ANDROID platforma
Bluetooth, notifications & sensors

Mobilni i distribuirani informacioni sistemi

Mr Bratislav Predić

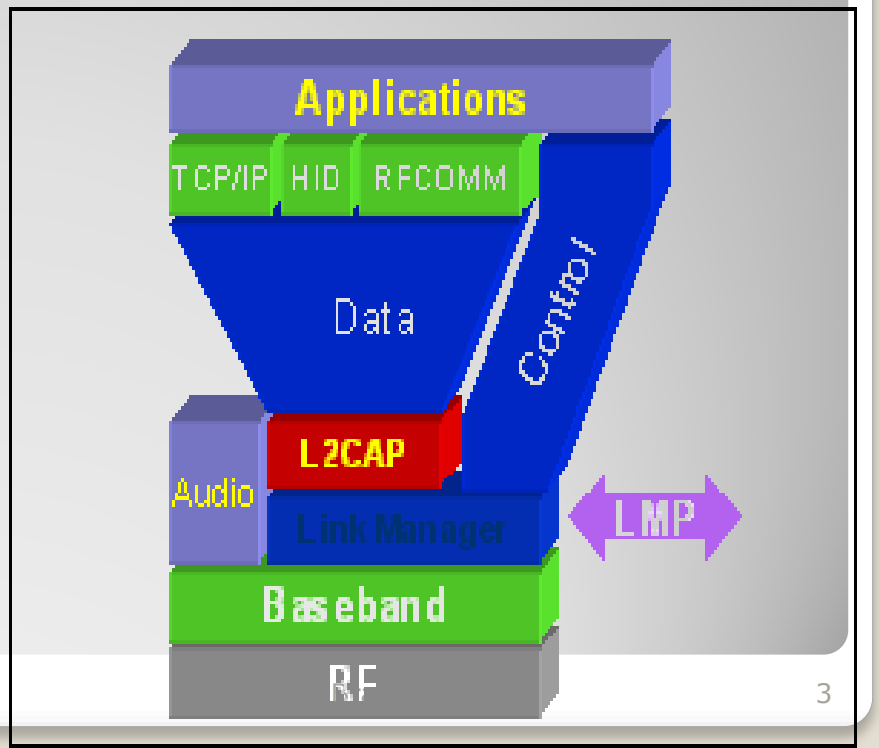
2012. godina

Bluetooth

- Bluetooth je radio komunikaciona tehnologija u PAN (Personal Area Network)
- Razvijena uglavnom zog industrije mobilnih uređaja za povezivanje različitih periferija koje korisnik nosi sa sobom
  - Slušalice, štampač, tastatura, eksterni (medicinski senzori)...
- Ograničen domet: tipično oko 10m
- Bluetooth rešava osnovni problem prethodno aktuelne IR tehnologije – line of sight
- Mreža bluetooth uređaja se često zove *piconet*

Bluetooth - piconet

- Relativno mala brzina – 721kbps
- U ISM (Industrial Scientific Medical) 2.45GHz frekventnom opsegu
- Komunikacija je peer-to-peer ali jedan uređaj je u piconet-u *master*
- *Piconet* se dinamički formira i rastura



Android i Bluetooth

- Android ključne klase za rad sa Bluetooth-om
  - *BluetoothAdapter* – lokalni Bluetooth radio
  - *BluetoothDevice* – udaljeni Bluetooth uređaj
  - *BluetoothServerSocket* – otvoren serversocket koji osluškuje dolazne konekcije
  - *BluetoothSocket* – komunikacioni endpoint
- Privilegije u manifestu
  - *android.permission.BLUETOOTH*
    Za uspostavljanje konekcije i prenos podataka
  - *android.permission.BLUETOOTH_ADMIN*
    Za uključivanje/isključivanje Bluetooth radija i za pretraživanje piconet-a za uređaje

Android i Bluetooth

- Koraci
  - Podesiti bluetooth
  - Pronaći udaljene bluetooth uređaje koji su već upareni ili skenirati okolinu u potrazi za uređajima
  - Povezati uređaje korišćenjem klase *BluetoothSocket*
  - Startovati prenos podataka
- Upravljanje Bluetooth adapterom korišćenjem klase *BluetoothAdapter*
  - *BluetoothAdapter.getDefaultAdapter();* Vraća *null* ako Bluetooth nije dostupan
- Proveriti da li je Bluetooth uključen
  - *BluetoothAdapter.isEnabled();* Vraća true ili false

