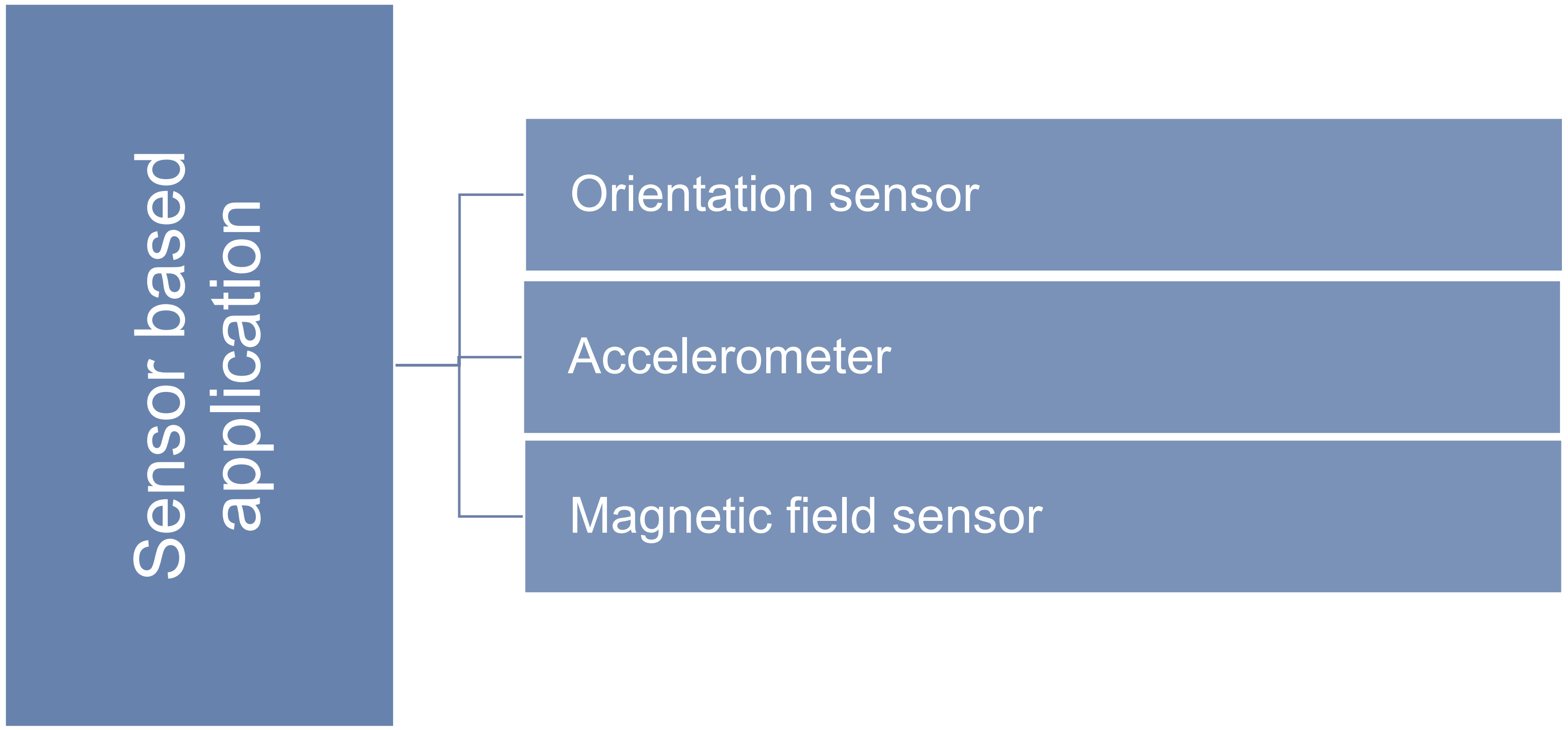
2019 级 - 本科

移动应用开发

2022

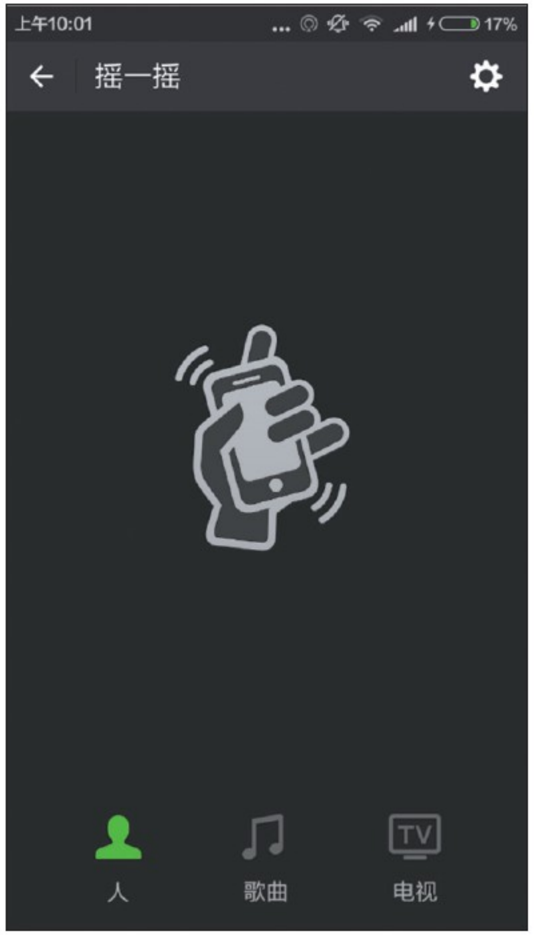
Sensor based application



Why Handler Message

- Sensors are used to monitor the alteration concerning position or postures as well as the located scenario

- Sensor objects
  - Specifies the attributes of the corresponding hardware
    - Type, name, manufacturer, precision, scope, etc.
    - Employed as Sensor.TYPE_<TYPE>



Types of sensors

Sensor.TYPE_ACCELEROMETER	获取 Android 设备在 X 、 Y 、 Z 三个方向上的加速度，单位为 m/s²
Sensor.TYPE_GRAVITY	返回一个三维向量，这个三维向量可显示重力的方向和强度，单位为 m/s² 。其坐标系统与加速度传感器的坐标系统相同
Sensor.TYPE_LINEAR_ACCELEROMETER	用于获取 Android 设备在 X 、 Y 、 Z 三个方向上不包括重力的加速度，单位为 m/s² 。
Sensor.TYPE_GYROSCOPE	用于获取 Android 设备在 X 、 Y 、 Z 这三个方向上的旋转速度，单位是弧度 / 秒。
Sensor.TYPE_LIGHT	用于获取 Android 设备所处外界环境的光线强度，单位是勒克斯 （ Lux 简称 lx ） 。
Sensor.TYPE_MAGNETIC_FIELD	用于获取 Android 设备在 X 、 Y 、 Z 三个方向上的磁场数据，单位是微特斯拉 （ μT ） 。
Sensor.TYPE_ORIENTATION	返回三个角度，这三个角度可以确定设备的摆放状态。
Sensor.TYPE_PRESSURE	用于获取 Android 设备所处环境的压力的大小，单位为毫巴 （ millibars ） 。
Sensor.TYPE_PROXIMITY	用于检测物体与 Android 设备的距离，单位是厘米。
Sensor.TYPE_AMBIENT_TEMPERATURE	用于获取 Android 设备所处环境的温度，单位是摄氏度。
Sensor.TYPE_RELATIVE_HUMIDITY	用于获取 Android 设备所处环境的相对湿度，以百分比的形式表示。
Sensor.TYPE_ROTATION_VECTOR	返回设备的方向，它表示为 X 、 Y 、 Z 三个轴的角度的组合。

