AlarmManager & Notifications

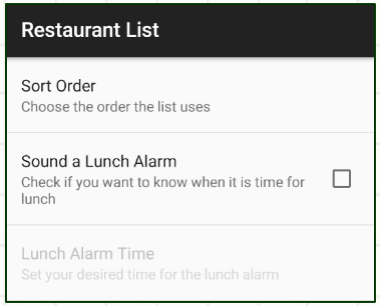
Today's Overview

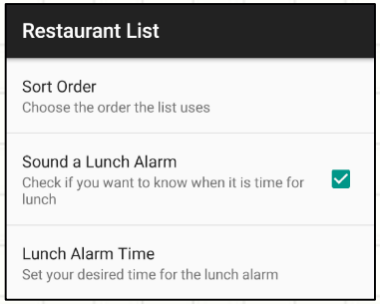
1. AlarmManager
2. Notifications

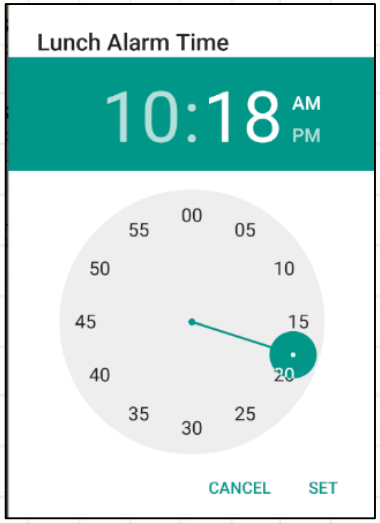
AlarmManager

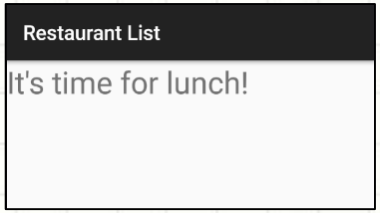
AlarmManager

- In first part of Practical 5 exercise, a Lunch Alarm is added to the “Setting” MENU item of Restaurant List app. It allows users to set the daily alert time for lunch









AlarmManager

- Let’s check what do we need to modify from the previous exercise to add alarm alert feature?
  - Model - Any change in Data Model?
  - View - Do you need to modify any of the user interface view?
  - Controller - Do you need to tell the Controller to do any thing new?

AlarmManager

- Model - NO

Since there is nothing change in data format and handling methods, the restaurant table Model and *Cursor* Model will remain unchanged

AlarmManager

- View - YES

there will be some changes involved

- ✓ *preferences.xml* layout is added with
  - *CheckBoxPreference* widget for alarm ON or OFF setting, and
  - *TimePreference* widget (customized *DialogPreference*) loaded with *TimePicker* for alarm time setting
- ✓ *alarm.xml* layout file for alert *View* display when alarm is triggered

AlarmManager

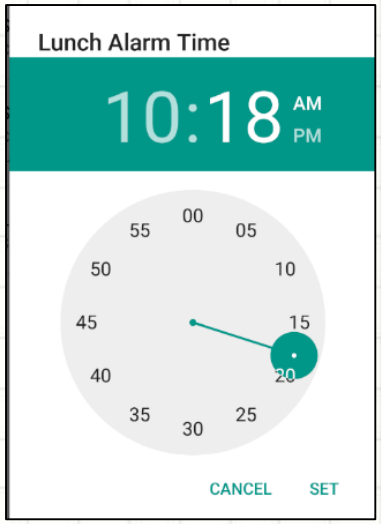
- Controller – More things will be done at here
  - ✓ *EditPreference* – add functionality to capture any change in *SharedPreference* XML data about ON/OFF alarm and alarm time modification. It will activate the *AlarmManager* using **setAlarm** or **cancelAlarm** method from *OnBootReceiver* (sub-class of *BroadcastReceiver*) to register or cancel alarm event

AlarmManager

- Controller – More things will be done at here.
  - ✓ *OnAlarmReceiver* (sub-class of *BroadcastReceiver*) – when alarm is triggered, the *AlarmManager* from Android system will activate the *OnAlarmReceiver* to run the *AlarmActivity* using **getPendingIntent** method to show the *alarm.xml* layout on display

AlarmManager

- Controller – More things will be done at here.
  - ✓ *TimePreference* (sub-class of *DialogPreference*) – to provide a dialog box to load Time Picker for lunch alarm setting