Aktiviranje Bluetooth -a

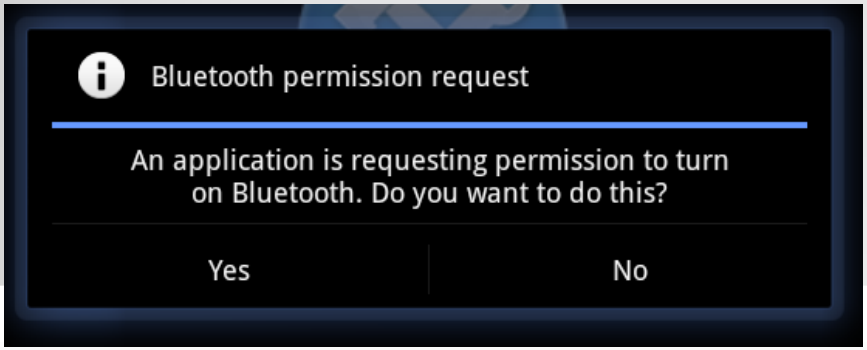
```
public class DataTransferActivity extends Activity {
  …
  public void onCreate(Bundle savedInstanceState) {
    …
    mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
    if (mBluetoothAdapter == null) {
      // No Bluetooth support
      finish();
    }
    if (!mBluetoothAdapter.isEnabled()) {
      Intent enableBluetoothIntent =
         new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
      startActivityForResult(
                          enableBluetoothIntent,REQUEST_ENABLE_BT);
    }

    …
```

- Proveravamo da li bluetooth adapter uopste postoji na mobilnom telefonu
- Ako postoji i nije uključen, aktiviramo ga

Aktiviranje Bluetooth -a

```
BluetoothAdapter BT = BluetoothAdapter.getDefaultAdapter();
if (!BT.isEnabled()) {
  //Taking user's permission to switch the bluetooth adapter On.
  Intent enableIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
  startActivityForResult(enableIntent, REQUEST_ENABLE_BT);
}
```

- Šaljemo *Intent* sa akcijom kako bi aktivirali Bluetooth
  - *BluetoothAdapter.ACTION_REQUEST_ENABLE*
- Ukoliko je Bluetooth isključen ovaj kod prikazuje *AlertDialog* u kome se korisnik pita da odobri uključivanje



Aktiviranje Bluetooth -a

- Po aktiviranju Bluetooth-a možemo pročitati podatke o lokalnom adapteru

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
  if(requestCode==REQUEST_ENABLE_BT && resultCode==Activity.RESULT_OK) {
    BluetoothAdapter BT = BluetoothAdapter.getDefaultAdapter();
    String address = BT.getAddress();
    String name = BT.getName();
    String toastText = name + " : " + address;
    Toast.makeText(this, toastText, Toast.LENGTH_LONG).show();
  }
}
```

- Discoverable stanje lokalnog adaptera

```
//Taking user's permission to make the device discoverable for 120 secs.
Intent discoverableIntent = new
        Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
startActivity(discoverableIntent);
```

  - I ova akcija dijalogom od korisnika traži odobrenje

Pretraživanje uređaja u blizini

- Upareni uređaji su uređaji sa kojima je već obavljana komunikacija i na kojima je korisnik prihvatio komunkaciju
  - *BluetoothAdapter.getBondedDevices();*
- Novi uređaji u blizini se pronalaze *discovery* procesom
  - *BluetoothAdapter.startDiscovery();*
  - Udaljeni uređaji moraju biti u *discoverable* stanju
- Započet proces otkrivanja je moguće i prekinuti
  - *BluetoothAdapter.cancelDiscovery();*
  - Proces otkrivanja zauzima puno resursa pa treba da traje što kraće

Pretraživanje uređaja u blizini

- Prikaz uparenih uređaja

```
private void selectServer() {
  Set<BluetoothDevice> pairedDevices =
                          mBluetoothAdapter.getBondedDevices();
  ArrayList<String> pairedDeviceStrings = new ArrayList<String>();
  if (pairedDevices.size() > 0) {
    for (BluetoothDevice device : pairedDevices) {
      pairedDeviceStrings.add(device.getName() + "\n" + device.getAddress());
    }
  }
  Intent showDevicesIntent = new Intent(this, ShowDevices.class);
  showDevicesIntent.putStringArrayListExtra("devices", pairedDeviceStrings);
  startActivityForResult(showDevicesIntent, SELECT_SERVER);
}
```

- Posebna aktivnost prikazuje listu uparenih uređaja
- U toj aktivnosti je potrebno prikazati uparene uređaja i startovati discovety proces

Pretraživanje uređaja u blizini

- Aktivnost za prikaz uređaja

