     public void setVisitedCheckpoints(List<CheckpointModel> visitedCheckpoints) {
49         this.visitedCheckpoints = visitedCheckpoints;
50     }
51
52     public CheckpointModel getNextCheckpoint() {
53         return nextCheckpoint;
54     }
55
56     public void setNextCheckpoint(CheckpointModel nextCheckpoint) {
57         this.nextCheckpoint = nextCheckpoint;
58     }
59     //endregion
60

```

Übung 3

```

61 //region Utilities
62 public boolean isDone() {
63     return nextCheckpoint == null;
64 }
65 //endregion
66
67 public String toItemString(final String finishedString, final String doneString) {
68     Objects.requireNonNull(finishedString);
69     Objects.requireNonNull(doneString);
70
71     final StringBuilder sb = new StringBuilder(name).append(" (");
72     if (isDone()) {
73         sb.append(finishedString);
74     } else {
75         sb.append(visitedCheckpoints.size()).append("x").append(doneString);
76     }
77     sb.append(")");
78     return sb.toString();
79 }
80
81 @Override
82 public boolean equals(Object o) {
83     if (this == o) return true;
84     if (o == null || getClass() != o.getClass()) return false;
85     RouteModel that = (RouteModel) o;
86     return Objects.equals(id, that.id);
87 }
88
89 @Override
90 public int hashCode() {
91     return Objects.hash(id);
92 }
93 }

```

Listing 12: RouteRequestModel.java

```
1 package at.fh.ooe.moc5.amazingrace.model.json;
2
3 import com.google.gson.annotations.SerializedName;
4
5 import java.io.Serializable;
6 import java.util.Objects;
7
8 /**
9  * Created by Thomas on 12/25/2015.
10 */
11 public class RouteRequestModel extends CredentialsRequestModel {
12
13     @SerializedName("RouteId")
14     public final String routeId;
15
16     public RouteRequestModel(CredentialsRequestModel model, String routeId) {
17         this(model.userName, model.password, routeId);
18     }
19
20     public RouteRequestModel(String username, String password, String routeId) {
21         super(username, password);
22         this.routeId = routeId;
23     }
24
25     @Override
26     public boolean equals(Object o) {
27         if (this == o) return true;
28         if (o == null || getClass() != o.getClass()) return false;
29         if (!super.equals(o)) return false;
30         RouteRequestModel that = (RouteRequestModel) o;
31         return Objects.equals(routeId, that.routeId);
32     }
33
34     @Override
35     public int hashCode() {
36         return Objects.hash(super.hashCode(), routeId);
37     }
38 }
```

1.2.5 Task-Models

Listing 13: AsyncTaskResult.java

```
1 package at.fh.ooe.moc5.amazingrace.model.task;
2
3 /**
4  * Created by Thomas on 12/25/2015.
5  * This class represents an async task result which allows handling of any occurred exception
6  * in the postExecute method instead of the doInBackground where we are not in the UI Thread.
7  */
8 public class AsyncTaskResult<T> {
9
10     public final T result;
11     public final Exception exception;
12
13     public AsyncTaskResult(T result, Exception exception) {
14         this.result = result;
15         this.exception = exception;
16     }
17 }
```

Übung 3

1.2.6 View-Models

Listing 14: LoginViewModel.java

```

1 package at.fh.ooe.moc5.amazingrace.model.view;
2
3 import org.apache.commons.lang3.StringUtils;
4
5 import java.io.Serializable;
6
7 import at.fh.ooe.moc5.amazingrace.model.json.CredentialsRequestModel;
8 import at.fh.ooe.moc5.amazingrace.service.ServiceProxy;
9 import at.fh.ooe.moc5.amazingrace.service.ServiceException;
10 import at.fh.ooe.moc5.amazingrace.service.ServiceProxyFactory;
11
12 /**
13  * Created by Thomas on 12/24/2015.
14  */
15 public class LoginViewModel implements Serializable {
16
17     private String username;
18     private String password;
19
20     private final ServiceProxy restProxy;
21
22     public LoginViewModel() {
23         restProxy = ServiceProxyFactory.createServiceProxy();
24     }
25
26     /**
27      * Answers the question if this model is valid whic its is if a username and password are set.
28      *
29      * @return true if valid, false otherwise
30      */
31     public boolean isValid() {
32         return ((username != null) && (!StringUtils.isEmpty(username))) && ((password != null) &&
33             ↪ (!StringUtils.isEmpty(password)));
34     }
35
36     //region Actions
37
38     /**
39      * Logs the user in by validating the user credentials via the proxy service.
40      *
41      * @return true if username password are valid, false otherwise
42      * @throws ServiceException if an rproxy service exception occurred
43      */
44     public UserContextModel loginAction() throws ServiceException {
45         final CredentialsRequestModel model = new CredentialsRequestModel(username, password);
46         final boolean isValidCredentials = restProxy.validateCredentials(model);
47         if (isValidCredentials) {
48             return new UserContextModel(model);
49         }
50         return null;
51     }
52
53     /**
54      * Resets this model by resetting the username and password
55      */
56     public void reset() {
57         username = null;
58         password = null;

```

Übung 3

```
59     }
60
61     //endregion
62     //region Getter and Setter
63     public void setUsername(String username) {
64         this.username = username;
65     }
66
67     public void setPassword(String password) {
68         this.password = password;
69     }
70     //endregion
71 }
```

Übung 3

Listing 15: RoutesViewModel.java