Steps

- **调用 Context 的 getSystemService(Context.SENSOR_SERVICE) 方法获取 SensorManager 对象**
  - **SensorManager 是所有传感器的一个综合管理类，包括传感器的种类、采样率、精准度等**

    SensorManager sensorManager = (SensorManager)getSystemService(Context.SENSOR_SERVICE);
- **调用 SensorManager 的 getDefaultSensor(int type) 方法来获取指定类型的传感器**

  Sensor defaultPressure = sensorManager.getDefaultSensor(Sensor.TYPE_PRESSURE);
- **在 Activity 的 onResume() 方法中调用 SensorManager 的 registerListener() 方法为指定传感器注册监听器**

  sensorManager.registerListener(SensorEventListener listener, Sensor sensor, int rate)

  listener： 监听传感器事件的监听器。该监听器需要实现 SensorEventListener 接口。

  sensor： 传感器对象。

  rate： 指定获取传感器数据的频率

Frequency values of fetching sensor data

SensorManager.SENSOR_DELAY_FASTEST	尽可能快地获得传感器数据，延迟最小
SensorManager.SENSOR_DELAY_GAME	适合游戏的频率
SensorManager.SENSOR_DELAY_NORMAL	正常频率
SensorManager.SENSOR_DELAY_UI	适合普通用户界面的频率，延迟较大

SensorEventListener

- **As the core of utilizing sensors, two methods should be implemented**
  - **onSensorChanged(SensorEvent event) method**

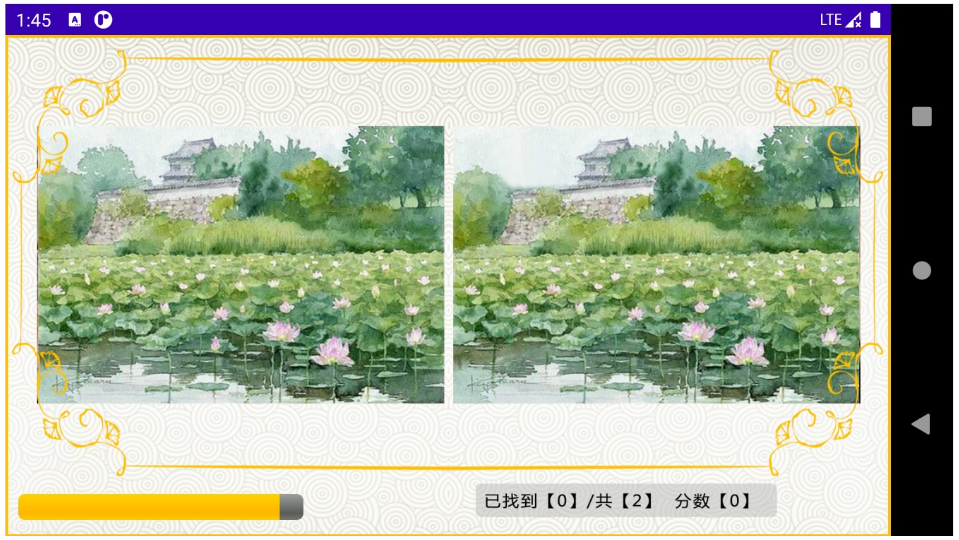
```
SensorManager sensorManager = (SensorManager)getSystemService(Context.SENSOR_SERVICE);

Sensor defaultPressure = sensorManager.getDefaultSensor(Sensor.TYPE_PRESSURE);

sensorManager.registerListener(SensorEventListener listener, Sensor sensor, int rate)
```

listener： 监听传感器事件的监听器。该监听器需要实现 SensorEventListener 接口。

sensor： 传感器对象。

rate： 指定获取传感器数据的频率

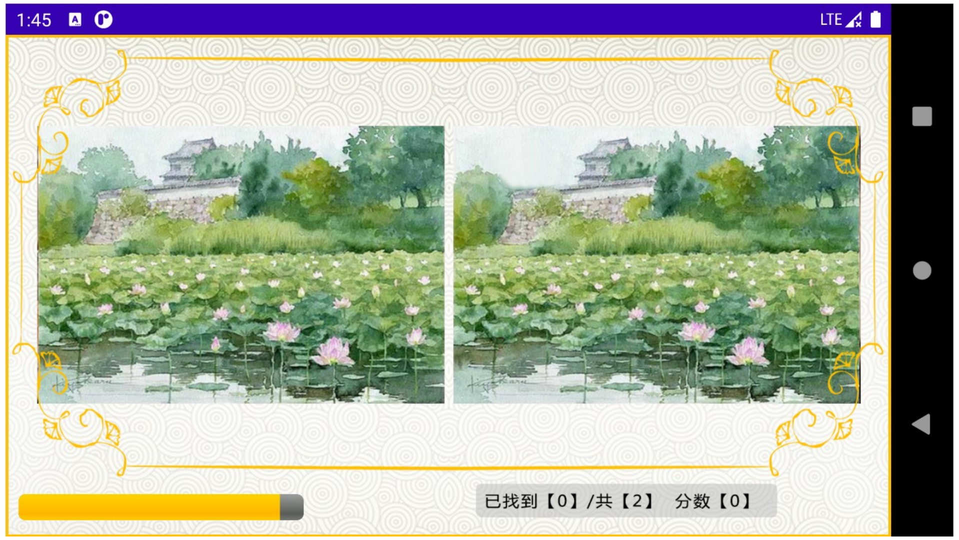
Small Demo



```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/bg"
    tools:context="com.example.c12.MainActivity">
    <ProgressBar
        android:id="@+id/timer"
        style="@android:style/Widget.ProgressBar.Horizontal"
        android:layout_width="@dimen/layout_width"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_marginBottom="@dimen/marginBottom"
        android:layout_marginLeft="@dimen/marginLeft"
        android:max="60" />
</RelativeLayout>
```

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.c12">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.C12">
        <activity android:name=".MainActivity"
            android:screenOrientation="landscape">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Small Demo



```java
public class MainActivity extends Activity {
    final int TIME = 60;
    final int TIMER_MSG = 0x001;
    private ProgressBar timer;
    private int mProgressStatus = 0;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        timer = (ProgressBar) findViewById(R.id.timer);
        handler.sendEmptyMessage(TIMER_MSG);
    }
    Handler handler = new Handler() {
        @Override
        public void handleMessage(Message msg) {
            if (TIME - mProgressStatus > 0) {
                mProgressStatus++;
                timer.setProgress(TIME - mProgressStatus);
                handler.sendEmptyMessageDelayed(TIMER_MSG, 1000);
            } else {
                Toast.makeText(MainActivity.this,
                        "Time is up. Game over", Toast.LENGTH_SHORT).show();
            }
        }
    };
}
```

