# Android Activities i Layouts

**Što je aktivnost?**

Aktivnost je samo jedan ekran. Zaslon glavnog izbornika je jedna aktivnost, drugi zaslon je druga aktivnost. Aktivnost je zaslon koji korisniku prikazuje neke informacije ili zahtijeva određeni unos od korisnika. Aktivnost upravlja interakcijom s korisnikom.

Općenito, svaki zaslon u aplikaciji je aktivnost koja je podklasa klase aktivnosti definirane u paketu android.app koji se nalazi kao dio Android SDK-a.

**Što je Layout?**

Layout (izgled) definira kako vaša aktivnost izgleda. Vaša aktivnost prikazuje tekstualne okvire, oznake, padajuće izbornike, slike itd. Kako su ti objekti raspoređeni i prikazani na zaslonu upravlja datoteka Layout. Ova datoteka izgleda je XML datoteka u kojoj definirate kontrole potrebne za zaslon.

**Kako je izgled pridružen aktivnosti?**

Android stvara objekt Aktivnosti i poziva svoju onCreate() metodu. Metoda onCreate() je konstruktor gdje se obavlja sva inicijalizacija. Izgled se dodjeljuje Aktivnosti u onCreate() pozivanjem metode pod nazivom setContentView(LayoutId).

**public class** MainActivity **extends** AppCompatActivity

Svaka klasa *MainActivity* je podklasa klase *AppCompatActivity*.

@Override  
**protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_main);}

Android poziva u *onCreate(Bundle savedInstanceState)* i kreiran je objekt Aktivnosti.

Pozivom metode setContentView(R.layout.activity\_main) povezuju se datoteke Layout i Activity.

Activity (.java) …………setContentView…………………… Layout(XML)

Aktivnost upravlja interakcijom s korisnikom i govori što treba učiniti kada se nešto dogodi (na primjer, pritisak na gumb). Svaka aktivnost nasljeđuje se od AppCompatActivity i onCreate() metoda se poziva nakon kreiranja objekta aktivnosti.

Izgled je XML resurs koji ima ID. Zapravo, svaki resurs u mapi resursa ima ID koji se koristi za pristup njima u Java kodu. Ova datoteka Layout definira izgled Aktivnosti i kako su kontrole postavljene. Kontrole su Views i ViewGroups.

# Views, View Groups i Layouts

Android aplikacije zahtijevaju sjajan dizajn. Za izvrstan dizajn nužno je savladati rasporede. Izgled je jedan od osnovnih građevnih blokova Android korisničkog sučelja. Izgled je jedan od osnovnih građevnih blokova Android korisničkog sučelja.

Što su ViewGroups i Views?

**VlewGroups** su spremnici za poglede kao što su TextView, Button, Checkboxes itd., a mogu sadržavati i druge grupe pogleda. Na primjer, možete imati linearni raspored (LinearLayout) unutar relativnog rasporeda (Relative Layout). Oni su potrebni za raspoređivanje prikaza na ekranu. Oni definiraju granicu unutar koje definirate svoje stavove. Svaka grupa pogleda ima svoj stil i svojstvo postavljanja pogleda. Ovo su upravitelji izgleda ili kontrole izgleda.

**Views** su građevni blokovi svake Android aplikacije. TextView, Button, Checkbox, Radio Button itd. - sve su to pogledi i oni nasljeđuju klasu View (import android.view.View). Svaki pogled zauzima pravokutno područje na zaslonu, a područje je definirano pomoću piksela neovisnih o gustoći (dp ili dip). Općenito se ti prikazi postavljaju unutar spremnika (ViewGroups) kako bi imali dosljedan izgled na različitim veličinama zaslona i razlučivostima zaslona. Postoje dvije posebne vrijednosti koje se koriste za definiranje širine i visine pogleda MATCH\_PARENT i WRAP\_CONTENT.

MATCH\_PARENT - zauzima sav raspoloživi prostor. WRAP\_CONTENT - zauzima prostor koji pokriva samo sadržaj. Na primjer, ako pogled ima android:layout\_width="match\_parent" && android:layout\_height="wrap\_content" tada će pokriti cijelu dostupnu širinu, a visina će pokriti prostor potreban za sadržaj.

**Što je dp i sp?**

**Dp** je jedinica veličine koja se koristi za različite atribute prikaza. Umjesto korištenja piksela, koriste se pikseli neovisni o gustoći. Android sam izračunava stvarnu veličinu piksela kako bi pružio dosljedan izgled aplikacijama. **Sp** je piksel neovisan o mjerilu. Ova jedinica veličine je ista kao i **dp**, ali se koristi samo za veličinu teksta ili veličinu fonta i poštuje korisničke preferencije veličine fonta.