```

1 package at.fh.ooe.moc5.amazingrace.model.view;
2
3 import java.io.Serializable;
4 import java.util.List;
5 import java.util.Objects;
6
7 import at.fh.ooe.moc5.amazingrace.model.json.RouteModel;
8 import at.fh.ooe.moc5.amazingrace.model.json.RouteRequestModel;
9 import at.fh.ooe.moc5.amazingrace.service.ServiceProxy;
10 import at.fh.ooe.moc5.amazingrace.service.ServiceProxyFactory;
11 import at.fh.ooe.moc5.amazingrace.service.ServiceException;
12
13 /**
14  * Created by Thomas on 12/25/2015.
15  */
16 public class RoutesViewModel implements Serializable {
17
18     private UserContextModel userContext;
19     private RouteModel selectedRoute;
20     private List<RouteModel> routes;
21
22     private ServiceProxy proxy;
23
24     public RoutesViewModel(UserContextModel userContext) {
25         Objects.requireNonNull(userContext, "UserContext must not be null");
26         this.userContext = userContext;
27         proxy = ServiceProxyFactory.createServiceProxy();
28     }
29
30     //region Actions
31
32     /**
33      * Loads the routes
34      *
35      * @return the found routes
36      * @throws ServiceException if the proxy service throw an exception
37      */
38     public List<RouteModel> loadRoutes() throws ServiceException {
39         routes = proxy.loadRoutes(userContext.getCredentialsModel());
40         return routes;
41     }
42
43     /**
44      * Resets the given route
45      *
46      * @param model the route to reset
47      * @return true if reset, false otherwise
48      * @throws ServiceException if the proxy service throw an exception
49      */
50     public boolean resetRoute(RouteModel model) throws ServiceException {
51         return proxy.resetRoute(new RouteRequestModel(userContext.getCredentialsModel(),
52             ↪ model.getId()));
53     }
54
55     /**
56      * Resets all routes
57      *
58      * @return true if reset, false otherwise
59      * @throws ServiceException if the proxy service throw an exception
60      */

```


Übung 3

```
60     public boolean resetAllRoutes() throws ServiceException {
61         return proxy.resetAllRoutes(userContext.getCredentialsModel());
62     }
63     //endregion
64
65     //region Getter and Setter
66     public RouteModel getSelectedRoute() {
67         return selectedRoute;
68     }
69
70     public void setSelectedRoute(RouteModel selectedRoute) {
71         this.selectedRoute = selectedRoute;
72     }
73
74     public List<RouteModel> getRoutes() {
75         return routes;
76     }
77     //endregion
78 }
```

Listing 16: UserContextModel.java

```

1 package at.fh.ooe.moc5.amazingrace.model.view;
2
3 import java.io.Serializable;
4 import java.util.Calendar;
5 import java.util.Objects;
6
7 import at.fh.ooe.moc5.amazingrace.model.json.CredentialsRequestModel;
8
9 /**
10  * Created by Thomas on 12/24/2015.
11  * This model represents an logged user for the application.
12  */
13 public class UserContextModel implements Serializable {
14
15     private final CredentialsRequestModel model;
16     private final Calendar loginDate;
17
18     public UserContextModel(CredentialsRequestModel model) {
19         Objects.requireNonNull(model);
20         this.model = model;
21         this.loginDate = Calendar.getInstance();
22     }
23
24     //region Getter and Setter
25     public CredentialsRequestModel getCredentialsModel() {
26         return model;
27     }
28
29     public Calendar getLoginDate() {
30         return loginDate;
31     }
32     //endregion
33 }

```

Listing 17: CheckpointViewModel.java

```

1 package at.fh.ooe.moc5.amazingrace.model.view;
2
3 import java.io.Serializable;
4 import java.util.List;
5 import java.util.Objects;
6
7 import at.fh.ooe.moc5.amazingrace.model.json.CheckpointModel;
8 import at.fh.ooe.moc5.amazingrace.model.json.RouteModel;
9 import at.fh.ooe.moc5.amazingrace.model.json.RouteRequestModel;
10 import at.fh.ooe.moc5.amazingrace.model.json.CheckpointRequestModel;
11 import at.fh.ooe.moc5.amazingrace.service.ServiceProxy;
12 import at.fh.ooe.moc5.amazingrace.service.ServiceException;
13 import at.fh.ooe.moc5.amazingrace.service.ServiceProxyFactory;
14
15 /**
16  * Created by Thomas on 12/25/2015.
17  */
18 public class CheckpointViewModel implements Serializable {
19
20     private String answer;
21     private final UserContextModel userContext;
22     private RouteModel route;
23
24     private ServiceProxy proxy;
25
26     public CheckpointViewModel(UserContextModel userContext, RouteModel route) {
27         Objects.requireNonNull(userContext, "View model needs user context set");
28         Objects.requireNonNull(route, "View model needs route set");
29
30         this.userContext = userContext;
31         this.route = route;
32         proxy = ServiceProxyFactory.createServiceProxy();
33     }
34
35     //region Actions
36     public boolean validateSecret() throws ServiceException {
37         if (!isValid()) {
38             return Boolean.FALSE;
39         }
40
41         return proxy.validateCheckpointSecret(new
42             ↪ CheckpointRequestModel(userContext.getCredentialsModel().userName,
43             ↪ userContext.getCredentialsModel().password, getNextCheckpoint().getId(), answer));
44     }
45
46     public boolean reloadRoute() throws ServiceException {
47         final List<RouteModel> routes = proxy.loadRoutes(userContext.getCredentialsModel());
48         int idx = -1;
49         if ((idx = routes.indexOf(route)) != -1) {
50             route = routes.get(idx);
51         }
52
53         return (route != null);
54     }
55
56     public boolean resetRoute() throws ServiceException {
57         return proxy.resetRoute(new RouteRequestModel(userContext.getCredentialsModel(),
58             ↪ route.getId()));
59     }
60
61 }

```

Übung 3

```
58     public boolean isValid() {
59         return (answer != null) && (!answer.trim().isEmpty());
60     }
61     //endregion
62
63     //region Getter and Setter
64     public String getAnswer() {
65         return answer;
66     }
67
68     public void setAnswer(String answer) {
69         this.answer = answer;
70     }
71
72     public UserContextModel getUserContext() {
73         return userContext;
74     }
75
76     public RouteModel getRoute() {
77         return route;
78     }
79
80     public CheckpointModel getNextCheckpoint() {
81         return route.getNextCheckpoint();
82     }
83
84     public List<CheckpointModel> getVisitedCheckpoints() {
85         return route.getVisitedCheckpoints();
86     }
87     //endregion
88 }
```

Übung 3

1.2.7 Service

Listing 18: ServiceProxy.java