Handler vs. Looper, and MessageQueue

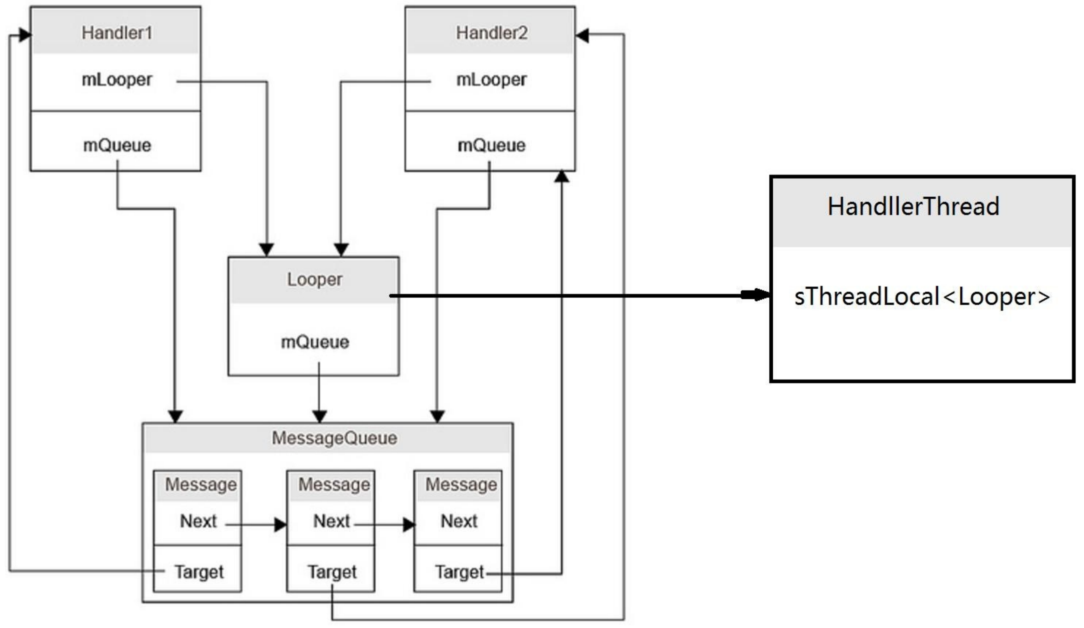
- To utilize Handler, there must be a Looper object within the current thread.
- To create Looper object
  - Within UI thread, system has created one Looper object. You could directly create Handler object. Then use Handler to send and handle messages.
  - Wthin sub thread, it is necessary to manually create Looper object, and then call loop() method to start Looper

Within one thread, there must be only one Looper and one MessageQueue and more than one Handlers. All the handlers can share the only Looper and the only MessageQueue.

    - Steps
      - Call prepare() method of Looper to create Looper object for the current thread. The MessageQueue object will be correspondingly created within the constructor of Looper object
      - Create the object of Handler’s subclass and override handlerMessage() method to handle the message from other thread(s)
      - Call loop() method of Looper to start Looper

Handler vs. Looper, and MessageQueue

- **Handler works with Message, Looper and MessageQueue**
  - **Message: the message object sent, received and handled by Handler**
  - **Looper: responsible for managing MessageQueue. One thread only has one Looper, and its loop() method is responsible for reading messages from MessageQueue. The read message is then returned to Handler for further treatment**
  - **MessageQueue: the container carrying messages. It employs FIFO mode to organize messages. When creating Looper object, the MessageQueue object will be created in the constructor**



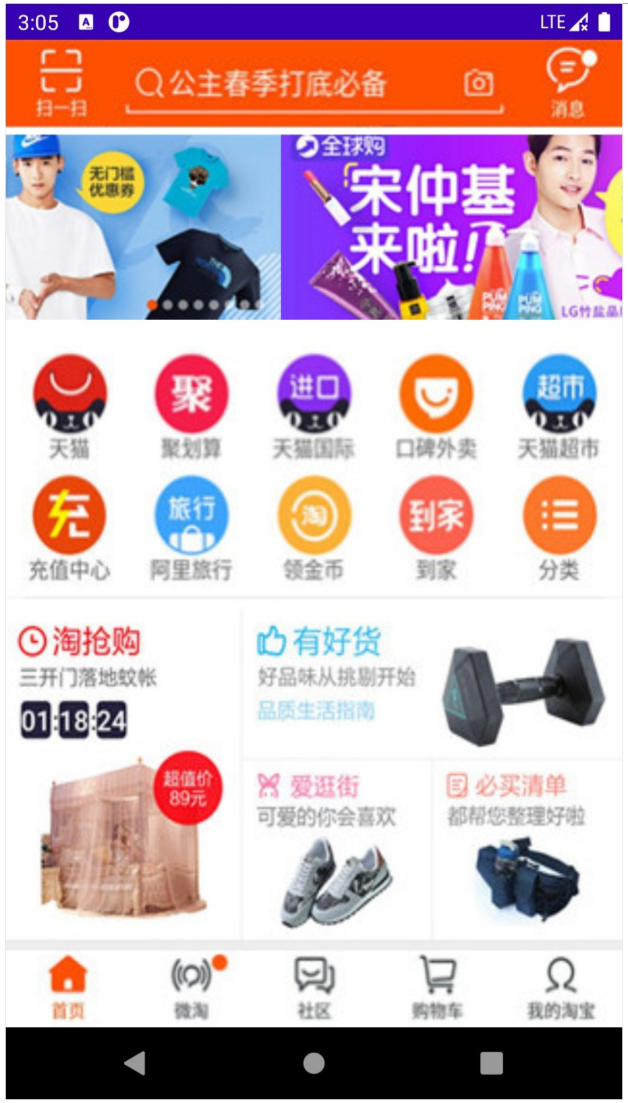
Message

- **Message is contained in MessageQueue, i.e., one MessageQueue could have one or more Message(s)**
- **Each Message object could be obtained by Message.obtain() or Handler.obtainMessage() methods**

arg1	int	To preserve integer data
arg2	int	To preserve integer data
obj	Object	To preserve the entity of Object type to be send to receiver
replyTo	Messenger	To specify the optional Messenger object stating the destination
what	int	To specify the customized message code for receivers to learn the content in the message

- ✓ Although Message has its own public constructor as default, it is usually suggested that developers should obtain empty Message object through calling Message.obtain() or Handler.obtainMessage() methods for saving resource.
- ✓ If one Message only carry simple information, i.e., int type, Message.arg1 and Message.arg2 attributes are highly recommended to carry information, which is more cost-effective than Bundle.
- ✓ Employ Message.what to label information as much as possible to easily handle messages in various modes.

Small Demo

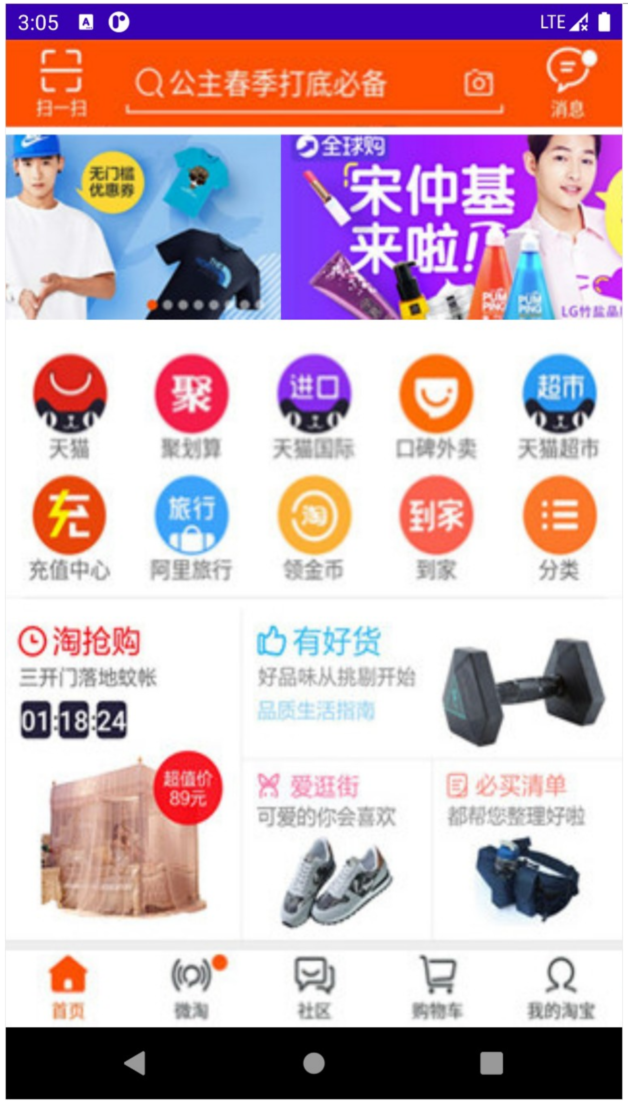


```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/bg"
    tools:context="com.example.carouseladv.MainActivity">
    <ViewFlipper
        android:id="@+id/viewFlipper"
        android:layout_width="match_parent"
        android:layout_height="130dp"
        android:layout_marginTop="@dimen/margin_top"/>
</RelativeLayout>
```

