**SKRIPTA PMA**

1. **XML (eXtensible Markup Language)** -način za organizaciju i predstavljanje podataka, nezavisno od platforme I aplikacije. Bazirano na tagovima, slično kao HTML Za razliku od HTML-a tagove definiše korisnik, a nisu unaprijed definisani – biraju se da budu deskriptivni
2. **Komponente Android aplikacije**
   1. Aktivnosti (Activities)
   2. Servisi (Services)
   3. Broadcast receiver-i
   4. Content provider-i
3. **Aktivnosti** (Activities) - ulazna tačka za interakciju sa korisnikom. Predstavlja jedan ekran sa korisničkim interfejsom (čitav ekran ili pop-up). Omogućava interakciju između sistema i aplikacije. Implementira se kao potklasa klase Activity.
4. **„Glavna“ aktivnost** je ona koja se pokreće kad se startuje aplikacija (dodirom ikonice), ali se može direktno pozvati i neka druga aktivnost – sa nekog drugog mjesta (iz notifikacija ili druge aplikacije)
5. **Da bi koristili aktivnosti** u svojoj aplikaciji, moraju se registrovati podaci o njima u manifestu aplikacije i mora se pravilno upravljati njihovim životnim ciklusima
6. **Aktivnosti – životni ciklus** (U toku svog životnog ciklusa aktivnost prolazi kroz niz “stanja”, a svako od njih se servisira odgovarajućom callback funkcijom)**:**
7. onCreate()
8. onStart()
9. onResume()
10. onPause()
11. onStop()
12. onRestart()
13. onDestroy()
14. **onCreate() -** Obavezna funkcija! Pokreće se kad OS kreira aktivnost, ali prije nego što ona postane vidljiva korisniku.U njoj se inicijalizuju najvažnije komponente aktivnosti. Mora se pozvati funkcija setContentView() da definiše izgled korisničkog interfejsa aktivnosti
15. **onStart() -** Aktivnost postaje vidljiva korisniku**.** Finalne pripreme za stavljanje aktivnosti u prvi plan i omogućavanjeinteraktivnosti
16. **onResume() -** Izvršava se prije nego aktivnost započne interakciju sa korisnikom. U tom trenutku aktivnost prihvata unos od strane korisnika. Najveći dio funkcionalnosti aplikacije se implementira u ovoj funkciji.
17. **onPause()** - Izvršava se kada aktivnost više nije u fokusu, odnosno pauzira se (npr.

aktivirano dugme back). Aktivnost je još djelimično vidljiva, ali je korisnik (vjerovatno) napušta –nuskoro će ući u stanje Stop iliResume**.** Nakon onPause() slijedi ili onStop() (ako se napušta aktivnost) ili onResume() (ako se vraćamo u aktivnost).

1. **onStop()** - Izvršava se kad aktivnost više nije vidljiva korisniku. Nakon onStop() slijedi ili onRestart() (ako će aktivnost ponovo da komunicira sa korisnikom) ili onDestroy() (ako se aktivnost definitivno završava)
2. **onRestart()** - Izvršava se kad aktivnost u stanju Stop treba da se ponovo pokrene onRestart() obnavlja stanje aktivnosti kakvo je bilo kada je zaustavljena nakon onRestart() slijedi onStart().
3. **onDestroy()** - Izvršava se prije nego je aktivnost “uništena”. Uobičajeno se koristi da osigura da su svi resursi koje je aktivnost zauzela oslobođeni kada aktivnost, ili process kojemu je aktivnost pripadala, više ne postoji.
4. **Servisi** (Services) - Ulazna tačka opšte namjene za aplikaciju koja se pokreće u pozadini, za obavljanje dugotrajnih operacija ili za obavljanje poslova za udaljene procese. Na primjer, **servis** može da pušta muziku u pozadini dok je korisnik u nekoj drugoj aplikaciji ili može prenositi podatke preko mreže bez blokiranja interakcije korisnika sa nekom aktivnošću. **Ne pruža** korisnički interfejs.

Implementira se kao potklasa klase **Service**.

1. **Vrste servisa:**
2. Pokrenuti (started) servisi:

* Ako je korisnik svjestan njihovog izvršavanja (npr. muzika) onda sistem treba da ih drži aktivnim sve do njihovog završetka
* Ako korisnik ne prati izvršavanje procesa (npr. sinhronizacija podataka) onda sistem može da ih privremeno suspenduje da bi oslobodio resurse za prioritetnije poslove

1. Povezani (bound) servisi:

* pokreće ih neka aplikacija ili sistem i zavise od izvršavanja te aplikacije

1. **Broadcast receiver** - omogućava sistemu da pošalje obavještenje o nekom događaju, a aplikaciji da odgovori na takvo obavještenje. **Ne pruža** korisnički interfejs. Nije predviđen da radi veliki posao, već da bude veza između komponenti. Implementira se kao potklasa klase BroadcastReceiver, a obavještenja (broadcast) se isporučuju u obliku Intent objekata
2. Sistem može isporučiti obavještenje i aplikaciji koja nije trenutno aktivna (na primjer zadavanje alarma)
3. **Primjeri sistemskih obavještenja**: o isključenju ekrana, o statusu baterije
4. **Primjeri obavještenja koje generiše aplikacija**: završen prenos podataka, podaci ili resursi su dostupni za korišćenje
5. **Content provider -** Upravlja setom podataka koji se dijeli između aplikacija – omogućava da se podacima iz jednog procesa može pristupiti iz koda drugog procesa. Podaci se mogu nalaziti u fajlovima, bazi podataka, na web-u, ili bilo kojoj lokaciji za čuvanje podataka kojoj aplikacija može pristupiti. Content provider je, gledano iz ugla sistema, ulazna tačka u aplikaciju za pristup objektima (koji imaju ime definisano sa URI) u kojima se nalaze podaci. Pogodan je i za rad sa podacima koji se ne dijele, već su privatni za pojedinu aplikaciju
6. **URI** - Uniform Resource Identifier je sekvenca karaktera koja označava resurs - fizički ili apstraktni. Jedna od uobičajenih formi URI je adresa web stranice – Uniform Resource Locator (**URL**). Android koristi URI string kao osnovu za pristup podacima u content provider-u ili za pokretanje neke akcije (npr. otvaranje web stranice u browser-u)
7. **Android studio – važni fajlovi:**