```

1 package at.fh.ooe.moc5.amazingrace.service;
2
3 import java.io.Serializable;
4 import java.util.List;
5
6 import at.fh.ooe.moc5.amazingrace.model.json.CredentialsRequestModel;
7 import at.fh.ooe.moc5.amazingrace.model.json.RouteModel;
8 import at.fh.ooe.moc5.amazingrace.model.json.RouteRequestModel;
9 import at.fh.ooe.moc5.amazingrace.model.json.CheckpointRequestModel;
10
11 /**
12  * Created by Thomas on 12/25/2015.
13  * The specification of the service proxy
14  */
15 public interface ServiceProxy extends Serializable {
16
17     /**
18      * Validates the given user credentials.
19      *
20      * @param model the model containing authentication data
21      * @return true if the credentials are valid, false otherwise
22      * @throws ServiceException if an service error occurs. See contained error code for details
23      */
24     boolean validateCredentials(CredentialsRequestModel model) throws ServiceException;
25
26     /**
27      * Validates the checkpoint secret.
28      *
29      * @param model the model containing the checkpoint data and authentication data
30      * @return true if valid, false otherwise
31      * @throws ServiceException if an service error occurs. See contained error code for details
32      */
33     boolean validateCheckpointSecret(CheckpointRequestModel model) throws ServiceException;
34
35     /**
36      * Loads all routes.
37      *
38      * @param model the credentials model for authentication
39      * @return the loaded routes
40      * @throws ServiceException if an service error occurs. See contained error code for details
41      */
42     List<RouteModel> loadRoutes(CredentialsRequestModel model) throws ServiceException;
43
44     /**
45      * Resets the given route.
46      *
47      * @param model the model containing the route data and authentication data
48      * @return true if reset, false otherwise
49      * @throws ServiceException if an service error occurs. See contained error code for details
50      */
51     boolean resetRoute(RouteRequestModel model) throws ServiceException;
52
53     /**
54      * Resets all routes.
55      *
56      * @param model the model containing the authentication data
57      * @return true if reset, false otherwise
58      * @throws ServiceException if an service error occurs. See contained error code for details
59      */

```

Übung 3

```
60 |         boolean resetAllRoutes(CredentialsRequestModel model) throws ServiceException;  
61 |     }
```

Listing 19: RestServiceProxyImpl.java

```

1 package at.fh.ooe.moc5.amazingrace.service;
2
3 import com.google.gson.GsonBuilder;
4 import com.google.gson.JsonSyntaxException;
5 import com.google.gson.reflect.TypeToken;
6
7 import java.io.BufferedReader;
8 import java.io.BufferedWriter;
9 import java.io.InputStreamReader;
10 import java.io.OutputStreamWriter;
11 import java.net.SocketTimeoutException;
12 import java.net.URL;
13 import java.util.List;
14 import java.util.Objects;
15
16 import javax.net.ssl.HttpsURLConnection;
17
18 import at.fh.ooe.moc5.amazingrace.model.json.CredentialsRequestModel;
19 import at.fh.ooe.moc5.amazingrace.model.json.RouteModel;
20 import at.fh.ooe.moc5.amazingrace.model.json.RouteRequestModel;
21 import at.fh.ooe.moc5.amazingrace.model.json.CheckpointRequestModel;
22 import at.fh.ooe.moc5.amazingrace.service.ServiceException.ServiceErrorCode;
23
24 /**
25  * Created by Thomas on 12/24/2015.
26  * <p/>
27  * Class for communicating with the rest backend.
28  */
29 public class RestServiceProxyImpl implements ServiceProxy {
30
31     public static final String REST_URL =
32         ↪ "https://demo.nexperts.com/MOC5/AmazingRaceService/AmazingRaceService.svc";
33     public static final String CHECK_CREDENTIALS = "/CheckCredentials";
34     public static final String GET_ROUTES_METHOD = "/GetRoutes";
35     public static final String METHOD_VISIT_CHECKPOINT = "/InformAboutVisitedCheckpoint";
36     public static final String METHOD_RESET_ALL_ROUTES = "/ResetAllRoutes";
37     public static final String METHOD_RESET_ROUTE = "/ResetRoute";
38
39     public static final int DEFAULT_TIME_OUT = 3000;
40
41     @Override
42     public boolean validateCredentials(CredentialsRequestModel model) throws ServiceException {
43         Objects.requireNonNull(model, "Cannot check credentials for null model");
44
45         try {
46             URL url = new URL(new
47                 ↪ StringBuilder(REST_URL).append(CHECK_CREDENTIALS).append("?").append(model.toQueryString()).toS
48             HttpsURLConnection connection = (HttpsURLConnection) url.openConnection();
49             connection.setRequestMethod("GET");
50             connection.setConnectTimeout(DEFAULT_TIME_OUT);
51             connection.setUseCaches(Boolean.FALSE);
52
53             return invokeBooleanResultMethod(connection, null, Boolean.FALSE);
54         } catch (ServiceException e) {
55             throw e;
56         } catch (Exception e) {
57             throw new ServiceException(ServiceErrorCode.UNKNOWN, e);
58         }
59     }
60 }

```

Übung 3

```

59  @Override
60  public boolean validateCheckpointSecret(CheckpointRequestModel model) throws ServiceException
    ↪ {
61      Objects.requireNonNull(model, "Cannot check checkpoint for null model");
62
63      try {
64          URL url = new URL(new
            ↪   StringBuilder(REST_URL).append(METHOD_VISIT_CHECKPOINT).toString());
65          HttpsURLConnection connection = (HttpsURLConnection) url.openConnection();
66          connection.setRequestMethod("POST");
67          connection.setRequestProperty("Content-Type", "application/json");
68          connection.setConnectTimeout(DEFAULT_TIME_OUT);
69          connection.setUseCaches(Boolean.FALSE);
70          connection.setDoInput(Boolean.TRUE);
71          connection.setDoOutput(Boolean.TRUE);
72
73          return invokeBooleanResultMethod(connection, model, Boolean.TRUE);
74      } catch (ServiceException e) {
75          throw e;
76      } catch (Exception e) {
77          throw new ServiceException(ServiceErrorCode.UNKNOWN, e);
78      }
79  }
80
81  @Override
82  public List<RouteModel> loadRoutes(CredentialsRequestModel model) throws ServiceException {
83      Objects.requireNonNull(model, "Cannot get routes with missing credentials");
84
85      if (!validateCredentials(model)) {
86          throw new ServiceException(ServiceErrorCode.INVALID_CREDENTIALS);
87      }
88
89      try {
90          URL url = new URL(new
            ↪   StringBuilder(REST_URL).append(GET_ROUTES_METHOD).append("?").append(model.toQueryString()).toS
91          HttpsURLConnection connection = (HttpsURLConnection) url.openConnection();
92          connection.setRequestMethod("GET");
93          connection.setConnectTimeout(DEFAULT_TIME_OUT);
94          connection.setUseCaches(Boolean.FALSE);
95
96          final String response = readResponse(connection);
97          GsonBuilder builder = new GsonBuilder();
98          return builder.create().fromJson(response, new TypeToken<List<RouteModel>>() {
99              .getType();
100      } catch (JsonSyntaxException e) {
101          throw new ServiceException(ServiceErrorCode.INVALID_REQUEST, e);
102      } catch (Exception e) {
103          throw new ServiceException(ServiceErrorCode.UNKNOWN, e);
104      }
105  }
106
107  @Override
108  public boolean resetRoute(RouteRequestModel model) throws ServiceException {
109      Objects.requireNonNull(model, "Cannot reset route for null model");
110
111      try {
112          URL url = new URL(new StringBuilder(REST_URL).append(METHOD_RESET_ROUTE).toString());
113          HttpsURLConnection connection = (HttpsURLConnection) url.openConnection();
114          connection.setRequestMethod("POST");
115          connection.setRequestProperty("Content-Type", "application/json");
116          connection.setConnectTimeout(DEFAULT_TIME_OUT);
117          connection.setUseCaches(Boolean.FALSE);
118          connection.setDoInput(Boolean.TRUE);

```


Übung 3