```
public class ShowDevices extends ListActivity {
   …
   protected void onCreate(Bundle savedInstanceState) {
     …
     // search for more devices
     mBluetoothAdapter.startDiscovery();
     …
      // user selects one device
     BluetoothDevice device =
                mBluetoothAdapter.getRemoteDevice(/* mac addr String */);
     Intent data = new Intent();
     data.putExtra(BluetoothDevice.EXTRA_DEVICE, device);
     setResult(RESULT_OK, data);
     finish();
     …
```

- Discovery je asinhroni proces
- Pronalazak udaljenog uređaja kreira broadcast

Pretraživanje uređaja u blizini

- Po pronalasku udaljenog uređaja se šalje broadcast *Intent*
  - *BluetoothDevice.ACTION_FOUND*
- Podaci o pronađenom uređaju su u *Extra* u objektu *BluetoothDevice*

```
final BroadcastReceiver mReceiver = new BroadcastReceiver() {
  public void onReceive(Context context, Intent intent) {
    String action = intent.getAction();
    if (BluetoothDevice.ACTION_FOUND.equals(action)) {
      BluetoothDevice device = intent
                          .getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
      // update display
    }
  }
};
```

- Klijent i server se prepoznaju po istom UUID
  - 128-bit Universally Unique Identifier

Bluetooth prenos podataka

- Preko bluetooth-a za prenos podataka se najčešće koristi serijski prenos i data stream-ovi
- Server prvo kreira BluetoothServerSocket
  - *BluetoothAdapter*
    .listenUsingRfcommWithServiceRecord(name, uuid);
- Server osluškuje dolazne konekcije
  - BluetoothServerSocket.accept()
  - Blokirajući poziv vraća *BluetoothSocket* kada se uspostavi konekcija
- Svi blokirajući pozivi tipično idu u zaseban thread

Osluškivanje konekcije

```
class AcceptThread extends Thread {
  public AcceptThread(Handler handler) {
     …
     try {
        mServerSocket = mBluetoothAdapter
                  .listenUsingRfcommWithServiceRecord("Bluetooth Demo",
                                                      DataTransferActivity.APP_UUID);
     } catch (IOException e) {}
  }
  …
  public void run() {
    while (true) {
      try {
        mBluetoothSocket = mServerSocket.accept();
        manageConnectedSocket();
        mServerSocket.close();
            break;
      } catch (IOException e1) { … }
    }
  }
```

Klijentska strana komunikacije

- Klijent u Bluetooth komunikaciji kreira *BluetoothSocket* instancu sa
  - *BluetoothDevice .createRfcommSocketToServiceRecord(uuid);*
- Klijent povezuje kreirani socket
  - *BluetoothSocket.connect();*

```
public class ConnectThread extends Thread {
  …
  public ConnectThread(String deviceID, Handler handler) {
    mDevice = mBluetoothAdapter.getRemoteDevice(deviceID);
    try {
      mBluetoothSocket = mDevice.createRfcommSocketToServiceRecord(
                                                DataTransferActivity.APP_UUID);
     } catch (IOException e) { … }
  }
  …
```

Klijentska strana komunikacije

- Po uspostavljanju konekcije

```
 public void run() {
    mBluetoothAdapter.cancelDiscovery();
    try {
      mBluetoothSocket.connect();
      manageConnectedSocket();
    } catch (IOException connectException) { … }
  }

  private void manageConnectedSocket() {
    ConnectionThread conn = new ConnectionThread(mBluetoothSocket, mHandler);
    mHandler.obtainMessage(DataTransferActivity.SOCKET_CONNECTED, conn)
                                                          .sendToTarget();
    conn.start();
}
```

- *ConnectionThread* radi sa stream-ovima

Klijentska strana komunikacije

- Rad sa podacima kroz socket stream-ove