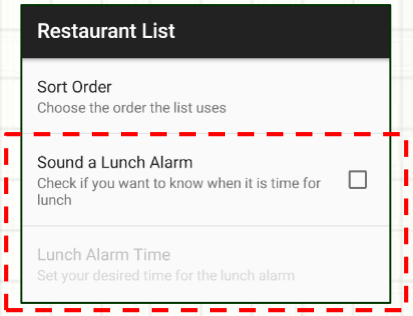
View – Preference & Alarm

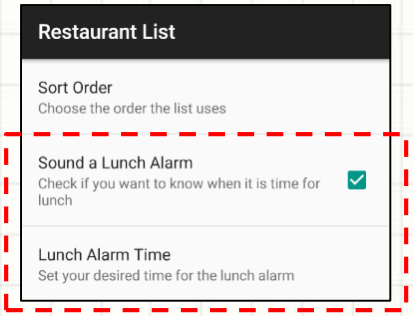
View

Preference Setting

- The *preferences.xml* layout is added with *CheckBoxPreference* and customized *DialogPreference* (*TimePreference*)

```
<PreferenceScreen
  xmlns:android="http://schemas.android.com/apk/res/android">
  <ListPreference
    android:key="sort_order"
    android:title="Sort Order"
    android:summary="Choose the order the list uses"
    android:entries="@array/sort_names"
    android:entryValues="@array/sort_clauses"
    android:dialogTitle="Choose a sort order" />
  <CheckBoxPreference
    android:key="alarm"
    android:title="Sound a Lunch Alarm"
    android:summary="Check if you want to know when it is time for lunch" />
  <sp.com.TimePreference
    android:key="alarm_time"
    android:title="Lunch Alarm Time"
    android:defaultValue="12:00"
    android:summary="Set your desired time for the lunch alarm"
    android:dependency="alarm" />
</PreferenceScreen>
```





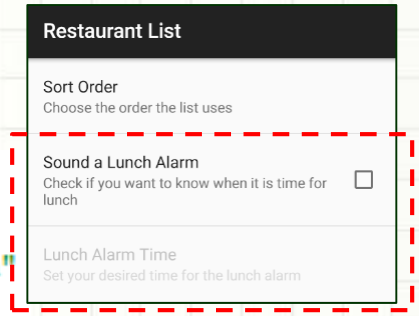
View

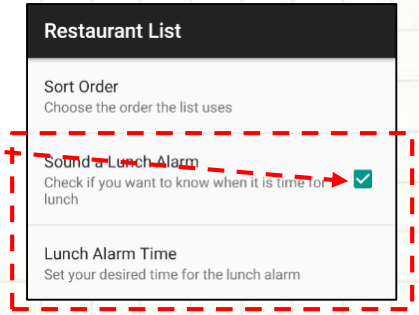
Preference Setting

- By having *android:dependency="alarm"*, the *TimePreference* widget will be disabled if *CheckBoxPreference* is unchecked

```
<CheckBoxPreference
    android:key="alarm"
    android:summary="Check if you want to know when it is time for lunch"
    android:title="Sound a Lunch Alarm" />

<com.sp.restaurantlist.TimePreference
    android:defaultValue="12:00"
    android:dependency="alarm"
    android:key="alarm_time"
    android:summary="Set your desired time for the lunch alarm"
    android:title="Lunch Alarm Time" />
```





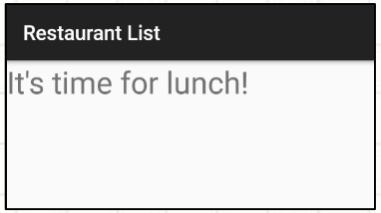
View

Alarm Alert View

- *alarm.xml* layout file is created in res/layout folder to display as an alert *View* when alarm sounds

```xml
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginBottom="0dp"
    android:layout_marginEnd="8dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="0dp"
    android:orientation="vertical"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    tools:layout_constraintBottom_creator="1"
    tools:layout_constraintLeft_creator="1"
    tools:layout_constraintRight_creator="1"
    tools:layout_constraintTop_creator="1">

    <TextView
        android:id="@+id/alarm_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="It's time for lunch!"
        android:textSize="30sp" />
</LinearLayout>
```



Controller – AlarmManager

Controller

- In order for the alarm to work, the Restaurant List app must
  - ✓ register request of alarm service to the *AlarmManager* of Android system (from app to system)
  - ✓ register the alarm service again if the Android device is reboot. To do that, the app will have a boot completed handler (*OnBootReceiver*) to do the task. It will be registered to the Android system for the boot completed alert

Controller

- In order for the alarm to work, the Restaurant List app must
  - ✓ register a handler for handling alarm set-off event (*OnAlarmReceiver*). This is because *AlarmManager* cannot direct interact with UI View
  - ✓ setup *EditPreferences.java* Controller to capture and register changes of *SharedPreference* XML data done by user, and activate the *OnBootReceiver.java* Controller to set or cancel alarm service