* app > java > com.example.naziv\_projekta > MainActivity

U ovom fajlu se nalazi glavna aktivnost (Activity), koja predstavlja ulaznu tačku prilikom izvršavanja aplikacije. Dakle, prilikom pokretanja aplikacije biće pokrenuta instanca ovog Activity-a.

* app > res > layout > activity\_main.xml

U ovom XML fajlu se definiše izgled i raspored komponenti Activity-a. Prilikom kreiranja projekta Android studio sam ubacuje TextView element sa tekstom „Hello world!“.

* app > manifests > AndroidManifest.xml

U ovom fajlu se opisuju osnovne karakteristike aplikacije i definiše svaka njena komponenta.

1. Aktivnosti (Activities) – **konfiguracija manifesta**. Da bi deklarisali aktivnost, u fajl manifesta treba dodati <activity> element, kao podređeni elementu <application>:

<manifest ... ><application ... ><activity **android:name=".ExampleActivity"** />..</application ... >...</manifest>

• Jedini zahtijevani atribut za ovaj element je android:name, koji određuje naziv klase aktivnosti

1. **Rekapitulacija** –

**protected void** onCreate(Bundle savedInstanceState) {

* **super**.onCreate(savedInstanceState);/\* super.nazivMetode - pristup metodu direktne nadklasekoji je u podklasi redefinisan\*/
* setContentView(R.layout.activity\_main);/\* **setContentView** je metoda kojom se bira xml fajl koji definiše layout prvog ekrana
* **R** je klasa generisana tokom build procesa, koja referencira resurse.
* **R.layout.\*** referencira layout resurs koji se obično nalazi u **/res/layout** folderu \*/

Toast.makeText(getApplicationContext(),

**„TEKST ZA ISPIS",**

* Toast.**LENGTH\_SHORT**).show();/\* prikazuje pop-up poruku; Toast.LENGTH\_SHORT: 2s; Toast.LENGTH\_LONG: 3,5s \*/}

1. **Paleta u Android studiju -** Ovi widget-i se nazivaju views
2. Button – dugme
3. TextView – prikaz tekstualne poruke korisniku
4. RadioButton
5. RadioGroup – kontejner za RadioButton-e
6. ImageView – prikaz slike
7. WebView – prikaz web stranice
8. ProgressBar
9. ListView – lista iz koje se bira element
10. ScrollView – kontejner, kada ne može sve da stane na ekran
11. **Paleta – napuštene opcije** 
    1. EditText– za unos teksta preko softverske tastature – sad je podijeljeno na specijalizovane unose: PlainText, Password, ...
    2. NumberPicker – za izbor broja iz predefinisanog opsega
    3. DatePicker – izbor datuma
    4. TimePicker– izbor vremena
    5. TextClock – prikaz vremena/datuma u obliku stringa
    6. Chronometer – tajmer
12. **Organizacija widget-a na ekranu:**
13. **Vertical LinearLayout**: ekran je podijeljen u redove i svaki red može prihvatiti widget(-e) redosledom kojim su navedeni
14. **Horizontal LinearLayout**: ekran je podijeljen u kolone
15. **RelativeLayout**: relativno jedni u odnosu na druge
16. **Layout-i su u suštini takođe widget-i** pa se može vršiti ugnježdavanje
17. Ekran **mora** da ima jedan “korijeni” layout koji sadrži sve što se nalazi na ekranu.
18. Previše ugnježdenih layout-a **degradira performanse** – problem se ublažava koristeći RelativeLayout.
19. Svaki **view** ima svoje **ime** odnosno **ID**
20. **Widget može da bude “vezan” za**:
21. akcije (actions) – npr. promijeni boju nečemu, povećaj, smanji, preuzmi podatak, pozovi novu aktivnost, …
22. događaje (events) – npr. pritisnuto dugme, završen prenos, izvršen gesture, …
23. Svaka aplikacija **može pokrenuti komponentu** druge aplikacije. Svaka aplikacija se izvršava u zasebnom procesu sa dozvolama koje ograničavaju pristup drugim aplikacijama. Zato aplikacija ne može direktno aktivirati komponentu druge aplikacije. Da bi se aktivirala komponenta druge aplikacije, šalje se poruka OS-u

kojom se izražava namjera (**intent**) da se pokrene određena komponenta. OS aktivira traženu komponentu