```
public class ConnectionThread extends Thread {
  …
  ConnectionThread(BluetoothSocket socket, Handler handler){
    super();
    mBluetoothSocket = socket;
    mHandler = handler;
    try {
      mInStream = mBluetoothSocket.getInputStream();
      mOutStream = mBluetoothSocket.getOutputStream();
    } catch (IOException e) { … }
  }
   …
  public void run() {
    byte[] buffer = new byte[1024];
    int bytes;
    while (true) {
      try {
        bytes = mInStream.read(buffer);
        String data = new String(buffer, 0, bytes);
        mHandler.obtainMessage(
               DataTransferActivity.DATA_RECEIVED,data).sendToTarget();
      } catch (IOException e) { break;  }
    }
  }
…
```

Obrada u aktivnosti - Handler

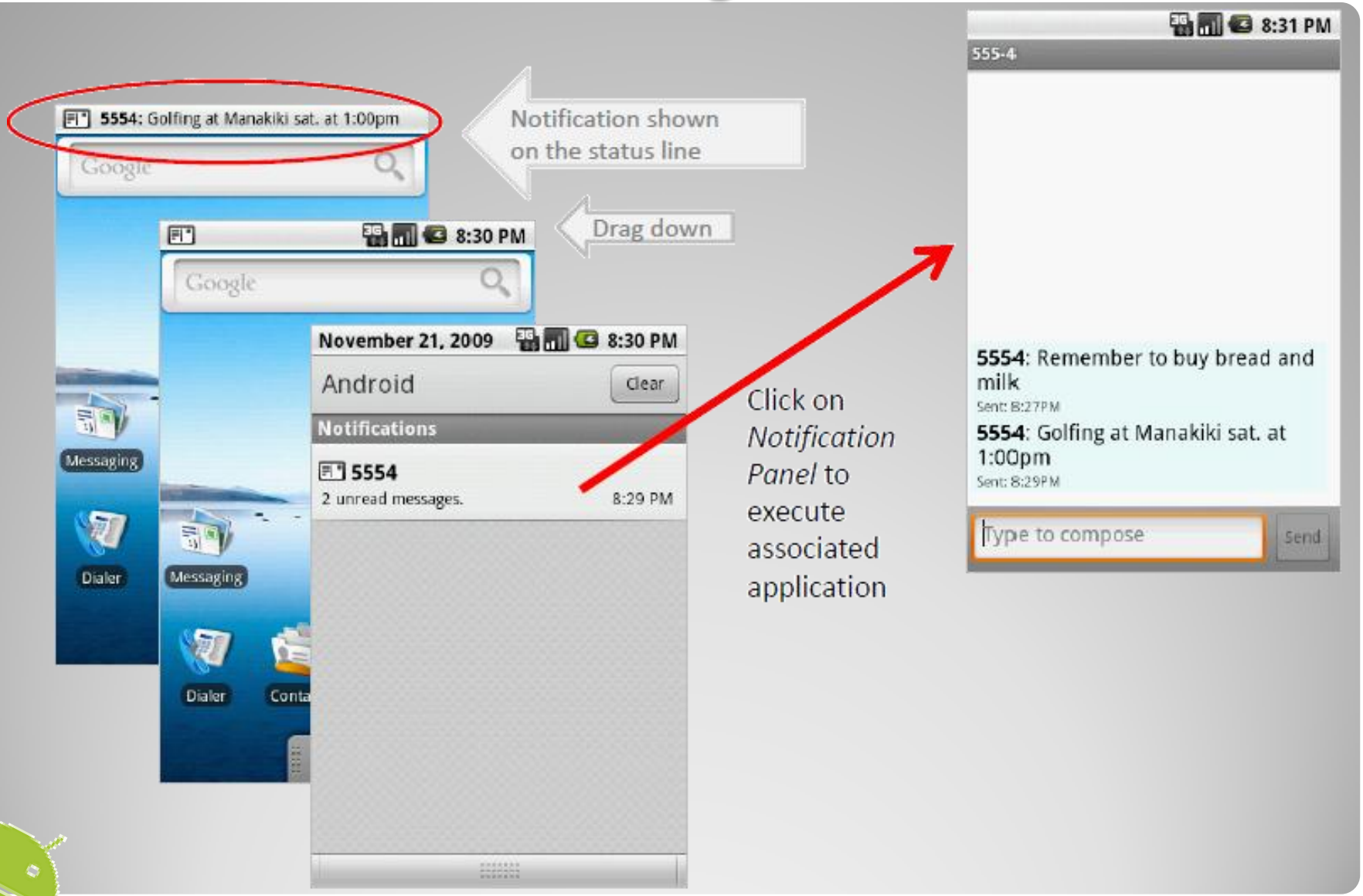
```
public void write(byte[] bytes) {
    try {
      mOutStream.write(bytes);
    } catch (IOException e) { … }
  }
}
```