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
        android:duration="500"
        android:fromXDelta="100%p"
        android:toXDelta="0"/>
</set>
```

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate
        android:duration="500"
        android:fromXDelta="0"
        android:toXDelta="-100%p"/>
</set>
```

Small Demo



```java
public class MainActivity extends Activity {
    final int FLAG_MSG = 0x001;
    private ViewFlipper flipper;
    private Message message;
    private int[] images = new int[]{R.drawable.img1, R.drawable.img2, R.drawable.img3,
            R.drawable.img4, R.drawable.img5, R.drawable.img6, R.drawable.img7, R.drawable.img8};
    private Animation[] animation = new Animation[2];
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        flipper = (ViewFlipper) findViewById(R.id.viewFlipper);
        for (int i = 0; i < images.length; i++) {
            ImageView imageView = new ImageView(this);
            imageView.setImageResource(images[i]);
            flipper.addView(imageView);
        }
        animation[0] = AnimationUtils.loadAnimation(this, R.anim.slide_in_right);
        animation[1] = AnimationUtils.loadAnimation(this, R.anim.slide_out_left);
        flipper.setInAnimation(animation[0]);
        flipper.setOutAnimation(animation[1]);
        message=Message.obtain();
        message.what=FLAG_MSG;
        handler.sendMessage(message);
    }
    Handler handler = new Handler() {
        @Override
        public void handleMessage(Message msg) {
            if (msg.what == FLAG_MSG) {
                flipper.showPrevious();
            }
            message=handler.obtainMessage(FLAG_MSG);
            handler.sendMessageDelayed(message, 3000);
        }
    };
}
```

Looper

- One Looper object is used to build up a message looper for one thread in order to manipulate MessageQueue
- System automatically creaste Looper object for UI thread and open message looper
  - Sub thread doesn’t have message looper
  - It is valid for UI thread to create Handler object

```
Handler handler = new Handler();
//however, it is invalid for sub thread to create Handler object as above
```

  - To create Handler object in sub thread
    - Call prepare() method of Looper class to instantialize one Looper object
    - Create Handler object
    - Call loop() method of Looper class to start the Looper, then obtain messages

Looper

prepare()	To initialize Looper
loop()	To start Looper thread, the thread will obtain and handle message from Message Queue
myLooper()	To obtain the Looper object of current thread
getThread()	To get the thread(s) which the Looper object is belonging to
quit()	To terminate Looper

Thread vs. Handler

- Creating one Thread means creating a sub thread in program. However, Android doesn’t allow to update the UI components owned by UI Thread from Work Thread.
- Message Handler is thus introduced to solve the issue
  - Looper, Handler and Message associate with each other

Service

- Service is long-running background application component without UI
  - Can be invoked by some application component and shared to use even switching to another application
  - Components bind to Service and keep interactions with it, and meanwhile components could even execute IPC with Service
  - Such as music playing, continuously positioning

Category of Services

- **Started Service**
  - **When a component, e.g., an Activity, is invoking Service through calling startService() method. Service is at starting status, and it will be indefinitely running**
- **Bound Service**
  - **When a component is bound to Service thorugh calling bindService() method. Service is at binding status, and it can be bound from multiple components. Once all the components release their binding relations, Service is disposed.**

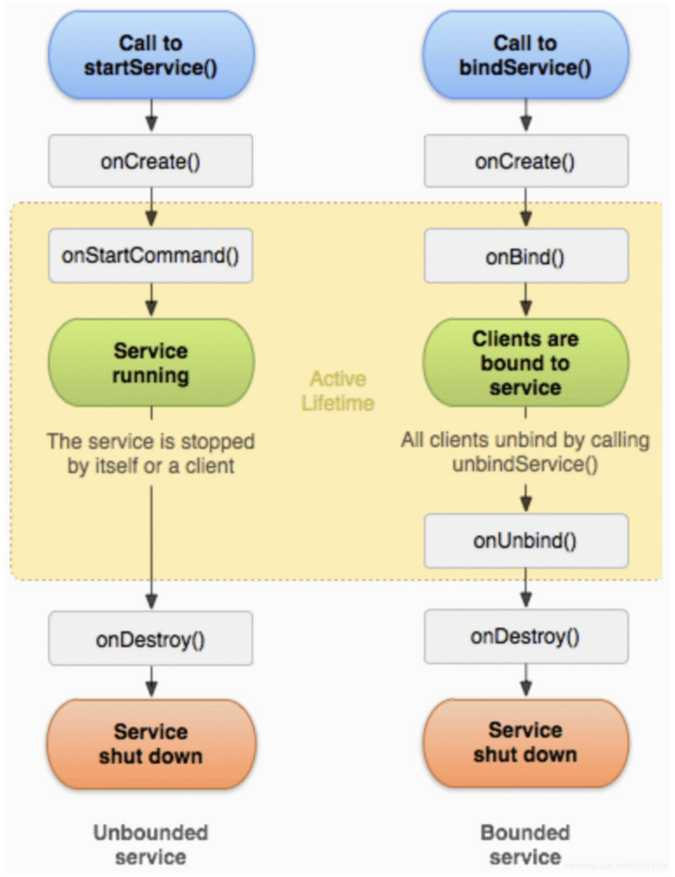
Started Service vs. Bound Service

Started Service	Bound Service
Invoked by calling startService() method	Bound by calling bindService()
Only starts without returning value	Sends request and return value
No relation between Service and the component starting the Service Service stays running even though the component is closed	Service is bound to component Service stops when the component is closed
Through calling-back onStartCommand() method, component is allowed to start Service	Through calling-back onBind() method, component is allowed to bind Service

- **A Service could be of both types**
  - **Component could employ Intent to utilize Service**
  - **Service could be configure as private to prevent the unwanted access**

Life cycle of Service

- **Starting Service with startService()**
  - **Indefinitely running**
  - **stopSelf() from Service or stopService() from other component to stop Service**
  - **Once stopped, Service is disposed**
- **Binding Service with bindService()**
  - **Service is created through calling bindService() by other components**
  - **Client communicates with Service through IBinder interface**
    - **Connection is closed through calling unbindService() method by client**
  - **Multiple clients could bind to the same Service. Once the clients unbind the Service, Service is disposed without being stopped**