1. Za razliku od drugih sistema, kod Androida nema jedne „ulazne tačke“ u aplikaciju (ne postoji **main()** funkcija)
2. **INTENT** - Asinhrona poruka kojom se aktivira aktivnost, servis ili broadcast receiver.
3. **Intent** povezuje pojedine komponente aplikacije tokom izvršavanja programa – iz iste ili druge aplikacije
4. **Intent** može biti:
5. **Eksplicitni** – aktivira tačno određenu komponentu (preko naziva klase)
6. **Implicitni** – aktivira određeni tip komponente ili podataka nad kojima se sprovodi neka radnja → sistem pronalazi komponentu na uređaju koja će obaviti posao i pokreće je (ako ih je više → korisnik bira)
7. **Kod aktivnosti** **i servisa intent** **definiše** radnju koju treba izvršiti i može navesti URI podataka sa kojima se radi (npr. otvoriti web stranicu)
8. **Kod broadcast receiver-a intent definiše** obavještenje koje se emituje (npr. poruku da je baterija pri kraju)
9. **AndroidManifest.xml** - aplikacija mora deklarisati sve svoje komponente u ovom fajlu
10. **Deklaracija komponenti:**
    1. **<activity>** za aktivnosti
    2. **<service>** za servise
    3. **<receiver>** za broadcast receiver-e
    4. **<provider>** za content provider-e
11. Broadcast receiver-i se mogu deklarisati u manifestu, ali se mogu i**dinamički** kreirati unutar programskog koda kao **BroadcastReceiver** objekti i registrovati pozivom metoda **registerReceiver()**
12. **Android Emulator** - softverski alat na računaru kojim se **simulira** android uređaj: izvršava aplikacije, pregleda web stranice, omogućava debagovanje. Može se konfigurisati više različitih emulatora sa odgovarajućim parametrima: veličina ekrana, rezolucija, orijentacija, verzija OS-a. Obavljaju se kompleksni i zahtjevni procesi pa je brzina rada emulatora značajno manja nego kod hardverskih uređaja. Pokretanje aplikacije takođe traje duže nego kod realnog hardvera. Na raspolaganju je više različitih konfiguracija nego što možemo imati hardverskih uređaja
13. Pokretanje emulatora: Tools → AVD Manager. Izabere se željeni uređaj sa liste i klikne na njegovo dugme Ukoliko nema (željenog) uređaja, kreira se novi: Create Virtual Device
14. **Korisnički interfejs kod Android aplikacije (layout) se gradi hijerarhijski preko:**
15. Layout-a (**ViewGroup** objekti – (nevidljivi) kontejneri koji kontrolišu

pozicioniranje objekata na ekranu)

1. Widget-a (**View objekti** – U/I komponente koje se vide na ekranu i sa kojima korisnik ima interakciju; npr. dugmad, labele, ...)
2. **U/I interfejs** se definiše pomoću XML fajlova.
3. Umjesto pisanja XML koda u tekstualnom editoru, može se koristiti **Layout Editor** koji je sastavni dio Android studija.
4. **ConstraintLayout** je layout koji definiše poziciju za svaki View na osnovu relacija u prikazu sa ostalim View objektima i roditeljskim layoutom – flat hijerarhija kojom se izbjegava ugnježdavanje layout-a kakvo je prikazano na prethodnom slajdu.
5. **Veličine se mogu specificirati u različitim mjernim jedinicama:**
6. **px** – piksel
7. **dp** ili **dip** – density-independent pixel (dp - za View)
8. **sp** – scale-independent pixel (sp - za fontove)
9. **in** – inči
10. **mm** - milimetri
11. **Definisanje sa dp omogućava** Android platformi da skalira GUI na osnovu rezolucije konkretnog uređaja. **dp - 1dp je jedan piksel na 160dpi** (dots per inch) ekranu – relativna jedinica. Npr: na ekranu od 240dpi, svaki dp piksel biće skaliran sa faktorom 240/160 (tj. 1,5): komponenta koja ima 100dp biće skalirana na 150 stvarnih piksela
12. Svi stringovi i numeričke vrijednosti **definišu se** u XML fajlovima koji se nalaze u podfolderima foldera res. Kasnije se oni referenciraju preko svojih naziva ispred kojih se stavlja simbol **@**
13. **Internacionalizacija** – omogućiti da se aplikacija prilagodi različitim jezicima i ostalim specifičnostima nekog područja
14. **Nazivi fajlova** u kojima se nalaze **resursi** (npr. slike) moraju biti sa malim slovima
15. **Kako se dugmetu pridružuje neka akcija?**

• Varijanta 1 (u primjeru „Pozivanje druge aktivnosti“). U xml fajlu se definiše „onClick“ atribut dugmeta povezan sa metodom koju treba pozvati kad se dugme pritisne (npr. „mojaMetoda“). U java kodu se napravi metoda public void mojaMetoda(**View v**) gdje v predstavlja widget koji je kliknut

• Varijanta 2(u primjeru „Konvertor milje kilometri“) U java kodu: Napraviti referencu prema dugmetu definisanom u xml fajlu **findViewById()** metodom. Zakačiti **OnClickListener** za dugme (metoda **setOnClickListener**). Definisati **onClick(View v)** metodu za klasu koja implementira. **OnClickListener** interfejs

1. **Kako se postavlja dugme na ekran?**

* Upisom u xml fajl, uz pomoć WYSIWYG alata (dizajn editor)
* Dinamički, kroz java kod: new Button(getApplicationContext()). Karakteristike se podešavaju metodama kao što su setText, setTextColor, setBackgroundColor. **Layout se definiše pomoću LayoutParams objekta.** Postavljanje na ekran: ekran.**addView**(dugme, parametri)

1. **Kako se pravi višejezična aplikacija?** Ne stavljati stringove direktno u java kod! Umjesto toga: Za svaki string unijeti par (**kljuc, sadrzaj\_stringa**) u fajl

**res>values>strings.xml.** Referencirati ovako definisane stringove:

• U xml-u: **@string/kljuc** **I**  U javi: **R.string.kljuc**

• Iskoristiti **Translation editor** kod fajla **strings.xml** da bi se: Dodao novi jezik I unijeli prevodi na novi jezik za svaki definisani string

1. **Kako se pravi lista sa različitim elementima?** Koristi se **ListView** kontejner widget. Pomoću **ArrayAdapter-a** se definiše: **- Sadržaj** liste (spisak elemenata iz polja stringova) i - **Layout**, odnosno izgled liste

**Da bi se reagovalo na izbor nekog elementa liste** treba dodati:

OnItemClickListener

1. **Kako rasporediti veći broj widget-a (views) na ekranu?** Koriste se **Layout-i** (**LinearLayout**, **RelativeLayout**, ...).