**Što je Linear Layout?**

**Linear Layout** je grupa pogleda i linearno raspoređuje poglede koji mogu biti ili vodoravno ili okomito. Zadana orijentacija je vodoravna. android:orientation je atribut za definiranje smjera. Ako želite urediti prikaze u retku ili stupcu, koristite linearni raspored. Za redak je orijentacija okomita(vertical), a za stupac orijentacija je vodoravna (horizontal).

Jedno važno svojstvo koje se koristi u linearnom rasporedu - android:layout\_weight. Ovaj atribut govori linearnom rasporedu kako podijeliti prostor između podređenih pogleda. Prema službenoj dokumentaciji - težina definira važnost dječjih pogleda na to koliko mjesta zauzimaju ti dječji pogledi. Dječji prikaz s većom težinom (0, 1, 2) znači da će zauzeti više prostora u roditeljskom izgledu od ostalih podređenih prikaza.

Prvo se raspoređuju svi podređeni Views prema njihovoj širini izgleda ili svojstvu visine izgleda. Ako linearni raspored ima horizontalnu orijentaciju, uzima u obzir širinu izgleda, inače visinu izgleda. Zatim ponovno provjerava ima li svojstvo težine rasporeda ako ono postoji. Sada, ako postoji, dodatni prostor se dijeli proporcionalno prema težini između dječjih Views.

**Što je Relative Layout?**

Postoji još jedan izgled koji je iznimno koristan i fleksibilan u dizajniranju aplikacija dobrog izgleda, tj. Relativni raspored. Ovaj izgled raspoređuje poglede u odnosu na roditeljske ili druge podređene poglede. Možete reći nešto poput - poravnajte ovaj TextView lijevo od EditTexta ili poravnajte ovaj gumb ispod EditTexta.

Za pozicioniranje pogleda unutar relativnog izgleda, podređeni pogledi moraju specificirati atribute izgleda koji ih poravnavaju unutar relativnog izgleda. Ako ne navedete nikakve atribute izgleda za podređene poglede, svi podređeni pogledi bit će poravnati u gornjem lijevom kutu relativnog izgleda.

Atributi za definiranje položaja

Poglede možete poravnati u odnosu na roditeljski ili bilo koji drugi podređeni pogled. Atributi u nastavku koriste se za poravnavanje ruba pogleda s rubom roditelja. Na primjer: android:layout\_alignParentTop znači poravnati gornji rub podređenog pogleda s gornjim rubom nadređenog, a isto vrijedi i za druge atribute. Za to je prihvatljiva vrijednost **"true"** ili **"false"**. Ostali parametri:

* android:layout\_alignParentTop
* android:layout\_alignParentRight
* android:layout\_alignParentBottom
* android:layout\_alignParentLeft

Atributi koji dopuštaju da centriramo dječji pogled unutar roditelja. android:layout\_centerInParent znači centriraj vodoravno i okomito unutar nadređenog izgleda. Za atribut je prihvatljiva vrijednost **"true"** ili **"false"**.

* android:layout\_centerInParent
* android:layout\_centerHorizontal
* android:layout\_centerVertical

Atributi u nastavku koriste se za poravnavanje ruba pogleda s rubom djeteta. Za ove atribute prihvatljiva vrijednost je id pogleda na koji želite usidriti svoj pogled.

Na primjer: android:layout\_alignTop = "@id/btn\_send" znači da se gornji rub pogleda treba poravnati s gornjim rubom btn\_send. Jednostavnim riječima, gornji rub prikaza mora biti isti kao i gornji rub btn\_send.

* android:layout\_alignTop
* android:layout\_alignRight
* android:layout\_alignBottom
* android:layout\_alignLeft

Sljedeći atributi se koriste da podese izgled - ispod ovog pogleda, iznad ovog pogleda, desno od ovog pogleda i lijevo od pogleda. Za ove vrijednosti je potrebno navesti id pogleda na koji se želite usidriti.

Ako piše nešto poput - android:layout\_below = "@id/btn\_send" onda to znači da se gornji rub poravna s donjim rubom pogleda btn\_send. Jednostavno je tako što stavite svoj pogled ispod btn\_send. Isto vrijedi i za ostale atribute.