```

119         connection.setDoOutput(Boolean.TRUE);
120
121         return invokeBooleanResultMethod(connection, model, Boolean.TRUE);
122     } catch (ServiceException e) {
123         throw e;
124     } catch (Exception e) {
125         throw new ServiceException(ServiceErrorCode.UNKNOWN, e);
126     }
127 }
128
129 @Override
130 public boolean resetAllRoutes(CredentialsRequestModel model) throws ServiceException {
131     Objects.requireNonNull(model, "Cannot reset all routes for null model");
132
133     try {
134         URL url = new URL(new
135             ↳ StringBuilder(REST_URL).append(METHOD_RESET_ALL_ROUTES).toString());
136         HttpURLConnection connection = (HttpURLConnection) url.openConnection();
137         connection.setRequestMethod("POST");
138         connection.setRequestProperty("Content-Type", "application/json");
139         connection.setConnectTimeout(DEFAULT_TIME_OUT);
140         connection.setUseCaches(Boolean.FALSE);
141         connection.setDoInput(Boolean.TRUE);
142         connection.setDoOutput(Boolean.TRUE);
143
144         return invokeBooleanResultMethod(connection, model, Boolean.TRUE);
145     } catch (ServiceException e) {
146         throw e;
147     } catch (Exception e) {
148         throw new ServiceException(ServiceErrorCode.UNKNOWN, e);
149     }
150
151     // region Helper
152
153     /**
154      * Invokes a remote service method with an boolean result.
155      *
156      * @param connection      the connection to the service method
157      * @param jsonModel       the jsonModel to write into the connection, maybe null.
158      * @param checkCredentials true if the credentials shall be checked befor calling the actual
159      ↳ service
160      * @param <T>              the type of the json model which must be at least the
161      ↳ CredentialsModel
162      * @return true or false
163      * @throws ServiceException if the request failed for any reason, see contained error code for
164      ↳ details
165      */
166     private <T extends CredentialsRequestModel> boolean
167     ↳ invokeBooleanResultMethod(HttpURLConnection connection, T jsonModel, boolean
168     ↳ checkCredentials) throws ServiceException {
169         Objects.requireNonNull(connection, "Cannot invoke method on null connection");
170         if (checkCredentials) {
171             Objects.requireNonNull(jsonModel, "Cannot check credentials for null model");
172         }
173         if ((checkCredentials) && (!validateCredentials(new
174             ↳ CredentialsRequestModel(jsonModel.userName, jsonModel.password)))) {
175             throw new ServiceException(ServiceErrorCode.INVALID_CREDENTIALS);
176         }
177
178         try {
179             // Write Data for Post requests
180             if ((connection.getRequestMethod().equals("POST")) && (jsonModel != null)) {

```

Übung 3

```

175         writeData(connection, jsonModel);
176     }
177     // Read response
178     final String response = readResponse(connection);
179     if ((!"true".equals(response)) && (!"false".equals(response))) {
180         throw new ServiceException(ServiceErrorCode.INVALID_REQUEST);
181     }
182     return Boolean.parseBoolean(response);
183 } catch (ServiceException e) {
184     throw e;
185 } catch (Exception e) {
186     throw new ServiceException(ServiceErrorCode.UNKNOWN, e);
187 }
188 }
189
190 /**
191  * Reads the response from the connection.
192  *
193  * @param connection the connection to read response from
194  * @return the response string
195  * @throws ServiceException if an error occurs
196  * @see ServiceErrorCode for the ServiceException contained error code
197  */
198 private String readResponse(URLConnection connection) throws ServiceException {
199     Objects.requireNonNull(connection, "Cannot read from null connection");
200     try {
201         if (connection.getResponseCode() != 200) {
202             throw new ServiceException(ServiceErrorCode.REQUEST_NOT_OK);
203         }
204         try (final BufferedReader reader = new BufferedReader(new
205             ↳ InputStreamReader(connection.getInputStream()))) {
206             final StringBuilder builder = new StringBuilder();
207             String line;
208             while ((line = reader.readLine()) != null) {
209                 builder.append(line);
210             }
211             return builder.toString();
212         } catch (SocketTimeoutException ste) {
213             throw new ServiceException(ServiceErrorCode.TIMEOUT, ste);
214         } catch (Exception e) {
215             throw new ServiceException(ServiceErrorCode.UNKNOWN, e);
216         }
217     }
218
219 /**
220  * Writes data to the connection.
221  *
222  * @param connection the connection to write to
223  * @param jsonModelInstance the json model instance
224  * @throws ServiceException if the write fails, see contained error code for details
225  */
226 private void writeData(URLConnection connection, Object jsonModelInstance) throws
227     ↳ ServiceException {
228     Objects.requireNonNull(connection, "Cannot write to null connection");
229     try {
230         try (final BufferedWriter writer = new BufferedWriter(new
231             ↳ OutputStreamWriter(connection.getOutputStream()))) {
232             final String data = new GsonBuilder().create().toJson(jsonModelInstance);
233             writer.write(data, 0, data.length());
234             writer.flush();
235         }
236     } catch (JsonSyntaxException jse) {

```

Übung 3