Life cycle of Service

- The two paths are not independent with each other
  - Developers could bind the Service created through calling startService() method
  - For example,
    - Background running music Service could be started through employing the Intent containing music information
    - If users need to control player or obtain the current music information, they could call bindService() method to bind Activity to Service
    - Only stopService() or stopSelf() methods executed by client(s) could stop Service

Control Service

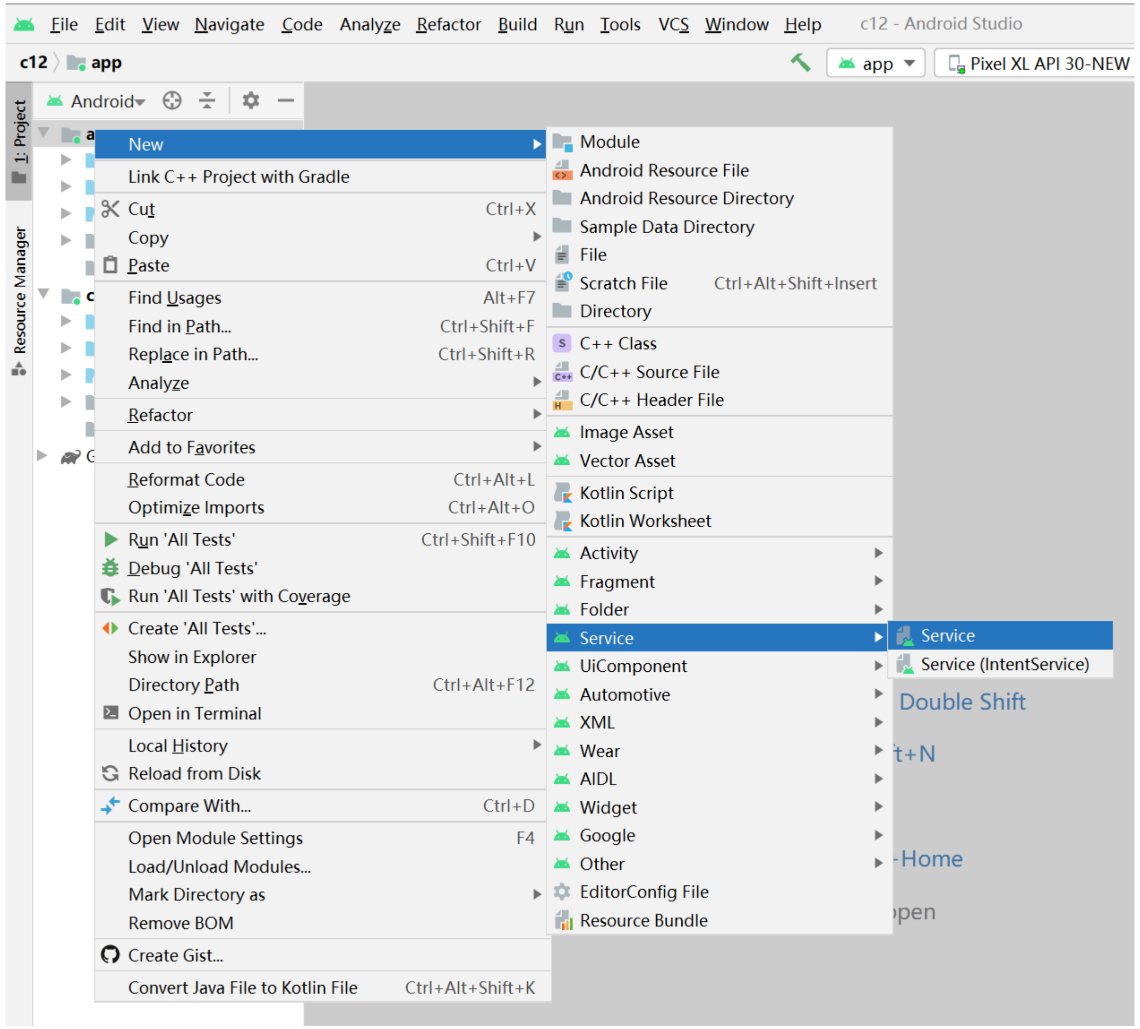
- To create Service,
  - Developers could create Service class, extend Service class, or extend its subclass
  - Within the create class, some callback methods should be overridden and the mechanism of binding components to Service should be provided

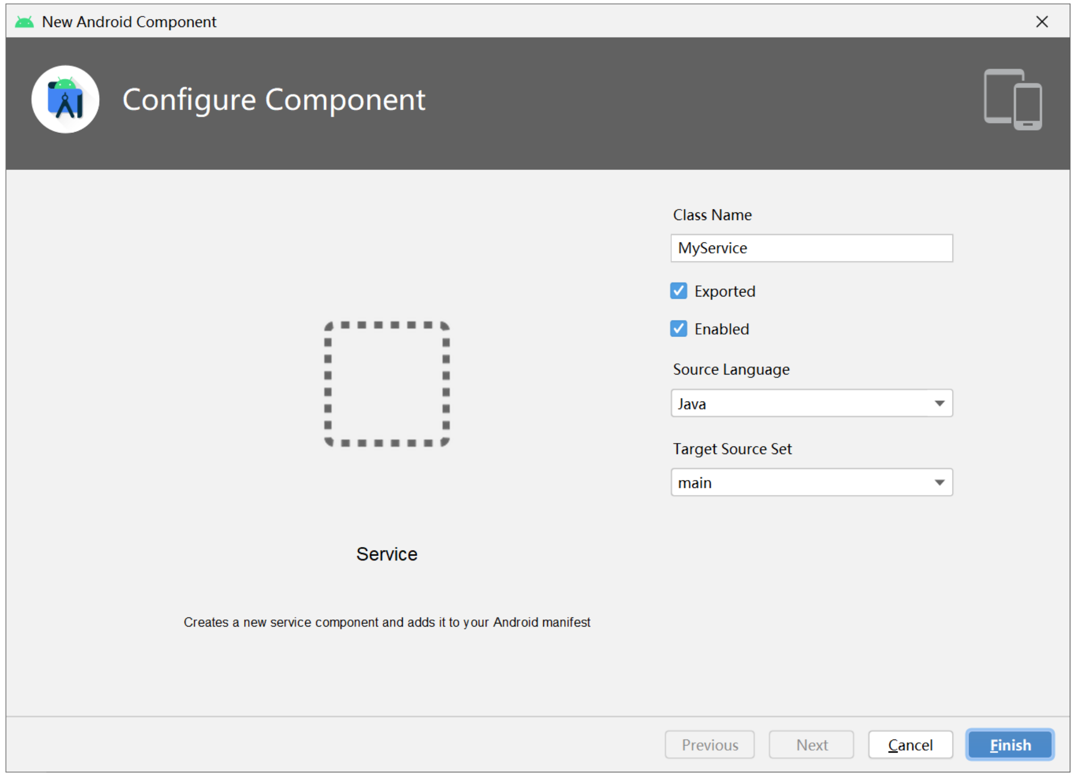
void onCreate()	When Service is created for the first time, system call this method for one time before calling onStartCommand() or onBind(). If Service is running, this method will not be called.
void onStartCommand(Intent intent, int flags, int startId)	When other component (e.g., Activity) is calling startService() to request starting Service, system call this method. Once this method is executed, Service will indefinitely keep running.
Ibinder onBind(Intent intent)	Sub class of Service must implement this method that returns an IBinder object. Application communicates with Service through employing the IBinder object
void onDestroy()	System calls this method to dispose Service