* **LinearLayout**: Orijentacija: **Horizontal**, **Vertical.**

Da bi **view** zauzeo određeni procenat layout-a: Zadati **weightSum** layout-a; Zadati da **view** ima visinu (ili širinu) od **0dp** i da atribut **layout\_weigth** ima vrijednost potrebnog dijela od **weightSum**

* **RelativeLayout**: Za svaki **view** definisati **ID.**  Pozicionirati svaki **view** pomoću **below**, **above**, **toRightOf**… uodnosu na **ID** za referentni **view**

1. **Kako postaviti sliku na pozadinu ekrana?** Kopirati fajl sa slikom (jpg, png, gif, ...) u folder **res/drawable** – ime fajla počinje slovom i sadrži slova, cifre i *underscore* karakter. Postaviti sliku iz fajla **slika.png** kao pozadinsku:

* U **XML-u**: android:background="@drawable/slika"
* U **Javi**: myView.setBackgroundResource(R.drawable.slika)

1. **Kako reprodukovati audio fajl?**  Napraviti **res/raw** folder u kome će se nalaziti audio fajl **ime\_fajla.mp3.** Korišćenjem **MediaPlayer** klase: MediaPlayer dzuboks = MediaPlayer.create(this, R.raw.ime\_fajla);

• dzuboks.**start()**

• dzuboks.**pause()**

• dzuboks.**setLooping(true)**

• dzuboks.**setLooping(false)**

1. **Kako snimiti podatke na Android uređaju?** Koristiti **Preferences**, sačuvane kao parovi **<ključ, vrijednost>** u XML fajlu SharedPreferences konfiguracija = getSharedPreferences(„IME\_FAJLA", Context.MODE\_PRIVATE);
2. Da bi se pristupilo fajlu potreban je **editor**: SharedPreferences.Editor editor = konfiguracija.edit();
3. Upis se vrši sa **putInt(), putDouble(), putString(), ... editor.putInt(„naziv\_kljuca", -16776961);**
4. Upis će se zaista obaviti kada se pozove **commit()** metoda **editor.commit();**
5. Čitanje se vrši sa **getInt(), getDouble(), getString(),** konfiguracija.getInt("naziv\_kljuca", 0);
6. **Kako prikazati sliku na ekranu?** Postaviti **ImageView** na odgovarajuće mjesto. Smjestiti fajl sa slikom (JPG, PNG ili GIF) u **res/drawable** folder (Ime fajla mora početi slovom i sadržati samo mala slova, cifre I donju crtu). **Iz XML fajla**:

<ImageView […] app:srcCompat="@drawable/imefajla"/>ili

<ImageView […] android:src=="@drawable/imefajla"/>

(**app:srcCompat** omogućava vektorske formate slike)

• **Iz java koda:** ImageView mojImageView= (ImageView)findViewById(R.id. mojImage);

mojImageView.setImageResource(R.drawable.imefajla);

1. **SeekBar**

SeekBar mojSeekBar=(SeekBar)findViewById(R.id.mojSeekBar);

• int max=mojSeekBar.getMax(); // **vraća maksimalnu vrijednost**

• int progres= mojSeekBar.getProgress(); // **vraća trenutnu vrijednost**

• mojSeekBar.setMax(150); // **programsko podešavanje max vrijedn.**

• android:max="150"// **podešavanje max vrijedn. u xml-u**

• mojSeekBar.setProgress(50); // **50 je default vrijednost - java**

• android:progress="50" // **50 je default vrijednost – xml**

• android:indeterminate="true" // **ciklična animacija bez stvarne** indikacije o progresu

• android:background="#0F0" // **zelena pozadina - xml**

• mojSeekBar.setBackgroundColor(Color.GREEN); // **zelena poz.-java**

• android:paddingTop="40dp" // **paddingStart, paddingEnd, paddingBottom**

• android:thumb="@drawable/ime\_slike" // **izgled dugmeta na klizaču**

1. **Kako sakriti/prikazati widget (view) na ekranu?** Koristi se metoda **setVisibility** sa argumentom: View.INVISIBLE – ako želimo da sakrijemo **view.** View.VISIBLE– ako želimo da prikažemo **view.**

* Primjer: ako želimo da se widget **natpis** vidi na ekranu: **natpis.setVisibility(View.VISIBLE);**
* Primjer: ako želimo da se widget **natpis** ne vidi na ekranu:

**natpis.setVisibility(View.INVISIBLE);**

1. **Upisivanje u Log.** Metode: **Log.v(), Log.d(), Log.i(), Log.w(), Log.e()**

* Dva argumenta: metode je **TAG**, a drugi je **poruka**

1. **TAG** je string koji na neki način označava ispis – npr. naziv klase ili komponente ili neka druga oznaka. Ispis se pojavljuje u **logcat** prozoru Android studija. Format ispisa: **datum vrijeme PID-TID/package prioritet/TAG: poruka**
2. Prioritet od najvišeg ka najnižem:

* **Log.e()** (error)
* **Log.w()** (warning)
* **Log.i()** (information)
* **Log.d()** (debug – kompajlira se, ali se izbacuje tokom izvršavanja)
* **Log.v()** (verbose – ne kompajlirati u finalnu aplikaciju, samo
* prilikom razvoja)

1. Praksa je da se **TAG** deklariše kao konstanta. Primjer:

**private static final String TAG = "VelikoDugme"; … Log.i(TAG, "Dugme je pritisnuto");**

1. U **logcat-u** se može kontrolisati koje poruke se prikazuju (i dalje se

sakupljaju sve poruke). Moguće je i filtrirati poruke koje se prikazuju

1. **GridLayout -** mreža ćelija slično tabeli

• Broj redova i kolona se može specificirati pomoću atributa: **android:rowCount** i **android:columnCount**

• Širina kolone je definisana širinom najšireg widget-a u toj koloni, a visina vrste je definisana visinom najvišeg widget-a u toj vrsti.