```
// Handler in DataTransferActivity
  public Handler mHandler = new Handler() {
    public void handleMessage(Message msg) {
      switch (msg.what) {
        case SOCKET_CONNECTED: {
          mBluetoothConnection = (ConnectionThread) msg.obj;
          if (!mServerMode)
            mBluetoothConnection.write("this is a message".getBytes());
          break;
        }
        case DATA_RECEIVED: {
          data = (String) msg.obj;
          tv.setText(data);
          if (mServerMode)
           mBluetoothConnection.write(data.getBytes());
    }
   …
```

Android notifikacije

- Notifikacija na Android sistemu je kratka poruka koja se prikazuje na statusnoj liniji
- Tipično na ovaj način obaveštavamo korisnika o nekom događaju u aplikaciji
- Kada korisnik otvori *notification panel* ima mogućnost da klikne na notifikaciju i izvrši aktivnost povezanu za tu notifikaciju
- Tipičan primer za notifikaciju je dolazak SMS poruke
- Klikom na tu notifikaciju se startuje aplikacija za pregled SMS poruka

Android notifikacije

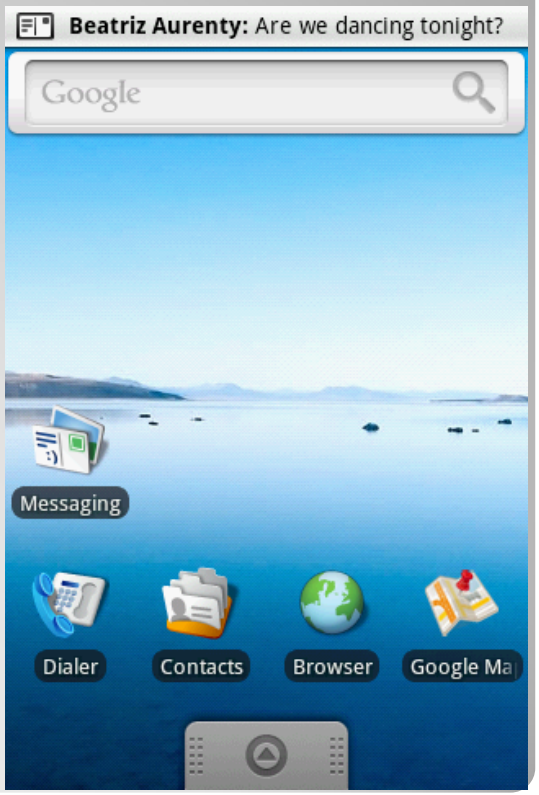


Android notifikacije

- Klasa *NotificationManager* obaveštava korisnika da se nešto desilo u pozadini
- Tri oblika notifikacija
  - Trajna ikona prikazana u status bar-u za koju se vezuje Intent i koju korisnik može da izabere
  - Uključeno trajno ili blinkanje LED indikatora na uređaju
  - Skrenuti pažnju korisnika blinkanjem pozadinskog osvetljenja, reprodukcijom zvuka ili vibracijom
- Referenca na *NotificationManager* se uzima kao sistemski servis
  - *notificationManager = (NotificationManager)*
    getSystemService*(Context.NOTIFICATION_SERVICE);*

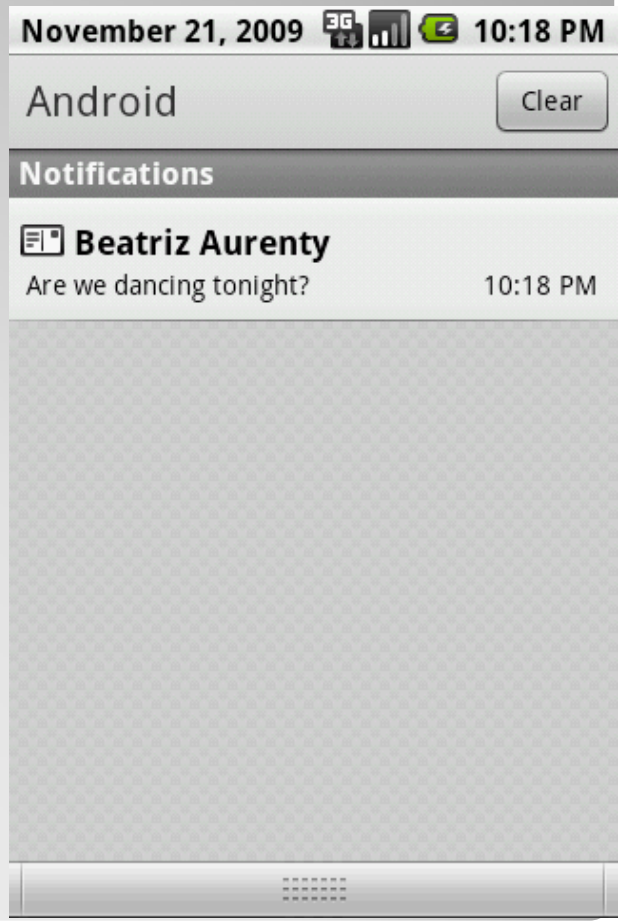
Android notifikacije

- Klasa *Notification* definiše kako će se prikazati notifikacija
- Konstruktor
  - *public Notification(int icon, CharSequencetickerText, long when);*
  - Argumenti: ikona, tekst i vreme koji će biti prikazani u status bar-u
- Prikaz notifikacije
  - *public void notify(int id, Notification notification)*
  - *Id* – jedinstvani identifikator notifikacije u okviru aplikacije



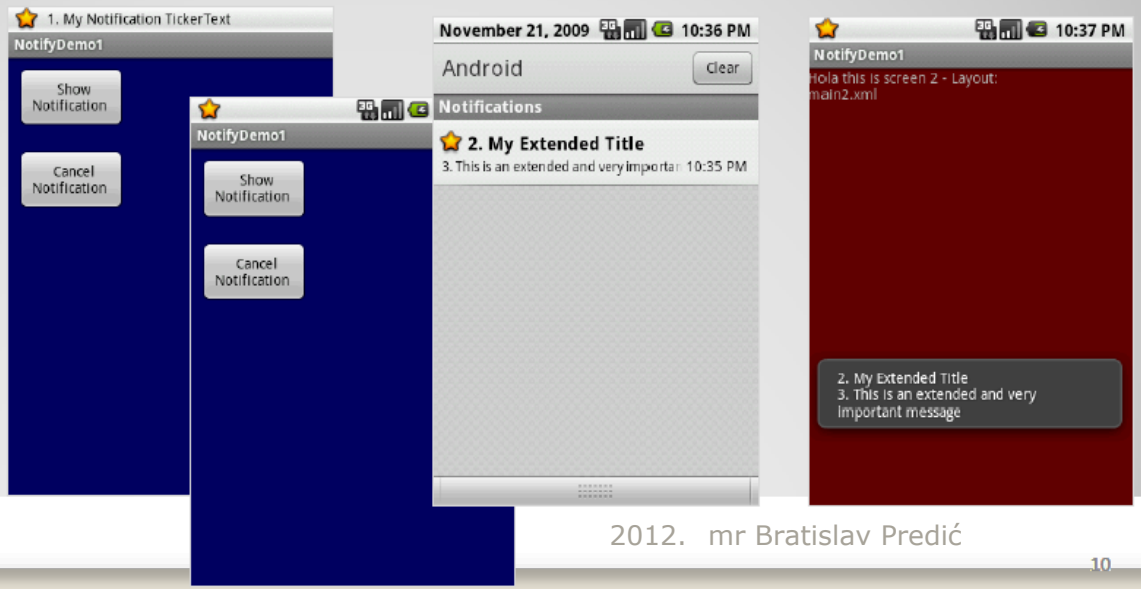
Android notifikacije

- Postavljanje detalja notifikacije
  - *public void setLatestEventInfo(*
    Context context,
    CharSequence contentTitle,
    CharSequence contentText,
    PendingIntent contentIntent)
- Argumenti
  - *context* – kontekst aplikacije
  - *contentTitle* – naslov prikazan u proširenom prikazu
  - *contentText* - tekst prikazan u proširenom prikazu
  - *contentIntent* – Intent koji se



Android notifikacije

- Možemo otkazati notifikaciju
  - *public void cancel (int id)*
  - *public void cancelAll( )*
- Id je prethodno definisan jedinstvani identifikator notifikacije
- Primer: prikazujemo notifikaciju koja startuje aktivnost



Android notifikacije

- Dva layout-a
  - Za aktivnost – main.xml
  - Za toast – main2.xml

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
android:id="@+id/myLinearLayout1"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="#ff000066"
android:orientation="vertical"
xmlns:android="http://schemas.android.com/apk/res/android" >


<Button
android:id="@+id/btnGo"
android:layout_width="106px"
android:layout_height="61px"
android:layout_margin="10px"
android:text=" Show Notification " >
</Button>
<Button
android:id="@+id/btnStop"
android:layout_width="106px"
android:layout_height="61px"
android:layout_margin="10px"
android:text=" Cancel Notification " >
</Button>
</LinearLayout>
```