```
235         throw new ServiceException(ServiceErrorCode.INVALID_REQUEST, jse);
236     } catch (SocketTimeoutException ste) {
237         throw new ServiceException(ServiceErrorCode.TIMEOUT, ste);
238     } catch (Exception e) {
239         throw new ServiceException(ServiceErrorCode.UNKNOWN, e);
240     }
241 }
242 // endregion
243 }
```

Listing 20: ServiceException.java

```

1 package at.fh.ooe.moc5.amazingrace.service;
2
3 /**
4  * Created by Thomas on 12/25/2015.
5  */
6 public class ServiceException extends Exception {
7     /**
8      * Enumeration which defines the common service errors
9      */
10    public static enum ServiceErrorCode {
11        INVALID_REQUEST,
12        TIMEOUT,
13        UNKNOWN,
14        REQUEST_NOT_OK,
15        INVALID_CREDENTIALS
16    }
17
18    private ServiceErrorCode errorCode;
19
20    public ServiceException() {
21        this(null, null, null);
22    }
23
24    public ServiceException(String detailMessage) {
25        this(null, detailMessage, null);
26    }
27
28    public ServiceException(String detailMessage, Throwable throwable) {
29        this(null, detailMessage, throwable);
30    }
31
32    public ServiceException(Throwable throwable) {
33        this(null, null, throwable);
34    }
35
36    public ServiceException(ServiceErrorCode errorCode) {
37        this(errorCode, null, null);
38    }
39
40    public ServiceException(ServiceErrorCode errorCode, String detailMessage) {
41        this(errorCode, detailMessage, null);
42    }
43
44
45    public ServiceException(ServiceErrorCode errorCode, Throwable throwable) {
46        this(errorCode, null, throwable);
47    }
48
49    public ServiceException(ServiceErrorCode errorCode, String detailMessage, Throwable throwable)
50    ↪ {
51        super(detailMessage, throwable);
52        this.errorCode = errorCode;
53    }
54
55    public ServiceErrorCode getErrorCode() {
56        return errorCode;
57    }
58 }

```

Listing 21: ServiceProxyFactory.java

```
1 package at.fh.ooe.moc5.amazingrace.service;
2
3 /**
4  * Created by Thomas on 12/24/2015.
5  */
6 public class ServiceProxyFactory {
7
8     private static ServiceProxy proxy;
9
10    private ServiceProxyFactory() {
11    }
12
13    public static ServiceProxy createServiceProxy() {
14        if (proxy == null) {
15            proxy = new RestServiceProxyImpl();
16        }
17        return proxy;
18    }
19 }
```

Übung 3

1.2.8 Utilities

Listing 22: DialogUtil.java

```

1 package at.fh.ooe.moc5.amazingrace.util;
2
3 import android.app.AlertDialog;
4 import android.content.Context;
5 import android.content.DialogInterface;
6
7 import java.util.Objects;
8
9 import at.fh.ooe.moc5.amazingrace.R;
10
11 /**
12  * Created by Thomas on 12/24/2015.
13  * Utility for creating dialogs.
14  */
15 public class DialogUtil {
16
17     /**
18      * Creates a error alert dialog.
19      *
20      * @param ctx the context to create the alert dialog for
21      * @param message the resource id of the alert dialog error message
22      * @param listener the button listener
23      * @return the created alert dialog
24      */
25     public static AlertDialog createErrorDialog(final Context ctx, final int message, final
26         DialogInterface.OnClickListener listener) {
27         Objects.requireNonNull(ctx, "Cannot create dialog for null context");
28         Objects.requireNonNull(listener, "Listener for buttons mut be given");
29
30         return new AlertDialog.Builder(ctx)
31             .setTitle(R.string.dialog_title_error)
32             .setMessage(message)
33             .setPositiveButton(R.string.action_ok, listener)
34             .create();
35     }
36
37     /**
38      * Creates a yes, no alert dialog
39      *
40      * @param ctx the context to create the alert dialog for
41      * @param title the resource id of the alert dialog title
42      * @param message the resource id of the alert dialog message
43      * @param listener the yes, no listener
44      * @return the created alert dialog
45      */
46     public static AlertDialog createYesNoAlertDialog(final Context ctx, final int title, final int
47         message, final DialogInterface.OnClickListener listener) {
48         Objects.requireNonNull(ctx, "Cannot create dialog for null context");
49         Objects.requireNonNull(listener, "Listener for buttons mut be given");
50
51         return new AlertDialog.Builder(ctx)
52             .setTitle(title)
53             .setMessage(message)
54             .setPositiveButton(R.string.action_yes, listener)
55             .setNegativeButton(R.string.action_no, listener)
56             .create();
57     }
58
59     /**

```

Übung 3

```
58     * Creates a alert dialog with an custom view as content.
59     *
60     * @param ctx      the context to create dialog for
61     * @param viewId    the resource id of the view
62     * @param title     the resource id of the alert dialog title
63     * @param listener  the button listener
64     * @return the created alert dialog
65     */
66     public static AlertDialog createCustomContentDialog(final Context ctx, final int viewId, final
↪     int title, final DialogInterface.OnClickListener listener) {
67         Objects.requireNonNull(ctx, "Cannot create dialog for null context");
68         Objects.requireNonNull(listener, "Listener for buttons mut be given");
69
70         return new AlertDialog.Builder(ctx)
71             .setTitle(title)
72             .setView(viewId)
73             .setPositiveButton(R.string.action_ok, listener)
74             .create();
75     }
76 }
```

Übung 3

1.2.9 Watcher

Listing 23: AnswerButtonTextWatcher.java

```

1  package at.fh.ooe.moc5.amazingrace.watcher;
2
3  import android.text.Editable;
4  import android.text.TextWatcher;
5  import android.widget.Button;
6
7  import at.fh.ooe.moc5.amazingrace.R;
8  import at.fh.ooe.moc5.amazingrace.activity.CheckpointActivity;
9  import at.fh.ooe.moc5.amazingrace.model.view.CheckpointViewModel;
10
11  /**
12   * Created by Thomas on 12/24/2015.
13   * A watcher which enables the answer button if the backed view model is valid, disables it
14   * ↪ otherwise
15   */
16  public class AnswerButtonTextWatcher implements TextWatcher {
17
18      private final CheckpointViewModel viewModel;
19      private final CheckpointActivity activity;
20
21      public AnswerButtonTextWatcher(CheckpointActivity activity, CheckpointViewModel viewModel) {
22          this.activity = activity;
23          this.viewModel = viewModel;
24      }
25
26      @Override
27      public void beforeTextChanged(CharSequence s, int start, int count, int after) {
28      }
29
30      @Override
31      public void onTextChanged(CharSequence s, int start, int before, int count) {
32      }
33
34      @Override
35      public void afterTextChanged(Editable s) {
36          ((Button) activity.findViewById(R.id.answerBtn)).setEnabled(viewModel.isValid());
37      }
38  }

```


Listing 24: CheckpointViewModelBindingTextWatcher.java