Controller

AndroidManifest.xml

- Register the *OnBootReceiver* (sub-class of *BroadcastReceiver*) to the Android system with Intent Filter for handling boot completed event

```
<receiver
    android:name=".OnBootReceiver"
    android:enabled="false" >
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED" />
    </intent-filter>
</receiver>
<receiver android:name=".OnAlarmReceiver" />
```

Controller

AndroidManifest.xml

- Set user permission to allow the Restaurant List app to receive “Boot Completed” alert from Android system

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.sp.restaurantlist" >

  <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
```

Controller

AndroidManifest.xml

- Register the *OnAlarmReceiver* (sub-class of *BroadcastReceiver*) to the Android system for handling the alarm set-off event from *AlarmManager*

```
<receiver
    android:name=".OnBootReceiver"
    android:enabled="false" >
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED" />
    </intent-filter>
</receiver>
<receiver android:name=".OnAlarmReceiver" />
```

Controller

EditPreferences

- This is the core part of the Controller where user preferences are captured
- This is done by monitoring the *SharedPreference* XML data – “alarm” or “alarm_time”
- If *CheckBoxPrerence* is “checked”, run the setAlarm() method in *OnBootReceiver* to set the alarm time and register the alarm service with *AlarmManager*

Controller

EditPreferences

- If *CheckBoxPrerence* is “checked”,
  - ✓ update the app package setting to enable the *OnBootReceiver* to handle boot complete event and save the *SharedPreference* XML data
  - ✓ run the setAlarm() method in *OnBootReceiver* to set the alarm time, register the alarm service and the alarm handler (*OnAlarmReceiver*) with *AlarmManager*

Controller

EditPreferences

- If *CheckBoxPrerence* is NOT “checked”,
  - ✓ update the app package setting to disable the *OnBootReceiver* to handle boot complete event
  - ✓ run the cancelAlarm() method in *OnBootReceiver* to cancel alarm service from the *AlarmManager*

Controller

EditPreferences

- SharedPreference Event Handler

```
SharedPreferences.OnSharedPreferenceChangeListener onChange
            = new SharedPreferences.OnSharedPreferenceChangeListener() {
    public void onSharedPreferenceChanged(SharedPreferences prefs,
            String key) {
        if ("alarm".equals(key)) {
            boolean enabled = prefs.getBoolean(key, false);
            int flag = (enabled ? PackageManager.COMPONENT_ENABLED_STATE_ENABLED
                    : PackageManager.COMPONENT_ENABLED_STATE_DISABLED);
            ComponentName component = new ComponentName(
                    EditPreferences.this, OnBootReceiver.class);

            getPackageManager().setComponentEnabledSetting(component, flag,
                    PackageManager.DONT_KILL_APP);

            if (enabled) {
                OnBootReceiver.setAlarm(EditPreferences.this);
            } else {
                OnBootReceiver.cancelAlarm(EditPreferences.this);
            }
        } else if ("alarm_time".equals(key)) {
            OnBootReceiver.cancelAlarm(EditPreferences.this);
            OnBootReceiver.setAlarm(EditPreferences.this);
        }
    }
};
```

CheckBoxPreference

TimePreference

Controller

EditPreferences

- CheckBoxPreference

Read the CheckBoxPreference state: Checked or Unchecked

Assign a value to flag according to CheckBoxPreference state

```
if ("alarm".equals(key)) {
    boolean enabled = prefs.getBoolean(key, false);
    int flag = (enabled ? PackageManager.COMPONENT_ENABLED_STATE_ENABLED
            : PackageManager.COMPONENT_ENABLED_STATE_DISABLED);
    ComponentName component = new ComponentName(
            EditPreferences.this, OnBootReceiver.class);

    getPackageManager().setComponentEnabledSetting(component, flag,
            PackageManager.DONT_KILL_APP);

    if (enabled) {
        OnBootReceiver.setAlarm(EditPreferences.this);
    } else {
        OnBootReceiver.cancelAlarm(EditPreferences.this);
    }
}
```

Enable or disable the app setting on boot completed handler according to CheckBoxPreference state

Set or cancel alarm service according to CheckBoxPreference state

Controller

EditPreferences

- TimePreference

```
} else if ("alarm_time".equals(key)) {
    OnBootReceiver.cancelAlarm(EditPreferences.this);
    OnBootReceiver.setAlarm(EditPreferences.this);
}
```

If time of TimePicker in TimePreference has been changed, the old alarm event needs to be cancelled from the AlarmManager and register a new alarm event to the AlarmManager with new time

Controller

OnBootReceiver

- The *BroadcastReceiver* sub-class will handle the tasks of registering or cancelling alarm service to *AlarmManager* and receiving Boot Complete event (when Android system has boot completed)

Controller
OnBootReceiver