Utilizing Service

- Android offers two classes that could be extended by developers to create Service
  - Service class
    - The base class of all the Services
    - When inheriting/extending Service class, it is necessary to create new thread to execute the tasks made by Service
      - Otherwise, Service would take UI thread, which may affect Activity performance
  - IntentService class
    - Subclass of Service class
    - It employs one Worker thread to cope with all the starting request, which is feasible and good choice if not all the requests would be responded simultaneously
    - Only need to implement onHandleIntent() method that accept the Intent bringing in starting request

Example





Class Name: MyService

☑ Exported

☑ Enabled

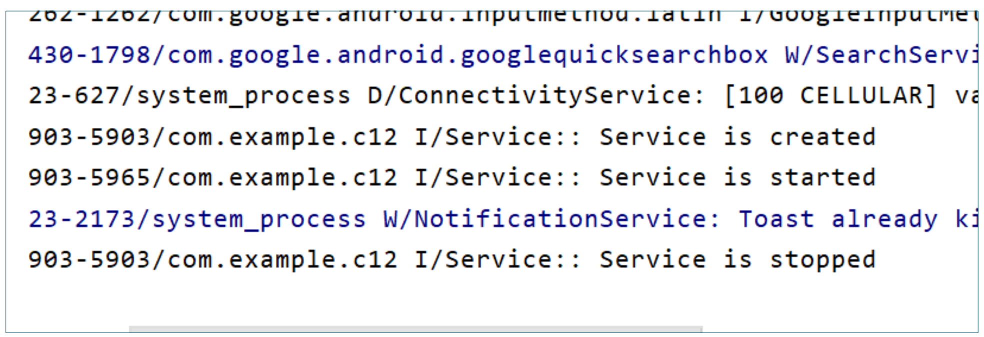
Source Language: Java

Target Source Set: main

Creates a new service component and adds it to your Android manifest

Example

```java
public class MainActivity extends Activity {
    final int TIME = 60;
    final int TIMER_MSG = 0x001;
    private ProgressBar timer;
    private int mProgressStatus = 0;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        timer = (ProgressBar) findViewById(R.id.timer);
        handler.sendEmptyMessage(TIMER_MSG);
    }
    Handler handler = new Handler() {
        @Override
        public void handleMessage(Message msg) {
            if (TIME - mProgressStatus > 0) {
                mProgressStatus++;
                timer.setProgress(TIME - mProgressStatus);
                handler.sendEmptyMessageDelayed(TIMER_MSG, 1000);
            } else {
                Toast.makeText(MainActivity.this,
                        "Time is up. Game over", Toast.LENGTH_SHORT).show();
                Intent intent = new Intent(MainActivity.this, MyService.class);
                startService(intent);
            }
        }
    };
}
```



```
262-1262/com.google.android.inputmethod.latin I/GoogleInputMet
430-1798/com.google.android.googlequicksearchbox W/SearchServi
23-627/system_process D/ConnectivityService: [100 CELLULAR] va
903-5903/com.example.c12 I/Service:: Service is created
903-5965/com.example.c12 I/Service:: Service is started
23-2173/system_process W/NotificationService: Toast already ki
903-5903/com.example.c12 I/Service:: Service is stopped
```

```java
public class MyService extends Service {
    public MyService() {
    }
    @Override
    public IBinder onBind(Intent intent) {
        // TODO: Return the communication channel to the service.
        throw new UnsupportedOperationException("Not yet implemented");
    }
    @Override
    public void onCreate() {
        Log.i("Service: ","Service is created");
        super.onCreate();
    }
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        new Thread(new Runnable(){
            @Override
            public void run() {
                Log.i("Service: ","Service is started");
                long endTime = System.currentTimeMillis() + 5*1000;
                while(System.currentTimeMillis() < endTime){
                    synchronized (this){
                        try{
                            wait(endTime - System.currentTimeMillis());
                        }catch(Exception e){
                            e.printStackTrace();
                        }
                    }
                }
                stopSelf();
            }
        }).start();
        return super.onStartCommand(intent, flags, startId);
    }
    @Override
    public void onDestroy() {
        Log.i("Service: ", "Service is stopped");
        super.onDestroy();
    }
}
```

Example

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.c12">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.C12">
        <service
            android:name=".MyService"
            android:enabled="true"
            android:exported="true"></service>
        <activity
            android:name=".MainActivity"
            android:screenOrientation="landscape">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

android:enabled

To specify whether Service could be instantialized

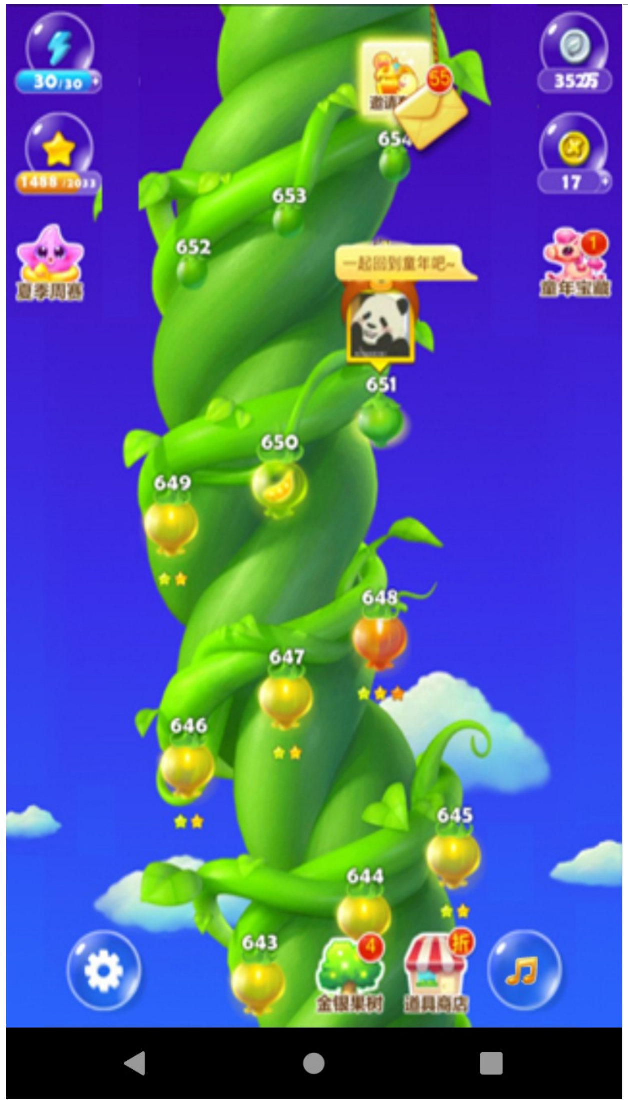
- ✓ Only if enable attributes within <application> and <service> are both set as true, Service is available and could be instantialized.
- ✓ If either is false, Service will be forbidden.

android:exported

To specify whether other components could call Service or communicate with Service

- ✓ If set as false, there is only one component or the app with the same User ID starting the Service or being bound to the Service.
- ✓ The default value of android:exported is depending on the condition whether Service has Intent Filter
  - ✓ Without filter, default value is false
  - ✓ One or more filters, default value is true