* android:layout\_above
* android:layout\_below
* android:layout\_toLeftOf
* android:layout\_toRightOf

# Primjer klika na gumb Androida i namjere (Intents) Androida

Aplikacije na početku zahtijevaju unose od korisnika da bi se poduzele radnje. U aplikaciji postoji gumb koji počinje obrađivati korisnički unos. Kada kliknete gumb, događaj klika (Event) se generira kao objekt. Za rukovanje ili poduzimanje radnje nakon klika na gumb, potreban vam je handler (upravitelj, rukovatelj) koji će izvršiti logiku aplikacije. Općenito, aplikacija će imati više od jedne aktivnosti. Jedna aktivnost se otvara iz druge aktivnosti na temelju neke logike ili korisničkog unosa. U našem slučaju, s glavnog zaslona na drugi zaslon kada korisnik pritisne tipku za reprodukciju. Ispod je isječak koda za rukovanje događajima klikova.

<**ImageButton  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/btnPlay"  
 android:src="@drawable/ic\_play"  
 android:padding="5dp"  
 android:layout\_margin="2dp"** />

Događajem klika upravljat će se gumbom kao što je prikazano u nastavku

**protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 btnPlay = (ImageButton) findViewById(R.id.btnPlay); *//FIND THE BUTTON* btnPlay.setOnClickListener(**new** View.OnClickListener() { *//SET ON CLICK LISTENER* @Override  
 **public void** onClick(View v) {  
 *// ON BUTTON CLICK do something* }  
 });  
}

Kao što se vidi iz isječka koda, prvo će se pronaći gumb koristeći jedinstveni ID koji mu je dodijeljen. Nakon što se kreira gumb, postavlja se slušatelja(Listener) da upravlja događajem onClick. Potrebno je upisati kod ili prilagođenu logiku aplikacije unutar metode onClick (View v) (Android Studio Auto-completion pomaže u pisanju ovog koda, čim počinje tipkanje automatski će ponuditi mogućnosti završavanja koda).

Slično kao u realnom životu: Slušam dok pričaju, ali kad kliknu na ovaj gumb, obavijestit ću ih da mogu raditi svoje poslove.

Sada kada je riješen događaj klika, piše se kod za otvaranje druge aktivnosti i iz glavne aktivnosti. Za ovo prvo moramo razumjeti što je namjera?

Postoje dvije vrste namjera (Intents):

1. Eksplicitna namjera.
2. Implicitna namjera.

Namjera je objekt koji se koristi za komunikaciju s Android operativnim sustavom. Namjera je poput poruke koja se prosljeđujete Android OS-u za obavljanje nekog posla u vaše ime. Slično kao da kažete Hej! Android, možeš li mi otvoriti ovu aktivnost? ili Hej! Android, želim podijeliti ovu poruku Možete li mi pomoći? Kada eksplicitno definirate svoju namjeru, onda je to eksplicitna namjera, inače je to implicitna namjera. Na primjer:

Eksplicitna namjera - Hej, Android! Otvori mi ovu aktivnost. Ovdje izričito rečeno da se otvori određenu aktivnost. Ovo je eksplicitna namjera. Definirano je što zapravo želite.

Intent i = **new** Intent(Context, ClassName); *//you specify the class name of the activity you want to open.*startActivity(i);

Implicitna namjera - Hej Android! Želim podijeliti ovu poruku. Možeš li mi pomoći? - U ovom slučaju želite da Android pronađe sve aplikacije koje mogu podijeliti poruku. Niste definirali nikakvu aktivnost za dijeljenje poruke. Android će sam pronaći i dati vam opcije za odabir aplikacije.

Zamislite scenarij - Napravili ste aplikaciju za pretraživanje nekih proizvoda. Sada želite dopustiti korisniku da pošalje e-poštu kako bi podijelio neke proizvode sa svojim prijateljima ili obitelji. Stvaranje aktivnosti e-pošte zajedno s vašom stvarnom aplikacijom bit će ogroman zadatak. Zašto ne koristite postojeću Gmail aplikaciju ili Outlook ili neku drugu aplikaciju za slanje e-pošte? Ovdje dolazi do izražaja implicitna namjera. Tražite od Androida sve aktivnosti koje mogu slati e-poštu. Android će vam pokazati moguće opcije za odabir.

Intent i = **new** Intent(Intent.ACTION\_SEND); *//you specify that you want to send a message*i.setType(**"text/plain"**); *// message type is plain text*i.putExtra(Intent.EXTRA\_TEXT, YOUR\_MESSAGE); *// you pass your message as extras in your intent object.*startActivity(intent);

Android prosljeđuje ovaj objekt, namjere aktivnosti koja se otvara putem namjere. Dodatne vrijednosti mogu se proslijediti korištenjem **intent.putExtra**. U metodi aktivnosti onCreate() možete pozvati **getIntent()** da biste dobili objekt namjere i izdvojili vrijednosti iz objekta namjere.

**protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
 String message = getIntent().getStringExtra(KEY);  
}

Sve navedeno znači, kreirati obrađivač za događaj klika na gumb (create a handler for button click event) i kreirati implicitnu i eksplicitnu namjeru s dodatcima koristeći putExtra().

**public class** MainActivity **extends** AppCompatActivity {  
  
 ImageButton **btnPlay**, **btnShare**, **btnRate**;  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
  
 **btnPlay** = (ImageButton) findViewById(R.id.btnMathPlay);  
 **btnShare**= (ImageButton) findViewById(R.id.btnMathShare);  
 **btnRate** = (ImageButton) findViewById(R.id.btnMathRate);  
  
 **btnPlay**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 *//  
 // Example of Explicit Intent  
 // When you click Play Button on the screen  
 // Activity will be started  
 //* Intent i = **new** Intent(MainActivity.**this**,SecondActivity.**class**);  
 startActivity(i);  
 }  
 });  
  
 **btnShare**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 *//  
 // Example of Implict Intent  
 // When you click Share Button on the screen  
 // Android will find the activities that allow to share messages.  
 //* Intent intent = **new** Intent(Intent.ACTION\_SEND);  
 intent.setType(**"text/plain"**);  
 intent.putExtra(Intent.EXTRA\_TEXT,**http://www.play.google.com"**);  
 startActivity(intent);  
 }  
 });  
  
 **btnRate**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 *//  
 // Simple Toast Message To Display Message For Short Duration  
 // Link that to your app landing page.  
 //* Toast.makeText(MainActivity.**this**,**"Open Google page"**,Toast.LENGTH\_LONG).show();  
 }  
 });  
 }  
}

# Android mjerač odbrojavanja - Android CountDown Timer

Mjerač vremena se koristi za praćenje vremena korisnika u kojem on mora odgovoriti na što više pitanja uz opciju Da ili Ne. Napraviti jednostavan mjerač vremena ili štopericu implementira se pomoću Handlera u Androidu. Koristit ćemo tzv. niti ili threads. Nit izvršenja je najmanji slijed programskih instrukcija. Izvršenje android aplikacije vrši se na glavnoj niti koja obrađuje interakciju s korisnikom.

Svi dugotrajni zadaci izvode se na drugim nitima. Recimo da želite implementirati mjerač vremena koji otkucava svake sekunde. Je li implementacija uz pomoć for petlje dobra ideja? Ne. Izvođenje takvog koda s petljom blokirat će vašu glavnu nit i zaustaviti druge operacije. Dakle, trebamo način da stvorimo novu nit koja otkucava sekunde i koja nam pomaže da implementiramo mjerač vremena štoperice u androidu. Druga važna stvar je da da samo glavna nit ažurira korisničko sučelje, što znači da druga nit koju se kreira za timer neće ažurirati korisničko sučelje. Potreban je izvesti ažuriranje i prikaz teksta putem glavne niti.

**Handler Class Android**

Ova klasa omogućuje izvršavanje koda u nekom trenutku u budućnosti i to u vlastitoj niti. Handler također pruža način interakcije s glavnom niti za sva ažuriranja korisničkog sučelja. Klasa Handler ima različite metode za planiranje izvršavanja koda. Koristit ćemo post(Runnable) i postDelayed(Runnable,long) za kreiranje jednostavnog mjerača vremena u androidu. post(Runnable) se pokreće odmah dok postDelayed(Runnable, long) radi nakon određenih milisekundi koje se prosljeđuju kao duga vrijednost. Isječak koda za jednostavan mjerač vremena u androidu

**private int** seconds = 10; *//THIS WILL RUN FOR 10 SECONDS***private boolean** stopTimer = **false**;  
  
**private void** timer() {  
 **final** Handler handler = **new** Handler();  
 handler.post(**new** Runnable() {  
  
 @Override  
 **public void** run() {  
 seconds--;  
 **if** (seconds < 0) {  
 *// DO SOMETHING WHEN TIMES UP* stopTimer = **true**;  
 }  
 **if**(stopTimer == **false**) {  
 handler.postDelayed(**this**, 1000);  
 }  
 }  
  
 });  
}

Handler object, je stvoren i metoda post(Runnable) se poziva da izvrši kod. Ovaj kod će se izvršiti odmah, a u samom njegovom izvršavanju druga metoda post(Runnable, 1000) poziv je napravljen s 1000 milisekundi da se napravi mjerač vremena. Jednostavnim riječima, prvo izvršite nešto i kažete izvršite isti kod ponovno nakon 1 sekunde = 1000 milisekundi. Kada su potrebne sekunde završene, zaustavljate mjerač vremena. To je poput ciklusa.