main2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
android:id="@+id/main2LinLayout"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="#ff660000"
android:orientation="vertical"
xmlns:android="http://schemas.android.com/apk/res/
android"
>
<TextView
android:id="@+id/widget29"
android:layout_width="251px"
android:layout_height="69px"
android:text="Hola this is screen 2 - Layout:main2.xml"
>
</TextView>
</LinearLayout>
```

Android notifikacije

- Manifest i ikona

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="cis493.demos"
    android:versionCode="1"
    android:versionName="1.0">

    <application android:icon="@drawable/icon"
                 android:label="@string/app_name">

    <activity android:name=".NotifyDemo1"
          android:label="@string/app_name">
       <intent-filter>
          <action android:name="android.intent.action.MAIN" />
          <category android:name="android.intent.category.LAUNCHER" />
       </intent-filter>
    </activity>

    <activity android:name=".NotifyHelper" >
    </activity>

  </application>

  <uses-sdk android:minSdkVersion="4" />

</manifest>
```

btn_star_big_on_selected.png

Android notifikacije – glavna aktivnost

```
public class NotifyDemo1 extendsActivity {
  Button btnGo;
  Button btnStop;
  int notificationId= 1;
  NotificationManager notificationManager;

  @Override
  publicvoidonCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    btnGo= (Button)findViewById(R.id.btnGo);
    btnGo.setOnClickListener(newOnClickListener() {
        public void onClick(View v) {
          String serName = Context.NOTIFICATION_SERVICE;
          notificationManager= (NotificationManager)getSystemService(serName);
          int icon = R.drawable.btn_star_big_on_selected;
          String tickerText= "1. My Notification TickerText";
          long when = System.currentTimeMillis();
          Notification notification= newNotification(icon, tickerText, when);
          String extendedTitle= "2. My Extended Title";
          String extendedText=
                "3. This is an extended and very important message";
          Intent intent = newIntent(getApplicationContext(), NotifyHelper.class);
          intent.putExtra("extendedText", extendedText);
          intent.putExtra("extendedTitle", extendedTitle);
          PendingIntentlaunchIntent=
                PendingIntent.getActivity(getApplicationContext(),0,intent,0);
```