• Po default-u nema margina između ćelija. Ako treba da ih bude: **android:useDefaultMargins="true"**

1. **GridLayout – widget-i.** Ako widget treba da se prostire preko npr. dvije kolone, njegov atribut je: **android:layout\_columnSpan="2"**

• Eksplicitno pozicioniranje widget-a pomoću atributa: **android:layout\_column** i **android:layout\_row**

• Poravnavanje widget-a unutar ćelije pomoću atributa: **android:layout\_gravity** (right, left, center, top, bottom, …)

• Da widget zauzme čitavu ćeliju: **android:layout\_gravity="fill\_horizontal“** (po horizontali)

(fill, fill\_vertical)

1. **Kako prikazati web stranicu?** Postaviti **WebView** na odgovarajuće mjesto.

• Ako je web stranica resurs aplikacije: Kreirati folder **assets**. Iskopirati fajlove u ovaj folder. Navesti URL u obliku: “**file:///android\_asset/ime\_fajla.html**”

• Ako je *web* stranica na Internetu: Navesti URL u obliku: ”**http://www.server.domen**”

• U **AndroidManifest** fajlu zahtijevati dozvolu za pristup Internetu:

**<uses-permission android:name="android.permission.INTERNET"/>**

**android:usesCleartextTraffic="true“ (sigurnosni razlozi)**

• Učitavanje web stranice: **webView.loadUrl(URL);**

• Da se ne otvara u novoj aktivnosti: **webView.setWebViewClient(new WebViewClient());**

1. **Kako napraviti aplikaciju sa više aktivnosti?**

• Kreirati nove aktivnosti: **desni klik na app>New>Activity...** U izvornoj aktivnosti kreirati **Intent** koji će pokrenuti druguaktivnost:

**Intent** pokreniDruguAktivnost = **new Intent();**

pokreniDruguAktivnost.**setClass**(this, DrugaAktivnost.**clas**s); startActivity(pokreniDruguAktivnost);

• Da uklonimo aktivnost sa **activity stack-a**: finish();

1. **Kako proslijediti podatke između aktivnosti?**

• U izvornoj aktivnosti: **putExtra**(ključ, vrijednost);

• U prijemnoj aktivnosti: **Intent** prijem = **getIntent**(); float podatak = prijem.**getFloatExtra**(„ključ“, 0);

1. **Kako iskoristiti aplikaciju za slanje SMS?**

• Kreirati implicitni **Intent** za obavljanje **ACTION\_SENDTO**

**Intent smsIntent = new Intent(Intent.ACTION\_SENDTO,**

**Uri.parse(„smsto:5556“));**

• OS pronalazi aktivnost koja se deklarisala da obavlja ovaj zadatak

• Ako ih je više, korisnik bira jednu od njih

**• smsIntent.putExtra("sms\_body",“poruka za tebe!“);**

**• startActivity(smsIntent);**

**Sa lab-a**

1. Program se može pokrenuti i testirati koristeći emulator ili na fizičkom (stvarnom) uređaju.
2. Ako želimo da se prilikom rotacije uređaja **ne mijenja** orijentacija ekrana, treba promijeniti fajl **- AndroidManifest.xml**:

android:screenOrientation="portrait">

1. **Super** je ključna riječ u Javi. Koristi se:

* Unutar definicije metode izvedene klase, da bi se pozvala metoda definisana u baznoj (roditeljskoj) klasi (super klasi). Ne mogu se pozivati privatne (private) metode bazne klase, već samo javne (public) i zaštićene (protected) metode.
* U okviru konstruktora izvedene klase da pozove konstruktor bazne klase. Uvijek je potrebno pozivati odgovarajući konstruktor bazne klase u konstruktorima izvedene klase. Poziv konstruktora bazne klase mora biti prva naredba u tijelu konstruktora izvedene klase. Ukoliko to nije slučaj, kompajler će sam ubaciti poziv podrazumijevanog konstrukotra bazne klase (bez parametara, što uzrokuje grešku).
* **Ne koristi** se kod static metoda.
* Napomena: kad se kreira instanca izvedene klase, implicitno se kreira instanca bazne klase, i na nju upućuje referentna promjenljiva super. **Sintaksa:**

super.([zero or more arguments]); ili super([zero or more arguments]);