Aplikacija 1: Brojač klikova

<**androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:id="@+id/relativeLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity"  
 tools:layout\_editor\_absoluteX="0dp"  
 tools:layout\_editor\_absoluteY="81dp"**>  
  
 <**TextView  
 android:id="@+id/display\_count"  
 android:layout\_width="110dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="150dp"  
 android:layout\_marginTop="8dp"  
 android:layout\_marginRight="150dp"  
 android:layout\_marginBottom="8dp"  
 android:contextClickable="false"  
 android:gravity="center"  
 android:text="0"  
 android:textSize="100dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.501"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 app:layout\_constraintRight\_toRightOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0.392"** />  
  
 <**Button  
 android:id="@+id/button\_inc"  
 android:layout\_width="344dp"  
 android:layout\_height="48dp"  
 android:layout\_centerHorizontal="true"  
 android:layout\_centerVertical="true"  
 android:layout\_marginLeft="8dp"  
 android:layout\_marginTop="8dp"  
 android:layout\_marginBottom="8dp"  
 android:gravity="center"  
 android:onClick="increment"  
 android:text="+"  
 android:textSize="18sp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 app:layout\_constraintRight\_toRightOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0.683"** />  
  
 <**Button  
 android:id="@+id/button\_dec"  
 android:layout\_width="344dp"  
 android:layout\_height="48dp"  
 android:layout\_marginStart="8dp"  
 android:onClick="decrement"  
 android:text="-"  
 android:textSize="18sp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0.79"** />  
  
 <**Button  
 android:id="@+id/reset"  
 android:layout\_width="344dp"  
 android:layout\_height="48dp"  
 android:layout\_marginStart="8dp"  
 android:layout\_marginTop="8dp"  
 android:layout\_marginRight="8dp"  
 android:layout\_marginBottom="8dp"  
 android:onClick="reset"  
 android:text="Reset"  
 android:textSize="18sp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.0"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 app:layout\_constraintRight\_toRightOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0.568"** />  
  
</**androidx.constraintlayout.widget.ConstraintLayout**>

**package** com.example.myapplication5;  
**import** android.os.Bundle;  
**import** android.view.View;  
  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.TextView;  
  
**public class** MainActivity **extends** AppCompatActivity {  
 **int count** = 0;

**int number**=0;

@Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);

}  
 **private void** display(**int** number){  
 TextView brojacTextView = (TextView) findViewById(R.id.***display\_count***);  
 brojacTextView.setText(**""**+number);  
 }  
 **public void** increment(View view){  
 **count** = **count**+1;  
 display(**count**);  
 }  
 **public void** decrement(View view){  
 **count** = **count**-1;  
 display(**count**);  
 }  
 **public void** reset(View view){  
 **count** =0;  
 display(0);  
 }  
}

Aplikacija 2

*<?***xml version="1.0" encoding="utf-8"***?>*<**androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity"**>  
  
 <**LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"**>  
  
 <**Button  
 android:id="@+id/plavibutton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:onClick="toDo"  
 android:text="Plavi"** />  
  
 <**Button  
 android:id="@+id/button2"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:onClick="toDo"  
 android:text="Roza"  
 android:textColor=""**/>  
 </**LinearLayout**>  
</**androidx.constraintlayout.widget.ConstraintLayout**>

**import** androidx.appcompat.app.AppCompatActivity;  
  
  
  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.Toast;  
  
**import** java.util.Objects;  
  
**public class** MainActivity **extends** AppCompatActivity {  
 Button **btnPlavo**, **btnPinkPanter**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
  
 **btnPlavo** = (Button) findViewById(R.id.***plavibutton***);  
 **btnPinkPanter** = (Button) findViewById(R.id.***button2***);  
 }  
  
 **public void** toDo (View v){  
 **if** (Objects.*equals*(v, **btnPlavo**))  
 *// gumb Plavo je aktiviran  
 // gumb nestaje* {  
 } **else** {  
 v.setVisibility(View.***INVISIBLE***);  
 }  
 *// botun nestaje* **if** (Objects.*equals*(v, **btnPinkPanter**))  
 *// gumbPinkPanter je aktiviran  
 // prikaz poruke u kratkom trajanju* Toast.*makeText*(getApplicationContext(),  
 **"dismiss…"**   
 Toast.***LENGTH\_SHORT***).show();  
 }  
 }

https://www.srednja.hr/ideja-godine/