```
public static void setAlarm(Context ctxt) {
  AlarmManager mgr=(AlarmManager)ctxt.getSystemService(Context.ALARM_SERVICE);
  Calendar cal=Calendar.getInstance();
  SharedPreferences prefs=PreferenceManager.getDefaultSharedPreferences(ctxt);
  String time=prefs.getString("alarm_time", "12:00");

  cal.set(Calendar.HOUR_OF_DAY, TimePreference.getHour(time));
  cal.set(Calendar.MINUTE, TimePreference.getMinute(time));
  cal.set(Calendar.SECOND, 0);
  cal.set(Calendar.MILLISECOND, 0);

  if (cal.getTimeInMillis()<System.currentTimeMillis()) {
    cal.add(Calendar.DAY_OF_YEAR, 1);
  }

  mgr.setRepeating(AlarmManager.RTC_WAKEUP, cal.getTimeInMillis(),
                   AlarmManager.INTERVAL_DAY,
                   getPendingIntent(ctxt));
}
```

Get the lunch alarm time. Add 24 hours if alarm has been set-off

Set the wake up time for the device and register the alarm alert event handler

Controller

OnBootReceiver

```
public static void cancelAlarm(Context ctxt) {
  AlarmManager mgr=(AlarmManager)ctxt.getSystemService(Context.ALARM_SERVICE);

  mgr.cancel(getPendingIntent(ctxt));
}

private static PendingIntent getPendingIntent(Context ctxt) {
  Intent i=new Intent(ctxt, OnAlarmReceiver.class);