1. Layout editor ima dva tab-a: **Design** i **Text**. U Design prozoru se korisnički interfejs kreira grafički, a u Text prozoru se opisuje tekstualno koristeći XML.
2. U **Deisgn prozoru** se nalaze dvije podloge, jedna bijele i jedna tamnije boje. Na bijeloj podlozi se prikazuje korisnički interfejs onako kako će izgledati na uređaju, a tamnija podloga je tzv. blueprint
3. **app > res > values > strings.xml.** U ovom fajlu se navode svi stringovi koji se koriste u projektu kako bi se nalazili na jednom mjestu i time se lakše uređivali (mijenjali, prevodili na više jezika
4. **Translations Editor -** pomoću kojega se unose i mijenjaju stringovi, a služi i za organizaciju prevoda tih stringova na više jezika.
5. **Match constraints** znači da će se objekat (u ovom slučaju tekst kontrola) raširiti da popuni prostor koji preostaje kada se odvoji prostor za ostale objekte (u ovom slučaju dugme) i predviđene margine.
6. **Ukoliko se pojavi greška** (Android studio ne može razriješiti klasu View koja se koristi kao argument metode – ključna riječ View je označena crvenom bojom) treba postaviti kursor unutar riječi View i pritisnuti **Alt + Enter**
7. **Da bi se neka metoda mogla koristiti** za poziv preko atributa **onClick**, ta metoda mora da bude public void i da ima jedan parametar i to View tipa.
8. **Intent -** omogućava povezivanje između odvojenih komponentinza vrijeme izvršavanja. Radi se o asinhronim porukama koje omogućavaju komponentama aplikacija da traže funkcionalnost od drugih Android komponenti. Omogućavaju komunikaciju sa komponentama iz istih aplikacija, kao i sa komponentama koje sadrže druge aplikacije. Može se koristiti za razne namjene, ali će se ovdje iskoristiti za pokretanje nove aktivnosti.
9. **Intent konstruktor** ima dva parametra: tipa Context i tipa Class.
10. **Context** je interfejs sa globalnim informacijama o okruženju aplikacije. Ovo je apstraktna klasa koja omogućava pristup resursima aplikacije i klasama, kao i operacije na nivou aplikacije (pokretanje aktivnosti, emitovanje i prihvatanje Intenta, ...).
11. **Klasa tipa Class** reprezentuje klase i interfejse u Java aplikaciji
12. **Metoda putExtra()** dodaje sadržaj stringa poruka u Intent. Za identifikaciju se koristi tzv. ključ, koji se navodi kao prvi parametar metode (u gornjem primjeru sadržaj stringa "EXTRA\_PORUKA"). Drugi parametar je vrijednost koja se pridružuje ključu.
13. **Prilikom dodavanja dugmeta za navigaciju** treba navesti koja aktivnost je roditeljska, u fajlu AndroidManifest.xml. Otvoriti fajl app > manifests > AndroidManifest.xml, i zamijeniti sadržaj taga za aktivnost PrikaziPoruku sa sljedećim:

<activity android:name=".PrikaziPoruku" android:parentActivityName=".MainActivity"><!-- meta-data tag je potreban za API nivoa 15 i nize --><meta-data android:name="android.support.PARENT\_ACTIVITY" android:value=".MainActivity" /></activity>

1. **Ukoliko želite da tekst bude ispisan baš onako kako je unijet**, u xml fajl treba ručno unijeti: android:textAllCaps="false"
2. **Da bi se dobila ta referenca,** koristi se funkcija findViewById()
3. **Pravljenje instancu nekog objekta**:

dugmeKonvKmUMilje = new Button(getApplicationContext());

1. **Boje se specificiraju** kao RGB (red-green-blue) ili ARGB (alpha-red-green-blue) vrijednosti.
2. **Tekst na dugmetu se zadaje na sledeći način:** dugmeKonvKmUMilje.setText("KOnvertuj km u milje");
3. **Da bi se definisala pozicija na ekranu**, mora se prvo zadati ID objektu u okviru kojeg se pozicionira, što je u ovom slučaju ConstraintLayout
4. **Da bi** **dugme bilo centrirano** u okviru layout-a:

parametri.startToStart

parametri.endToEnd

1. Ako želimo da ekran bude **podijeljen u redove**, potrebno je specificirati orijentaciju ovog LinearLayout-a: android:orientation="vertical"
2. Da element zauzme onoliko prostora koliko mu je potrebno: **„wrap\_content“.**
3. **Audio fajl**: Postoje dva dugmeta: „Reprodukcija“ i „Pauza“, i „prekidač“ preko koga se bira da li će se fajl reprodukovati samo jednom ili će se reprodukcija neprekidno iznova ponavljati.

**app>res>raw**

1. **Switch –** koristi se za ponavljanja
2. **Listener** koji će pratiti promjenu stanja prekidača i omogućiti da se prilikom promjene obavi neka akcija (**setOnCheckedChangeListener**)
3. **Preferences** gdje se informacije čuvaju u parovima „ključv rijednost“ u xml fajlu**.** Objekat koji omogućava manipulaciju sa ovim podacima je tipa **SharedPreferences.**
4. **kod LinearLayout-a** se veličina njegovih widget-a može zadavati proporcionalno preko **layout:weight** atributa, koji specificira relativnu veličinu widget-a u odnosu na druge widget-e istog layout-a. **Podrazumijevana vrijednost** layout:weight atributa za svaki dodati widget je 0, što znači da u početku nije predviđen za proporcionalno zadavanje veličine
5. **SeekBar** omogućava da se izabere neka cjelobrojna numerička vrijednost (podrazumijevano između 0 i 100) pomjerajući klizač u jednu ili drugu stranu
6. **Tri vrijednosti klizača:** Metoda onStartTrackingTouch() se poziva kada korisnik dodirne klizač. Metoda onStopTrackingTouch() se poziva kada korisnik prestane da dodiruje klizač. onProgressChanged() koja se poziva kad god se klizač pomjeri
7. **Podešavanje vidljivosti** se vrši metodom setVisibility(). Kada želimo da objekat učinimo nevidljivim ovoj metodi se prosleđuje View.INVISIBLE. U suprotnom se prosleđuje View.VISIBLE.
8. **Toast pop-up** poruka se dobije Informacija o tome koja callback funkcija se trenutno izvršava. Međutim, te poruke se kratko nalaze na ekranu i lako se mogu previdjeti. Mnogo pogodniji način da se saopšti i sačuva informacija o toku izvršavanja programa (kao i mnoge druge informacije) je da se izvrši upis u tzv. **log fajl.**
9. Mora se voditi računa i o tome da se **sjenke neće prikazivati** ako u Android manifestu postoji ova linija: **android:hardwareAccelerated="false"**
10. **Kad aplikacija nema pravo pristupa Internetu.** Ovo pravo (dozvola) se specificira u **AndroidManifest-u.** Odmah ispod definicije aplikacije se navodi:

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

* A unutar application taga: android:usesCleartextTraffic="true"

1. **Definisanje web klijenta:** webView.setWebViewClient(new WebViewClient());

**Probni Kolokvijum**

1. **AndroidManifest.xml je fajl koji sadrži**: karakteristike aplikacije
2. **Ulazna tačka za interakciju Android aplikacije sa korisnikom putem korisničkog interfejsa je:** Aktivnost (Activity)
3. **Android aplikacija može da se napiše jedino koristeći Java programski jezik:** Netačno
4. **Prije pokretanja Android aplikacije na virtuelnom uređaju (emulatoru), morate povezati pametni telefon ili tablet na vaš PC:** Netačno
5. **Označiti tačne iskaze vezane za Android emulator:** 1.Pokretanje aplikacije na emulatoru traje duže nego kod realnog hardvera; 2. Može se konfigurisati više različitih emulatora sa odgovarajućim parametrima - npr. različitim veličinama ekrana.
6. **Komponente Android aplikacije su**: Aktivnosti (Activities) ;Servisi (Services)
7. Koji layout omogućava da specificirate polažaj widget-a koristeći druge widget-e kao referentne tačke? RelativeLayout
8. Dat je sljedeći kod:

public void posaljiPoruku(View v) {

String poruka = ((EditText)findViewById(R.id.editText\_poruka)).getText().toString();

Uri odrediste = Uri.parse("smsto:5556");

Intent smsIntent = new Intent(Intent.ACTION\_SENDTO,odrediste);

smsIntent.putExtra("sms\_body",poruka);

startActivity(smsIntent);

}

U gornjem kodu se nalazi **implicitni Intent.**

1. **Koja metoda Toast klase omogućava prikazivanje pripremljene pop-up poruke?**

**Izaberite jedan odgovor:** show()

1. **Da bi napravili aplikaciju koja ima korisnički interfejs na više jezika najbolji je način da** : umjesto stringa koji želite prikazati navedete ključ za taj string i definišete više fajlova sa resursima (jedan po jeziku) gde je za svaki ključ dat odgovarajući, prevedeni string.
2. **U koji folder treba da stavite multimedijalne fajlove koje su dio vaše aplikacije?**

**Izaberite jedan odgovor:** app/res/raw

1. **Šta je URI?** Uniform resource identifier - sekvenca karaktera koja označava resurs – fizički ili apstraktni.
2. Koju komponentu sa palete dizajn editora treba izabrati da se uključi u xml layout fajl kako bi se u korisnički interfejs aplikacije uključilo polje preko koga korisnik može da umetne sliku na odgovarajuće mjesto? Odgovor: **ImageView**
3. Definisali ste **SeekBar** na ekranu vaše aktivnosti na sljedeći način:

SeekBar mojSeekBar=(SeekBar)findViewById(R.id.mojSeekBar);

Da bi gornji SeekBar bio zapravo ciklična animacija, bez stvarne indikacije o progresu, potrebno je u XML fajlu zadati njegov atribut na sljedeći način:

Izaberite jedan odgovor: android:indeterminate="true"

1. Koja metoda Activity klase omogućava da preuzmete **referencu** na widget definisan u XML layout fajlu (pod pretpostavkom da znate "ime" widget-a)? findViewById()

**Dodatna pitanja**

1. **Koji je naziv metode MainActivity klase koja podešava i prikazuje ekran sa korisničkim interfejsom aplikacije?** onCreate()
2. **Koja metoda Toast klase omogućava prikazivanje pripremljene pop-up poruke?** show()
3. Definisali ste **SeekBar** na ekranu vaše aktivnosti na sljedeći način:

Seekbar mojSeekBar=(SeekBar)findViewById(R.id.mojSeekBar);

Želite da se na klizaču, umjesto default-nog dugmeta, nađe symbol iz fajla moja\_slika.png koji je smješten u odgovarajućem resursnom folderu što u XML faju odgovarajuće aktivnosti definišete: android:thumb=”@drawable/moja\_slika”

1. Šta morate uraditi da bi vaš **WebView mogao prikazati web stranicu** koja se nalazi na Internetu? Deklarišete da aktivnost u kojoj se nalazi WebView treba dozvolu za pristup Internetu.
2. **Koji layout omogućava da specificirate položaj widget-a koristeći druge widget-e kao referentne tačke?** RelativeLayout
3. Šta bi mogao **biti razlog da se neki widget (na primjer dugme) kreira iz java koda**, a ne u XML layout fajla? Na taj način se koristi manje linija koda
4. **Ako kreirate resurs u XML fajlu (na primjer string), kako možete pristupiti tom resursu iz java koda?** Ako se string zove “moj\_string”, referencira se kao R.string.moj\_string
5. Ako želimo da se prilikom rotacije Android uređaja ili emulatora **ne mijenja orijentacija ekrana** aktivnosti activity\_main android:screenOrientation=”portrait”

u fajl … AndroidManifest.xml

1. U res/raw folderu se nalazi audio fajl moja\_muzika.mp3.Kreiran je objekat MediaPlayer klase: MediaPlayer muzika = MediaPlayer.create(this,R.raw.moja\_muzika);