Small Demo

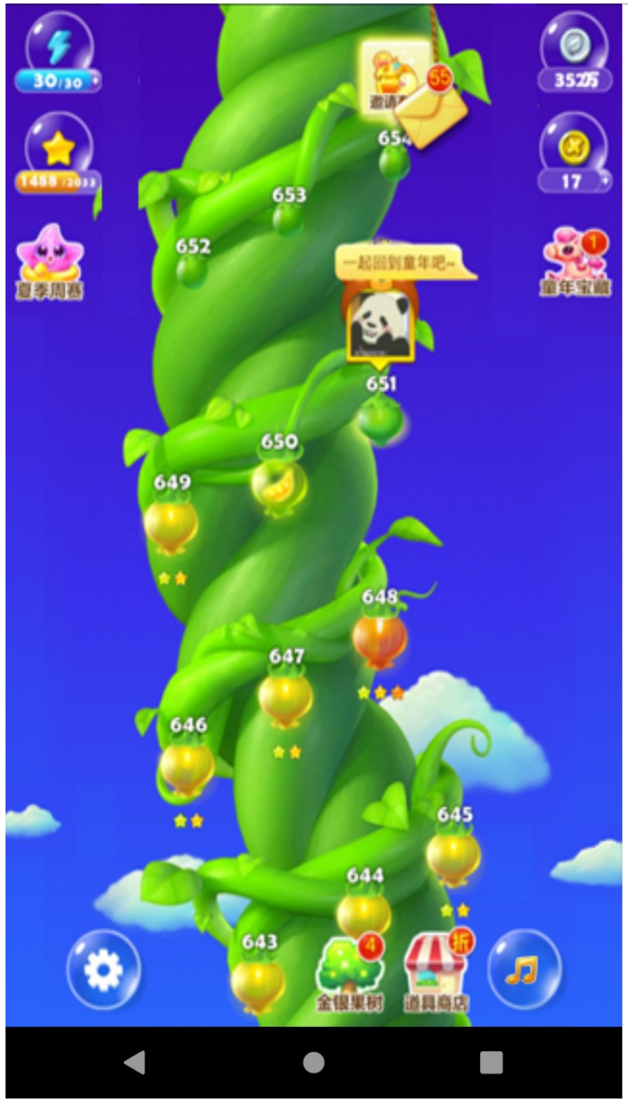


```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/bg"
    tools:context="com.example.serviceproject.MainActivity">
    <ImageButton
        android:id="@+id/btn_play"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/play"
        android:background="@color/btn_bg"
        android:layout_marginTop="636dp"
        android:layout_marginLeft="333dp"/>
</RelativeLayout>
```

Small Demo

```
public class MusicService extends Service {
    public MusicService() {
    }
    static boolean isplay;
    MediaPlayer player;
    @Override
    public IBinder onBind(Intent intent) {
        // TODO: Return the communication channel to the service.
        throw new UnsupportedOperationException("Not yet implemented");
    }
    @Override
    public void onCreate() {
        player = MediaPlayer.create(this, R.raw.music);
    }
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        if(!player.isPlaying()){
            player.start();
            isplay = player.isPlaying();
        }
        return super.onStartCommand(intent, flags, startId);
    }
    @Override
    public void onDestroy() {
        player.stop();
        isplay = player.isPlaying();
        player.release();
        super.onDestroy();
    }
}
```

Small Demo



```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);
        ImageButton btn_play = (ImageButton) findViewById(R.id.btn_play);
        btn_play.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (MusicService.isplay == false) {
                    startService(new Intent(MainActivity.this, MusicService.class));
                    ((ImageButton) v).setImageDrawable(getResources().getDrawable(R.drawable.play, null));
                } else {
                    stopService(new Intent(MainActivity.this, MusicService.class));
                    ((ImageButton) v).setImageDrawable(getResources().getDrawable(R.drawable.stop, null));
                }
            }
        });
    }
    @Override
    protected void onStart() {
        startService(new Intent(MainActivity.this, MusicService.class));
        super.onStart();
    }
}
```

Bound Service

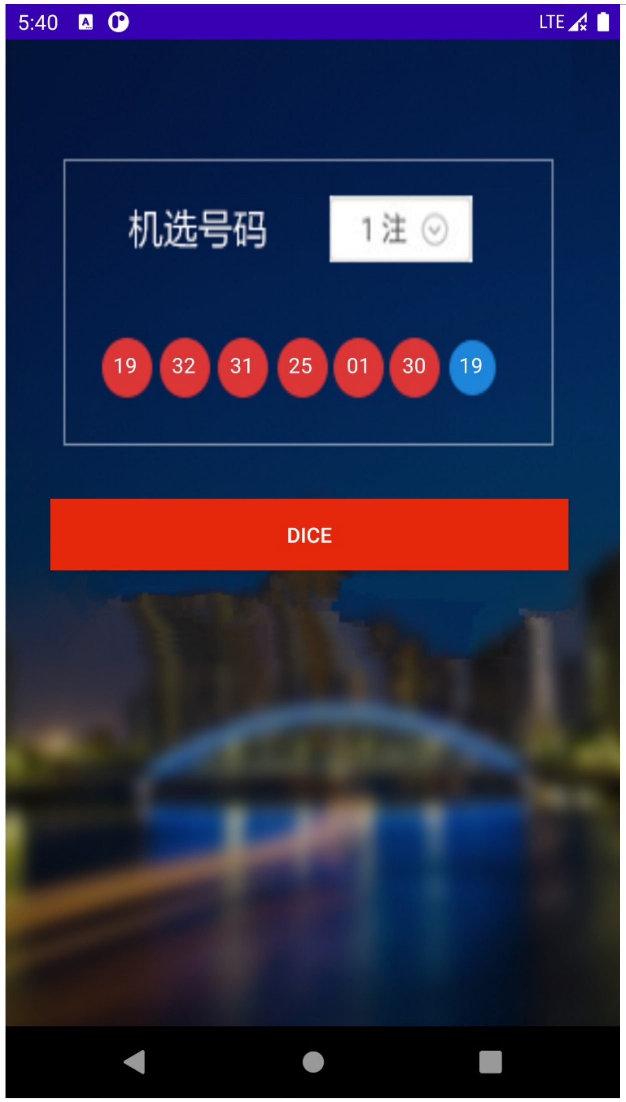
- **Components could employ bindService() method to bind themselves to one Service**
- **If Service is only used for the current application instead of IPC, i.e., client and Service are both located within the same application and process, developers could create customized Binder class to clients for their accessing Service**

bindService(Intent service, ServiceConnection conn, int flags)

- ✓ service: specifies the target Service to be invoked by Intent
- ✓ conn: to listen to the connection between accessors and Service
- ✓ flags: specifies whether to automatically create Service. 0 means no, otherwise, set as BIND_AUTO_CREATE