Android notifikacije

```
            notification.setLatestEventInfo(getApplicationContext(),
                            extendedTitle, extendedText, launchIntent);
            //trigger notification
            notificationId= 1;
            notificationManager.notify(notificationId, notification);
          }//click
    });
    btnStop= (Button)findViewById(R.id.btnStop);
    btnStop.setOnClickListener(newOnClickListener() {
         publicvoidonClick(View v) {
            //canceling a notification
            notificationId= 1;
            notificationManager.cancel(notificationId);
          }
    });
  }//onCreate
}//NotifyDemo1
```

Android notifikacije – druga aktivnost

- Ovu aktivnost korisnik startuje ako klikne na kreiranu notifikaciju

```
public class NotifyHelper extends Activity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main2);
    Intent myData= getIntent();
    // extract the extra-data in the Notification
    String msg= myData.getStringExtra("extendedTitle") + "\n"
        + myData.getStringExtra("extendedText");
    Toast.makeText(getApplicationContext(), msg, 1).show();
  }
}
```

- Title notifikacije se prikazuje u Toast-u

Android senzori

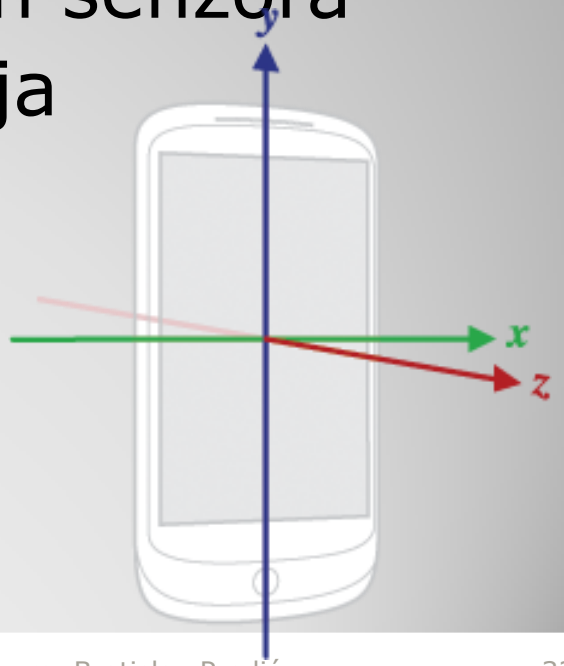
- Android mobilni telefoni imaju integrisane neke senzore koji omogućavaju opažanje okoline telefona
- Tipični primeri integrisanih senzora
  - Akceleracioni senzor sa 3 ose
  - Magnetni senzor sa 3 ose
  - Orijentacioni senzor
  - Senzor blizine
  - Senzor intenziteta svetlosti
  - Temperatirni senzor
- Sistemskom servisu koji kontroliše senzore se pristupa preko klase *SensorManager*

Android senzori

- Instanca klase *SensorManager* se pribavlja kao i svi sistemski servisi
  - *getSystemService(Context.SENSOR_SERVICE)*
- Konkretnom tipu senzora se pristupa
  - *SensorManager.getDefaultSensor(int type)*
- Tipovi senzora
  - *TYPE_ACCELEROMETER*
  - *TYPE_GRAVITY*
  - *TYPE_GYROSCOPE*
  - *TYPE_LIGHT*
  - *YPE_MAGNETIC_FIELD*
  - *TYPE_PRESSURE*
  - *TYPE_PROXIMITY*
  - *TYPE_TEMPERATURE*

Android senzori

- Senzor podatke šalje asinhrono
- Klasa *SensorEvent* sadrži podatke specifične za senzor
  - Tip, vreme uzorkovanja, preciznost, izmerene vrednosti
- Koordinatni sistem integrisanih senzora
- Koordinatni sistem se ne menja sa orijentacijom uređaja
  - Landscape/portrait