```

1 package at.fh.ooe.moc5.amazingrace.watcher;
2
3 import android.text.Editable;
4 import android.text.TextWatcher;
5
6 import java.util.Objects;
7
8 import at.fh.ooe.moc5.amazingrace.model.view.CheckpointViewModel;
9
10 /**
11  * Created by Thomas on 12/24/2015.
12  * Performs a binding to the given view model
13  */
14 public class CheckpointViewModelBindingTextWatcher implements TextWatcher {
15
16     private final CheckpointViewModel view;
17
18     public CheckpointViewModelBindingTextWatcher(CheckpointViewModel view) {
19         Objects.requireNonNull(view);
20         this.view = view;
21     }
22
23     @Override
24     public void beforeTextChanged(CharSequence s, int start, int count, int after) {
25     }
26
27     @Override
28     public void onTextChanged(CharSequence s, int start, int before, int count) {
29     }
30
31     @Override
32     public void afterTextChanged(Editable s) {
33         view.setAnswer(s.toString());
34     }
35 }

```

Listing 25: LoginButtonTextWatcher.java

```

1 package at.fh.ooe.moc5.amazingrace.watcher;
2
3 import android.text.Editable;
4 import android.text.TextWatcher;
5 import android.widget.Button;
6
7 import at.fh.ooe.moc5.amazingrace.R;
8 import at.fh.ooe.moc5.amazingrace.activity.LoginActivity;
9 import at.fh.ooe.moc5.amazingrace.model.view.LoginViewModel;
10
11 /**
12  * Created by Thomas on 12/24/2015.
13  * A watcher which enables the login button if the backed view model is valid, disables it
14  * ↪ otherwise
15  */
16 public class LoginButtonTextWatcher implements TextWatcher {
17     private LoginViewModel viewModel;
18     private final LoginActivity activity;
19
20     public LoginButtonTextWatcher(LoginActivity activity, LoginViewModel viewModel) {
21         this.activity = activity;
22         this.viewModel = viewModel;
23     }
24
25     @Override
26     public void beforeTextChanged(CharSequence s, int start, int count, int after) {
27     }
28
29     @Override
30     public void onTextChanged(CharSequence s, int start, int before, int count) {
31     }
32
33     @Override
34     public void afterTextChanged(Editable s) {
35         Button loginButton = (Button) activity.findViewById(R.id.loginBtn);
36         loginButton.setEnabled(viewModel.isValid());
37     }
38 }

```

Listing 26: LoginViewModelBindingTextWatcher.java

```

1  package at.fh.ooe.moc5.amazingrace.watcher;
2
3  import android.text.Editable;
4  import android.text.TextWatcher;
5  import android.widget.EditText;
6
7  import java.util.Objects;
8
9  import at.fh.ooe.moc5.amazingrace.R;
10 import at.fh.ooe.moc5.amazingrace.model.view.LoginViewModel;
11
12 /**
13  * Created by Thomas on 12/24/2015.
14  * Performs a binding to the given view model
15  */
16 public class LoginViewModelBindingTextWatcher implements TextWatcher {
17
18     private final LoginViewModel view;
19     private final EditText text;
20
21     public LoginViewModelBindingTextWatcher(LoginViewModel view, EditText text) {
22         Objects.requireNonNull(view);
23         Objects.requireNonNull(text);
24         this.view = view;
25         this.text = text;
26     }
27
28     @Override
29     public void beforeTextChanged(CharSequence s, int start, int count, int after) {
30     }
31
32     @Override
33     public void onTextChanged(CharSequence s, int start, int before, int count) {
34     }
35
36     @Override
37     public void afterTextChanged(Editable s) {
38         switch (text.getId()) {
39             case R.id.usernameEdTxt:
40                 view.setUsername(s.toString());
41                 break;
42             case R.id.passwordEdTxt:
43                 view.setPassword(s.toString());
44                 break;
45         }
46     }
47 }

```

Übung 3

1.3 Layout

Listing 27: activity_login.xml

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:tools="http://schemas.android.com/tools"
4      android:layout_width="match_parent"
5      android:layout_height="match_parent"
6      android:paddingBottom="@dimen/activity_vertical_margin"
7      android:paddingLeft="@dimen/activity_horizontal_margin"
8      android:paddingRight="@dimen/activity_horizontal_margin"
9      android:paddingTop="@dimen/activity_vertical_margin"
10     tools:context="at.fh.ooe.moc5.amazingrace.activity.LoginActivity">
11
12
13     <LinearLayout
14         android:layout_width="match_parent"
15         android:layout_height="wrap_content"
16         android:layout_centerVertical="true"
17         android:orientation="vertical"
18         android:weightSum="1">
19
20         <LinearLayout
21             android:layout_width="match_parent"
22             android:layout_height="wrap_content"
23             android:layout_gravity="center_horizontal"
24             android:orientation="horizontal">
25
26             <TextView
27                 android:id="@+id/usernameLabel"
28                 android:layout_width="wrap_content"
29                 android:layout_height="match_parent"
30                 android:layout_gravity="center"
31                 android:layout_weight="1"
32                 android:gravity="left|center"
33                 android:text="@string/label_username"
34                 android:textAppearance="?android:attr/textAppearanceMedium" />
35
36             <EditText
37                 android:id="@+id/usernameEdTxt"
38                 android:layout_width="wrap_content"
39                 android:layout_height="wrap_content"
40                 android:layout_gravity="center|center"
41                 android:layout_weight="8" />
42         </LinearLayout>
43
44         <LinearLayout
45             android:layout_width="match_parent"
46             android:layout_height="wrap_content"
47             android:layout_gravity="center_horizontal"
48             android:orientation="horizontal">
49
50             <TextView
51                 android:id="@+id/passwordLabel"
52                 android:layout_width="wrap_content"
53                 android:layout_height="match_parent"
54                 android:layout_gravity="left|center"
55                 android:layout_weight="1"
56                 android:gravity="left|center"
57                 android:text="@string/label_password"
58                 android:textAppearance="?android:attr/textAppearanceMedium" />
59

