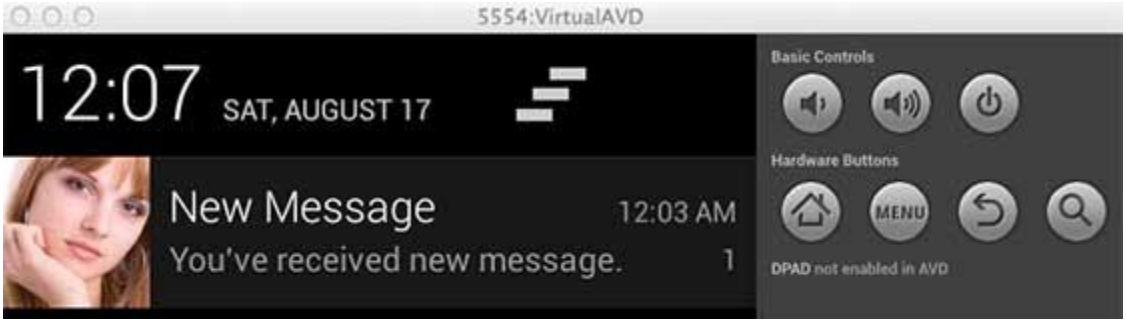
UNIT – VII

Notifications

Android **Toast** class provides a handy way to show users alerts but problem is that these alerts are not persistent which means alert flashes on the screen for a few seconds and then disappears.

For important messages to be given to the user, it is required to have more persistent method. A **notification** is a message you can display as an icon at the top of the device which we call notification bar or status bar.

To see the details of the notification, you will have to select the icon which will display notification drawer having detail about the notification. While working with emulator with virtual device, you will have to click and drag down the status bar to expand it which will give you detail as follows. This will be just **64 dp** tall and called normal view.



Notification Manager

2.1. Notification Manager

Android allows to put notification into the titlebar of your application. The user can expand the notification bar and by selecting the notification the user can trigger another activity.

An Activity or Service can initiate a status bar notification. Because an Activity can perform actions only while it is active and in focus, you should create your status bar notifications from a Service. This way, the notification can be created from the background, while the user is using

another application or while the device is asleep. To create a notification, you must use two classes: Notification and NotificationManager.

Use an instance of the Notification class to define the properties of your status bar notification, such as the status bar icon, the expanded message, and extra settings such as a sound to play. The NotificationManager is an Android system service that executes and manages all Notifications. You do not instantiate the NotificationManager. In order to give it your Notification, you must retrieve a reference to the NotificationManager with getSystemService() and then, when you want to notify the user, pass it your Notification object with notify().

2.2. Setting up Notifications

Notifications in Android are represented by the Notification class. To create notifications you use the NotificationManager class which can be received from the Context, e.g. an activity or a service, via the getSystemService() method.

```
NotificationManager notificationManager = (NotificationManager)
  getSystemService(NOTIFICATION_SERVICE);

Notification noti = new Notification.Builder(this)
        .setContentTitle("New mail from " + "test@gmail.com")
        .setContentText("Subject").setSmallIcon(R.drawable.icon)
        .setContentIntent(pIntent)
        .addAction(R.drawable.icon, "Call", pIntent)
        .addAction(R.drawable.icon, "More", pIntent)
        .addAction(R.drawable.icon, "And more", pIntent).build();
    NotificationManager notificationManager = (NotificationManager)
getSystemService(NOTIFICATION_SERVICE);
// hide the notification after its selected
    noti.flags |= Notification.FLAG_AUTO_CANCEL;

    notificationManager.notify(0, noti);
```

Introduction to Notifications

Notifications are used to alert users on some events that requires their attention. Notifications alert the users through various forms:

- Display a status bar icon
- Flash the LED
- Vibrate
- Ringtones

Notifying wih the status bar

Status bar notifications indicates some ongoing background services such as downloading or playing music or displays an alert/information.To see the notification user needs to open the notification drawer. Notifications are handled by Notification Manger in Android. Notification Manager is a system service used to manage notifications.

To create a status bar notification we need to use two classes, NotificationManager, Notification.

Lets see an simple example to display notification in status bar.

The NotificationManager is a system Service used to manage Notification. Get a reference to it using the getSystemService() method.

```
NotificationManager notificationManager =(NotificationManager)
getSystemService(NOTIFICATION_SERVICE);
```

Vibrating the phone

Vibrating Android device is found effective when you wanted to get notified on the incoming connection in various cases like, user cannot hear a ringtone or he wants to get notified silently. Its a very basic feature found in all mobile phones. Lets see how to vibrate an android device programatically. Its just two lines of code.

Vibrate pattern

public abstract void vibrate (long[] pattern, int repeat)

Pattern for vibration is nothing but an array of duration's to turn ON and OFF the vibrator in milliseconds. The first value indicates the number of milliseconds to wait before turning the vibrator ON. The next value indicates the number of milliseconds for which to keep the vibrator on before turning it off. Subsequent values, alternates between ON and OFF.

```
long pattern[]={0,100,200,300,400};
```

If you feel not to have repeats, just pass -1 for 'repeat'. To repeat patterns, just pass the index from where u wanted to start. I wanted to start from 0'th index and hence I am passing 0 to 'repeat'.

```
vibrator.vibrate(pattern, 0);
```

Android manifest permission for Vibration

Last but not the least add permission to Android manifest file.

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

Activity.java

```
importandroid.os.Bundle;
importandroid.os.Vibrator;
importandroid.app.Activity;
importandroid.content.Context;
importandroid.view.View;

publicclassMainActivityextends Activity {
public Vibrator vibrator;

@Override
protectedvoidonCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
 }

publicvoidstartVibrate(View v) {
long pattern[] = { 0, 100, 200, 300, 400 };
  vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
  vibrator.vibrate(pattern, 0);
 }

publicvoidstopVibrate(View v) {
  vibrator.cancel();
 }
}
```

➢ Blinking the lights

If you have a notification LED flashing, you know that your phone has something to tell you. But you don't necessarily know what. Light Flow allows you to set different LED colors for different

notifications. You could have a yellow light flash when you have a missed call, a green light when you have a text message, a blue light when you have a new email, and a red light when the phone is low on battery and needs to be plugged in. You can control which types of notifications produce a notification light – Light Flow supports notifications from over 550 different apps.