  return(PendingIntent.getBroadcast(ctxt, 0, i, 0));
}

@Override
public void onReceive(Context ctxt, Intent intent) {
  setAlarm(ctxt);
}
```

Cancel the alarm service and remove the alarm alert event handler

Assign the “alarm alert event handler”

Receive the Boot Completed event from Android system and set the alarm service

Controller

TimePreference

- It is a *DialogPreference* sub-class
- It provides a dialog box with *TimePicker*
- When the dialog box is closed, the state changed will be captured by onDialogClosed() method. If “Set” button is pressed, the *SharedPreference* XML data will be saved in “alarm_time” which is defined in *preferences.xml* file

Controller

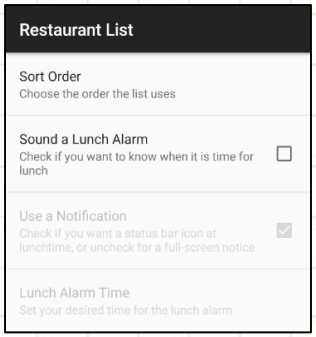
OnAlarmReceiver

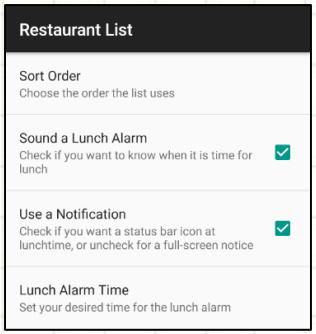
- The *BroadcastReceiver* sub-class will receive an alarm alert event from *AlarmManager* when an alarm is set-off. It will launch the *AlarmActivity* through Intent call to handle the event by displaying an *alarm.xml* layout in UI View

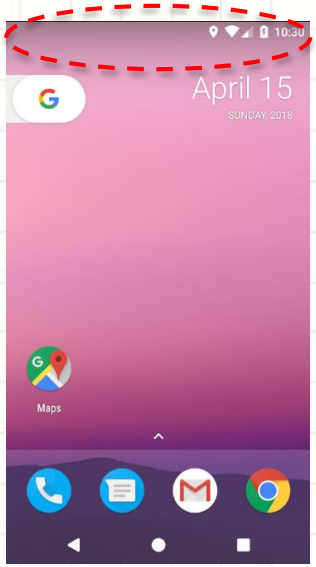
NOTIFICATIONS

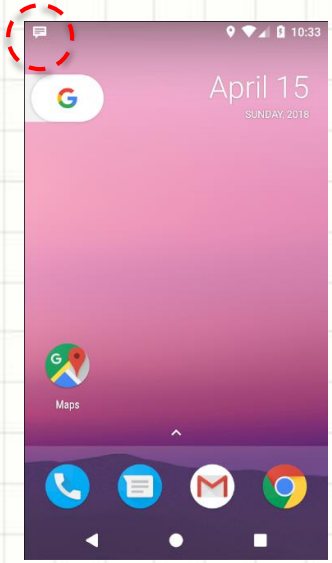
Notifications

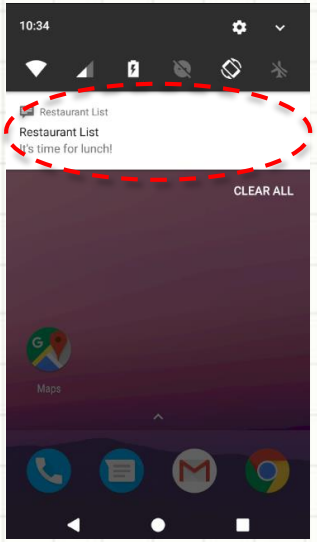
- In second part of Practical 5 exercise, System Notifications is used as alternated lunch alarm alert











Notifications

- Let's check what do we need to modify from the previous exercise to add System Notification as alternated alarm alert feature?
  - Model - Any change in Data Model?
  - View - Do you need to modify any of the user interface view?
  - Controller - Do you need to tell the Controller to do any thing new?

Notifications

- Model - NO

  Since there is nothing change in data format and handling methods, the restaurant table Model and *Cursor* Model will remain unchanged

Notifications

- View - YES

  an extra CheckBoxPreference widget is added to *preferences.xml* layout to monitor ON or OFF of alarm alert using System Notification

Notifications

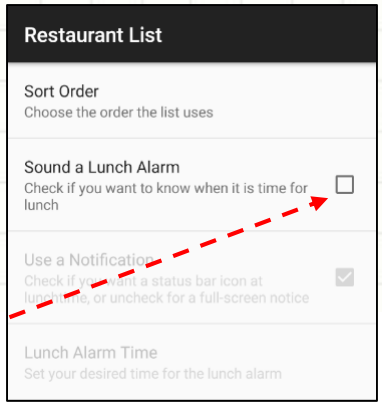
- Controller - YES

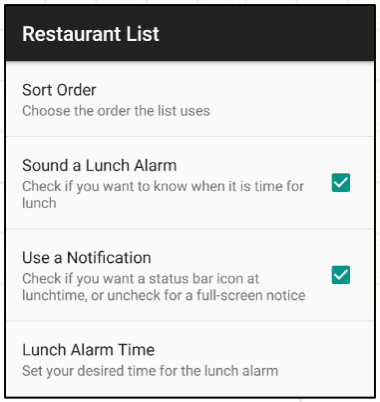
  Since *OnAlarmReceiver* is the Controller to handle the alarm alert event from *AlarmManager*, the onReceive() method is added with extra to monitor the status (ON or OFF) of *Notifications*

View – Notifications

View

- An extra widget is added to *preferences.xml* layout for user to ON or OFF Notifications
- The widget is enabled when CheckBoxPreference for Alarm is checked






```
<CheckBoxPreference
    android:defaultValue="true"
    android:dependency="alarm"
    android:key="use_notification"
    android:summary="Check if you want a status bar icon at lunchtime, or uncheck for a full-screen notice"
    android:title="Use a Notification" />
```

Controller - OnAlarmReceiver

Model

OnAlarmReceiver

- The onRecive() method in *OnAlarmReceiver* makes decision on showing the alarm alert
  - ✓using System Notifications or
  - ✓startActivity() method to activate the *AlarmActivity* for the *alarm.xml* layout display

Model

OnAlarmReceiver

```
public void onReceive(Context ctxt, Intent intent) {
    SharedPreferences prefs = PreferenceManager
            .getDefaultSharedPreferences(ctxt);
    boolean useNotification = prefs.getBoolean("use_notification", true);
    if (useNotification) {
        NotificationManager mgr = (NotificationManager) ctxt
                .getSystemService(Context.NOTIFICATION_SERVICE);
        PendingIntent i = PendingIntent.getActivity(ctxt, 0, new Intent(ctxt, AlarmActivity.class), 0);
        Notification note = new Notification.Builder(ctxt)
                .setContentTitle("Restaurant List")
                .setContentText("It's time for lunch!")
                .setSmallIcon(android.R.drawable.stat_notify_chat)
                .setContentIntent(i)
                .setAutoCancel(true).build();
        note.flags |= Notification.FLAG_AUTO_CANCEL;
        mgr.notify(NOTIFY_ME_ID, note);
    } else {
        Intent i = new Intent(ctxt, AlarmActivity.class);
        i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        ctxt.startActivity(i);
    }
}
```

Read the status of Notifications

Register Notifications service with NotificationManager

Use AlarmActivity for the alarm alert

END