**Započinjanje reprodukcije** se vrši pozivom metode … dzuboks.start()

1. **Android emulatoru (koji je dio Android studija) se može pristupiti spolja, preko telnet sesije.** Tačno
2. Definisali ste korisnički interfejs u XML layout fajlu koji ste nazvali myUl.xml. Kreirate novu aktivnost (activity) u fajlu MyActivity.java. Kako specificirati da MyActivity prikaže myUl? U onCreate metodi u fajlu MyActivity.java poziva se setContentView metoda i prosljeđuje joj se parametar “R.layout.myUl”
3. **Komponenta koja omogućava sistemu da pošalje obavještenje o nekom događaju**, a aplikaciji da odgovori na takvo obavještenje je .. Broadcast receiver
4. **Dugme se može postaviti na ekran**: Upisom u xml fajl ili dinamički, kroz java kod
5. Sadržaj konfiguracionog fajla koji se zove starost.xml izgleda ovako ... value=“44“
6. Na ekranu aktivnosti nalaze se tri radio dugmeta preko kojih korisnik vrši izbor načina plaćanja neke usluge. Ova radio dugmad su smještena ID: nacinPlacanja\_RadioGroup. Ispod dugmadi se nalazi TextView čiji ID je izabranoPlacanje\_TextView I preko koga se ispisuje izabrani string “Način plaćanja:” I teksta koji se nalazi pored izabranog radio dugmeta….. radioGrupa
7. **Kad se poziva metoda onDestroy() klase Service**? Kada zaustavimo servis koristeći metodu stopService()
8. Ukoliko želimo da na radnu površinu ekrana postavimo dugme koje ima samo sliku, bez tekst, iskoristićemo widget:ImageButton
9. **Svi servisi** koje kreiramo **moraju biti deklarisani** u fajlu: AndroidManifest.xml

Fajl colors.xml služi da se u njemu definišu boje koje želimo da koristimo u našem projektu, kako bi se umjesto brojnih vrijednosti u RGB format

Fajl koji se nalazi u folderu: res>values

1. Izgled ekrana sa korisničkim interfejsom pravljenim za ekran male širine (portrait mod) često nije pogodan za prikaz na ekranu velike širine (landscape)

dva korisnička interfejsa, po jedan za svaki mod. Izgled ekrana neke aktivnosti u portrait modu I izgled ekrana za istu aktivnost u landscape modu se fajla, koji se nalaze u različitim folderima I imaju identičan naziv.. Tačno

1. Android studio omogućava programiranje aplikacija korišćenjem sljedećih programskih jezika: Kotlin, Java, C++
2. Želimo da prikažemo neke podatke u više redova (u obliku liste) I u tu svrhu koristimo widget ListView. Da bi izabrali čime će se popuniti razmak

Potrebno je podesiti atribut… divider

1. Sljedeći XML kod za definisanje grafičkog korisničkog interfejsa je neispravan: Tačno
2. Izgled ekrana sa korisničkim interfejsom pravljenim za ekran male širine (portrait mod) često nije pogodan za prikaz na ekranu velike širine (landscape mod)

po jedan za svaki mod. Ako se izgled ekrana neke aktivnosti za portrait mod definiše u fajlu moja\_aktivnost.xml onda izgled ekrana za neku aktivnost: moja\_aktivnost.xml

1. Dugmetu pozicioniranom na ekranu pomoću xml layout fajla se …

Može pridružiti neka akcija ili u xml fajlu definisanjem onClick atributa povezanog sa metodom koju treba pozvati kada se dugme pritisne, ili u java kodu tako što će onClickListener

1. Tokom izvršavanja servisa potrebno je prenijeti nekoliko fajlova sa internet. U tu svrhu se metodi doinBackground() prosleđuje niz URL objekata.Metoda izgleda ovako: Umjesto \*\*\*\*\*\*\*\*\*\*\*\* treba da stoji: (url[i])
2. Sadržaj konfiguracionog fajla koji se zove starost.xml izgleda ovako ... value=“44“
3. Ako kreirate string u XML fajlu (strings.xml), kako možete referencirati taj string iz drugog XML fajla (na primjer iz activity\_main.xml)? Ako se string zove „moj\_string“, referencira se kao @string/moj\_string
4. Podrazumijevani tekst na **ToggleButton** dugmetu u isključenom stanju je „OFF“. Da bi umjesto ovoga programski upisali neki drugi tekst, koristi se metoda... setTextOff()
5. Izgled ekrana sa korisničkim interfejsom pravljenim za ekran male širine često nije pogodan za prikaz na ekranu velike širine (landscape mod). Zato se često prave dva korisnička interfejsa, po jedan za svaki mod. Izgled ekrana neke aktivnosti u portrait modu i izgled ekrana za istu aktivnost u landscape modu se definišu u dva odvojena fajla, koji se nalaze u različitim folderima i imaju identičan naziv.

**Tačno**

1. Želimo da korisniku aplikacije omogućimo izbor između više ponuđenih opcija. Ponude su date u vidu RadioButton-a, po jedan RadioButton za svaku ponuđenu opciju. Ako želimo da korisnik može izabrati samo jednu od ponuđenih opcija, onda sve te RadioButton-e moramo staviti u RadioGroup kontejner.

**Tačno**

1. Kada jedna aktivnost želi da pokrene drugu aktivnost, ona kreira objekat koji specificira koju aktivnost ili koji tip aktivnost želi da pokrene. Koji je tip ovog objekta?

**Intent**

1. Ukoliko želimo da dugme (widget ImageButton) ima drugačiju sliku kada je pritisnuto, potrebno je kreirati XML fajl u kome se nalaze informacije o fajlovima sa slikama i povezati ga sa atributom srcCompat.

**Tačno**

1. Tokom izvršavanja servisa potrebno je prenijeti nekoliko fajlova sa interneta. U tu svrhu se metodi doinBackground() prosleđuje niz URL objekata. Metoda izgleda ovako.. Umjesto \*\*\*\*\* treba da stoji: (urls[1])