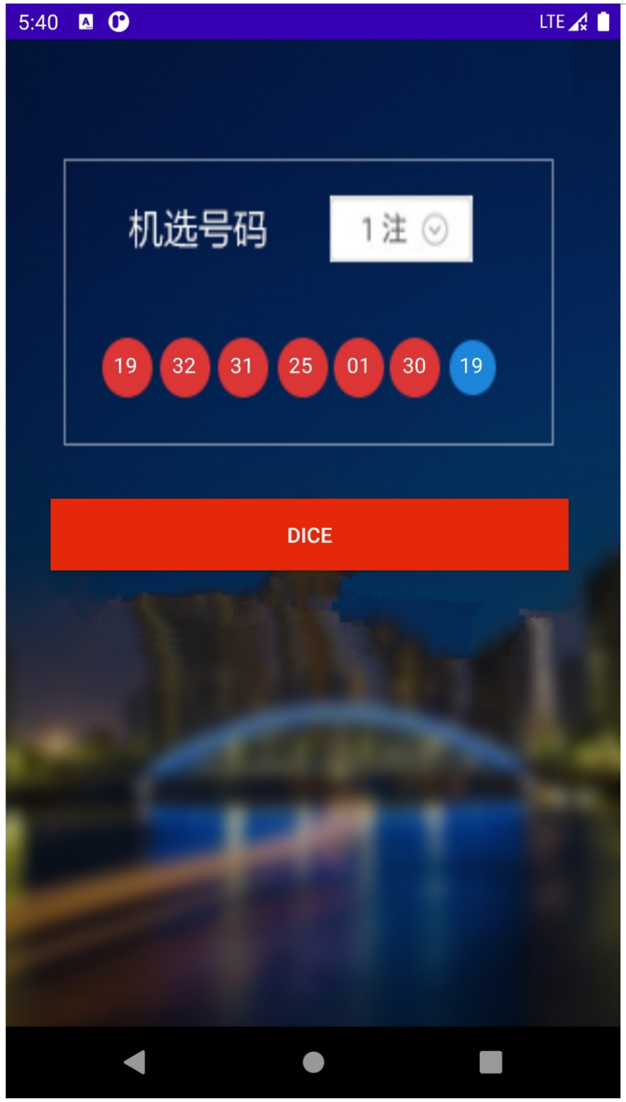
- **After calling bindService() method, Android system call onBind() method of Service to return IBinder object for the further communication with Service**
  - **Only Activity, Service and ContentProvider are able to be bound to Service, but BroadcastReceiver can not.**

Demo



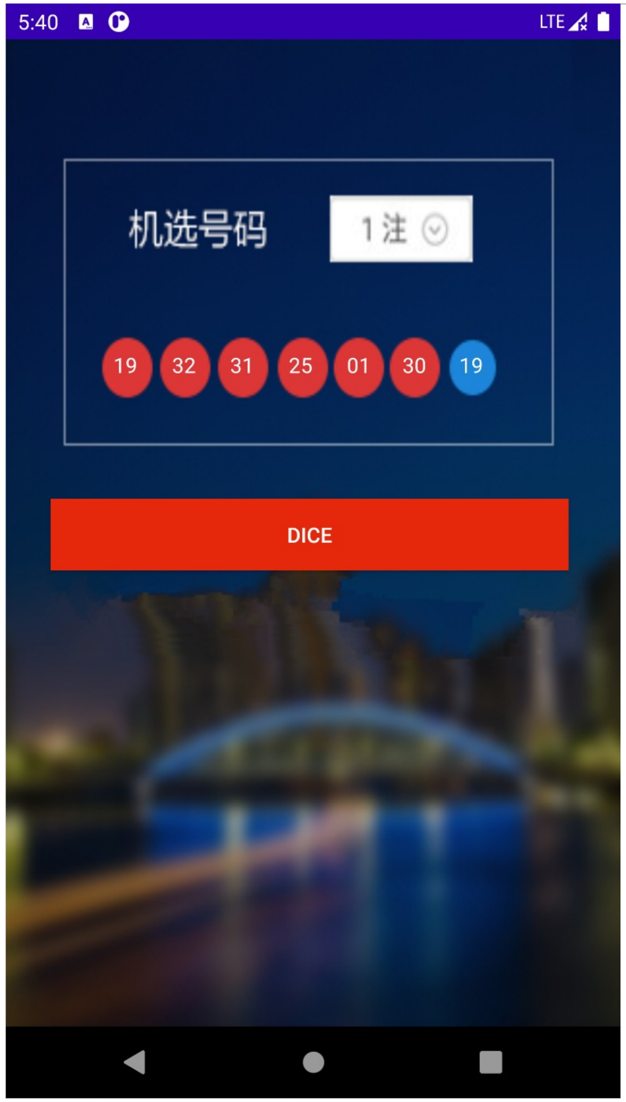
```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/bg"
    tools:context="com.example.randomlyselectingnumber.MainActivity">
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="@dimen/textView1_marginLeft"
        android:layout_marginTop="@dimen/textView_marginTop"
        android:textColor="@color/textColor"/>
    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="@dimen/textView2_marginLeft"
        android:layout_marginTop="@dimen/textView_marginTop"
        android:textColor="@color/textColor"/>
    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="@dimen/textView3_marginLeft"
        android:layout_marginTop="@dimen/textView_marginTop"
        android:textColor="@color/textColor"/>
    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="@dimen/textView4_marginLeft"
        android:layout_marginTop="@dimen/textView_marginTop"
        android:textColor="@color/textColor"/>
    <TextView
        android:id="@+id/textView5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="@dimen/textView5_marginLeft"
        android:layout_marginTop="@dimen/textView_marginTop"
        android:textColor="@color/textColor"/>
    <TextView
        android:id="@+id/textView6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="@dimen/textView6_marginLeft"
        android:layout_marginTop="@dimen/textView_marginTop"
        android:textColor="@color/textColor"/>
    <TextView
        android:id="@+id/textView7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="@dimen/textView7_marginLeft"
        android:layout_marginTop="@dimen/textView_marginTop"
        android:textColor="@color/textColor"/>
    <Button
        android:id="@+id/btn"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView1"
        android:layout_marginTop="@dimen/btn_marginTop"
        android:text="Dice"
        android:background="@color/btn_Color"
        android:textColor="@color/textColor"
        android:layout_marginLeft="@dimen/btn_marginLeft"
        android:layout_marginRight="@dimen/btn_marginRight"/>
</RelativeLayout>
```

Demo



```java
public class BinderService extends Service {
    public BinderService() {
    }
    @Override
    public IBinder onBind(Intent intent) {
        // TODO: Return the communication channel to the service.
        return new MyBinder();
    }
    public class MyBinder extends Binder {
        public BinderService getService(){
            return BinderService.this;
        }
    }
    public List getRandomNumber() {
        List resArr = new ArrayList();
        String strNumber="";
        for (int i = 0; i < 7; i++) {
            int number = new Random().nextInt(33) + 1;
            if (number<10) {
                strNumber = "0" + String.valueOf(number);
            } else {
                strNumber=String.valueOf(number);
            }
            resArr.add(strNumber);
        }
        return resArr;
    }
    @Override
    public void onDestroy() {
        super.onDestroy();
    }
}
```

Demo



```java
public class MainActivity extends Activity {
    BinderService binderService;
    int[] tvid = {R.id.textView1, R.id.textView2, R.id.textView3, R.id.textView4, R.id.textView5,
            R.id.textView6, R.id.textView7};
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button btn_random = (Button) findViewById(R.id.btn);
        btn_random.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                List number = binderService.getRandomNumber();
                for (int i = 0; i < number.size(); i++) {
                    TextView tv = (TextView) findViewById(tvid[i]);
                    String strNumber = number.get(i).toString();
                    tv.setText(strNumber);
                }
            }
        });
    }
    @Override
    protected void onStart() {
        super.onStart();
        Intent intent = new Intent(this, BinderService.class);
        bindService(intent, conn, BIND_AUTO_CREATE);
    }
    @Override
    protected void onStop() {
        super.onStop();
        unbindService(conn);
    }
    private ServiceConnection conn = new ServiceConnection() {
        @Override
        public void onServiceConnected(ComponentName name, IBinder service) {
            binderService = ((BinderService.MyBinder) service).getService();
        }
        @Override
        public void onServiceDisconnected(ComponentName name) {
        }
    };
}
```