Android senzori

- Interfejs *SensorEventListener* deklariše callback metode
  - *void onAccuracyChanged(Sensor sensor, int accuracy)*
    Poziva se kada se promeni preciznost senzora
  - *void onSensorChanged(SensorEvent event)*
    Poziva se kada se uzorkuju kovi podaci sa senzora
- Ovaj listener se registruje korišćenjem *SensorManager* instance
  - *public boolean registerListener (*
    SensorEventListener listener, Sensor sensor, int rate)
    public void unregisterListener (
    SensorEventListener listener, Sensor sensor)

Android senzori - primer

```
public class SensorShowValuesActivity extends Activity
                                    implements SensorEventListener {
  …
  public void onCreate(Bundle savedInstanceState) {
    …
    mSensorManager =
         (SensorManager) getSystemService(SENSOR_SERVICE);
    mAccelerometer = mSensorManager
         .getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
}
 …
  protected void onResume() {
    super.onResume();
    mSensorManager.registerListener(this, mAccelerometer,
         SensorManager.SENSOR_DELAY_NORMAL);
  }
  protected void onPause() {
    mSensorManager.unregisterListener(this);
    super.onPause();
  }
```

- Treba implementirati i interfejs metode

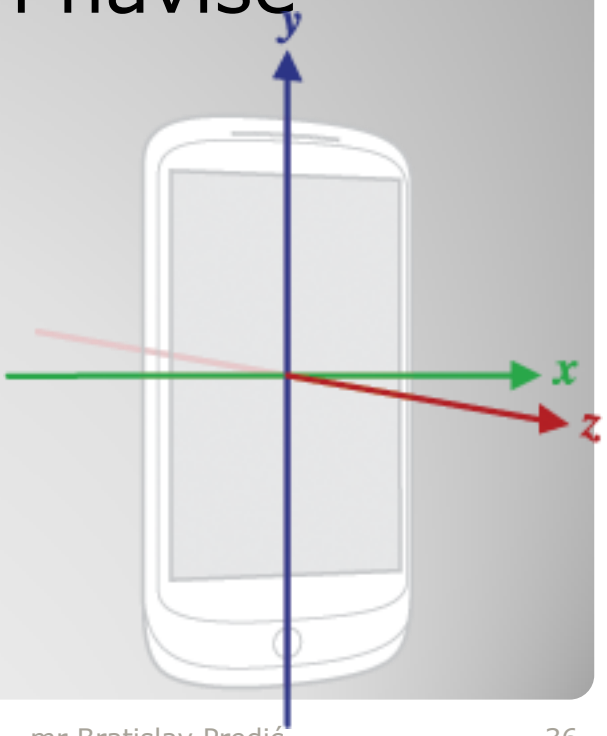
Android senzori - primer

```
 …
public void onSensorChanged(SensorEvent event) {
    if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
      long actualTime = System.currentTimeMillis();
      if (actualTime - mLastUpdate > 500) {
        mLastUpdate = actualTime;
        float x = event.values[0],  y = event.values[1], z = event.values[2];
        xView.setText(String.valueOf(x));
        yView.setText(String.valueOf(y));
        zView.setText(String.valueOf(z));
        …
      }
    }
```

- Izmerene vrednosti se nalaze u *values* nizu float vrednosti
- Njihov broj i tumačenje zavise od tipa senzora

Android senzori - primer

- Akceleracioni senzor ima tri ose, pa *values* niz ima tri elementa
- Vrednosti elemenata su ubrzanja po svakoj od osa
- Ako telefon miruje sa ekranom naviše
  - X = 0m/s^2
  - Y = 0m/s^2
  - Z = 9.81m/s^2
- Senzor je statičkog tipa što znači da detektuje i silu gravitacije



Android senzori - primer

- Često je potrebno filtrirati podatke sa akceleracionog senzora
- Low-pass filter
  - Eliminiše kratke i brze promene
  - Naglašava sile koje konstantno deluju (npr. Izdvajamo komponente sile gravitacije)
- High-pass filter
  - Izdvaja komponente sila koje deluju brzo i kratkotrajno
  - Eliminiše konstantne sile (npr. silu gravitacije)

Android senzori - primer

```
mAlpha = 0.9f;
// simple low-pass filter
float lowPass(float current, float filtered) {
  return mAlpha * current + (1.0f - mAlpha) * filtered;
}
// simple high-pass filter
float highPass(float current, float last, float filtered) {
  return mAlpha * (filtered + current - last);
}

x = event.values[0];
y = event.values[1];
z = event.values[2];

mLowPassX = lowPass(x, mLowPassX);
mLowPassY = lowPass(y, mLowPassY);
mLowPassZ = lowPass(z, mLowPassZ);

mHighPassX = highPass(x, mLastX, mHighPassX);
mHighPassY = highPass(y, mLastY, mHighPassY);
mHighPassZ = highPass(z, mLastZ, mHighPassZ);

mLastX = x;
mLastY = y;
mLastZ = z;
```