```

Übung 3

```

60         <EditText
61             android:id="@+id/passwordEdTxt"
62             android:layout_width="wrap_content"
63             android:layout_height="wrap_content"
64             android:layout_gravity="center|center"
65             android:layout_weight="8"
66             android:inputType="textPassword" />
67     </LinearLayout>
68
69     <RelativeLayout
70         android:layout_width="match_parent"
71         android:layout_height="wrap_content"
72         android:layout_weight="0.07"
73         android:orientation="horizontal"
74         android:weightSum="1">
75
76         <LinearLayout
77             android:layout_width="wrap_content"
78             android:layout_height="wrap_content"
79             android:layout_alignParentLeft="false"
80             android:layout_alignParentTop="false"
81             android:layout_centerHorizontal="true"
82             android:layout_centerInParent="true"
83             android:layout_centerVertical="true"
84             android:orientation="horizontal">
85
86             <Button
87                 android:id="@+id/loginBtn"
88                 android:layout_width="wrap_content"
89                 android:layout_height="wrap_content"
90                 android:layout_margin="5pt"
91                 android:gravity="center|center"
92                 android:text="@string/action_login" />
93
94             <Button
95                 android:id="@+id/cancelBtn"
96                 android:layout_width="wrap_content"
97                 android:layout_height="wrap_content"
98                 android:layout_margin="5pt"
99                 android:gravity="center|center"
100                 android:text="@string/action_cancel" />
101
102         </LinearLayout>
103
104     </RelativeLayout>
105 </LinearLayout>
106 </RelativeLayout>

```

Listing 28: activity_route.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:tools="http://schemas.android.com/tools"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     android:paddingBottom="@dimen/activity_vertical_margin"
7     android:paddingLeft="@dimen/activity_horizontal_margin"
8     android:paddingRight="@dimen/activity_horizontal_margin"
9     android:paddingTop="@dimen/activity_vertical_margin"
10    tools:context="at.fh.ooe.moc5.amazingrace.activity.RouteActivity">
11
12    <LinearLayout
13        android:layout_width="match_parent"
14        android:layout_height="match_parent"
15        android:orientation="vertical"
16        android:weightSum="1">
17
18        <ListView
19            android:id="@+id/listRoute"
20            android:layout_width="match_parent"
21            android:layout_height="match_parent"/>
22    </LinearLayout>
23 </RelativeLayout>
```

Listing 29: activity_checkpoint.xml

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:tools="http://schemas.android.com/tools"
4      android:layout_width="match_parent"
5      android:layout_height="wrap_content"
6      android:paddingBottom="@dimen/activity_vertical_margin"
7      android:paddingLeft="@dimen/activity_horizontal_margin"
8      android:paddingRight="@dimen/activity_horizontal_margin"
9      android:paddingTop="@dimen/activity_vertical_margin"
10     tools:context=".activity.CheckpointActivity">
11
12     <ScrollView
13         android:layout_width="match_parent"
14         android:layout_height="match_parent"
15         android:fillViewport="true">
16
17         <LinearLayout
18             android:layout_width="match_parent"
19             android:layout_height="match_parent"
20             android:layout_alignParentStart="true"
21             android:layout_alignParentTop="true"
22             android:orientation="vertical">
23
24             <TextView
25                 android:id="@+id/routeCheckpointContainerLabel"
26                 android:layout_width="match_parent"
27                 android:layout_height="wrap_content"
28                 android:layout_margin="5pt"
29                 android:text="@string/checkpoint_group_secret"
30                 android:textAppearance="?android:attr/textAppearanceLarge"
31                 android:textStyle="bold" />
32
33             <LinearLayout
34                 android:id="@+id/routeCheckpointContainer"
35                 android:layout_width="match_parent"
36                 android:layout_height="wrap_content"
37                 android:layout_alignParentStart="true"
38                 android:layout_alignParentTop="true"
39                 android:layout_margin="5pt"
40                 android:orientation="vertical">
41
42                 <TextView
43                     android:id="@+id/nextCheckpointTxt"
44                     android:layout_width="match_parent"
45                     android:layout_height="wrap_content"
46                     android:layout_marginBottom="5pt"
47                     android:textAppearance="?android:attr/textAppearanceMedium" />
48
49                 <TextView
50                     android:id="@+id/nextCheckpointHintTxt"
51                     android:layout_width="match_parent"
52                     android:layout_height="wrap_content"
53                     android:layout_marginBottom="5pt"
54                     android:textAppearance="?android:attr/textAppearanceMedium" />
55
56                 <EditText
57                     android:id="@+id/hintAnswerEdTxt"
58                     android:layout_width="match_parent"
59                     android:layout_height="wrap_content"
60                     android:layout_marginBottom="5pt" />

```

Übung 3

```

61
62         <Button
63             android:id="@+id/answerBtn"
64             android:layout_width="wrap_content"
65             android:layout_height="wrap_content"
66             android:gravity="center|center"
67             android:text="@string/action_answer" />
68     </LinearLayout>
69
70     <LinearLayout
71         android:id="@+id/routeCheckpointMapContainerLabel"
72         android:layout_width="match_parent"
73         android:layout_height="wrap_content"
74         android:layout_marginBottom="5pt"
75         android:orientation="horizontal">
76
77         <ImageView
78             android:layout_width="20pt"
79             android:layout_height="20pt"
80             android:clickable="true"
81             android:onClick="toggleVisibility"
82             android:src="@drawable/map_icon" />
83
84         <TextView
85             android:layout_width="match_parent"
86             android:layout_height="match_parent"
87             android:layout_marginLeft="5pt"
88             android:clickable="true"
89             android:gravity="left|center"
90             android:onClick="toggleVisibility"
91             android:text="@string/checkpoint_group_map"
92             android:textAppearance="?android:attr/textAppearanceLarge"
93             android:textStyle="bold" />
94     </LinearLayout>
95
96     <LinearLayout
97         android:id="@+id/routeCheckpointMapContainer"
98         android:layout_width="match_parent"
99         android:layout_height="200dp"
100        android:layout_marginBottom="10pt"
101        android:layout_alignParentStart="true"
102        android:layout_alignParentTop="true"
103        android:orientation="vertical">
104
105        <fragment
106            android:id="@+id/routeCheckpointMap"
107            android:name="com.google.android.gms.maps.MapFragment"
108            android:layout_width="match_parent"
109            android:layout_height="match_parent" />
110    </LinearLayout>
111
112    <LinearLayout
113        android:id="@+id/routeCheckpointListContainerLabel"
114        android:layout_width="match_parent"
115        android:layout_height="wrap_content"
116        android:layout_marginBottom="5pt"
117        android:orientation="horizontal">
118
119        <ImageView
120            android:layout_width="20pt"
121            android:layout_height="20pt"
122            android:clickable="true"
123            android:onClick="toggleVisibility"

```


Übung 3

```

124         android:src="@drawable/list_icon" />
125
126     <TextView
127         android:layout_width="match_parent"
128         android:layout_height="match_parent"
129         android:layout_marginLeft="5pt"
130         android:clickable="true"
131         android:gravity="left|center"
132         android:onClick="toggleVisibility"
133         android:text="@string/checkpoint_group_list"
134         android:textAppearance="?android:attr/textAppearanceLarge"
135         android:textStyle="bold" />
136 </LinearLayout>
137
138 <LinearLayout
139     android:id="@+id/routeCheckpointListContainer"
140     android:layout_width="match_parent"
141     android:layout_height="wrap_content"
142     android:layout_alignParentStart="true"
143     android:layout_alignParentTop="true"
144     android:orientation="vertical">
145
146     <LinearLayout
147         android:id="@+id/routeCheckpointListView"
148         android:layout_width="match_parent"
149         android:layout_height="wrap_content"
150         android:layout_below="@+id/routeListTitleTxt"
151         android:orientation="vertical" />
152     </LinearLayout>
153 </LinearLayout>
154 </ScrollView>
155 </RelativeLayout>

```

Listing 30: view_checkpoint_item.xml

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="wrap_content"
5      android:layout_centerVertical="true"
6      android:orientation="horizontal">
7
8      <TextView
9          android:id="@+id/visitedCheckpointNameLabel"
10         android:layout_width="match_parent"
11         android:layout_height="wrap_content"
12         android:layout_margin="5pt"
13         android:layout_weight="1"
14         android:gravity="left" />
15 </LinearLayout>

```

Listing 31: view_route_item.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="match_parent"
4     android:layout_height="wrap_content"
5     android:layout_centerVertical="true"
6     android:orientation="vertical"
7     android:weightSum="1">
8
9     <TextView
10         android:id="@+id/routeNameLabel"
11         android:layout_width="match_parent"
12         android:layout_height="match_parent"
13         android:layout_margin="5pt"
14         android:layout_weight="1"
15         android:gravity="left" />
16 </LinearLayout>
```

Listing 32: view_congratulations_dialog.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="match_parent"
4     android:layout_height="wrap_content"
5     android:layout_centerVertical="true"
6     android:orientation="vertical"
7     android:weightSum="1">
8
9     <ImageView
10         android:layout_width="50pt"
11         android:layout_height="50pt"
12         android:src="@drawable/congratulations_icon"
13         android:layout_gravity="center|center"/>
14
15     <TextView
16         android:layout_width="match_parent"
17         android:layout_height="match_parent"
18         android:layout_margin="5pt"
19         android:layout_weight="1"
20         android:gravity="center|center"
21         android:text="@string/info_congratulations" />
22 </LinearLayout>
```

Übung 3

1.4 Values

Listing 33: strings.xml

```

1 <resources>
2   <string name="application_title">AmazingRace</string>
3   <string name="activity_title_login">Login AmazingRace</string>
4   <string name="activity_title_route">AmazingRace Routes</string>
5   <string name="activity_title_checkpoint">Route checkpoints</string>
6   <string name="action_answer">Answer</string>
7   <string name="action_cancel">Cancel</string>
8   <string name="action_ok">Ok</string>
9   <string name="action_yes">Yes</string>
10  <string name="action_login">Login</string>
11  <string name="action_settings">Settings</string>
12  <string name="action_no">No</string>
13  <string name="action_play">Play</string>
14  <string name="action_reset">Reset</string>
15  <string name="action_reset_all">Reset All</string>
16  <string name="action_close">Close</string>
17  <string name="action_open">Open</string>
18  <string name="action_reload">Reload</string>
19  <string name="label_username">Username</string>
20  <string name="label_password">Password</string>
21  <string name="progress_login">Logging in ...</string>
22  <string name="progress_loading_routes">Loading routes ...</string>
23  <string name="progress_updating_route">Updating route ...</string>
24  <string name="progress_validating_secret">Validating secret</string>
25  <string name="progress_title">Processing</string>
26  <string name="progress_resetting_routes">Resetting route(s) ...</string>
27  <string name="dialog_title_error">Error</string>
28  <string name="dialog_title_warning">Warning</string>
29  <string name="dialog_title_info">Info</string>
30  <string name="error_request_invalid">Sorry. Request failed because it was invalid</string>
31  <string name="error_request_timeout">Sorry. Request failed because of an timeout</string>
32  <string name="error_unknown">Sorry. An unknown error occurred</string>
33  <string name="error_login_failed">User and/or password not correct</string>
34  <string name="error_user_became_invalid">Your user is not valid anymore. You will be logged
    ↪ out.</string>
35  <string name="error_user_not_logged">Your are no longer logged in. You will be returned to the
    ↪ login page</string>
36  <string name="error_route_reset_failed">The route(s) could not be reset. Please try
    ↪ again</string>
37  <string name="error_no_network">This application needs an active internet connection to
    ↪ work</string>
38  <string name="warning_want_quit">Do you really want to quit ?</string>
39  <string name="info_secret_ok">Secret ok. Next checkpoint has been enabled</string>
40  <string name="info_secret_wrong">Sorry wrong secret. Guess again</string>
41  <string name="info_congratulations">Congratulations, you have finished this route</string>
42  <string name="finished">finished</string>
43  <string name="done">done</string>
44  <string name="open">open</string>
45  <string name="checkpoint_group_secret">Checkpoint Secret</string>
46  <string name="checkpoint_group_list">Checkpoint List</string>
47  <string name="checkpoint_group_map">Checkpoint Map</string>
48  <string name="google_maps_key">AIzaSyCNV6I75G-wopgeM3st0HeUiTzgWcINjy8</string>
49 </resources>

```