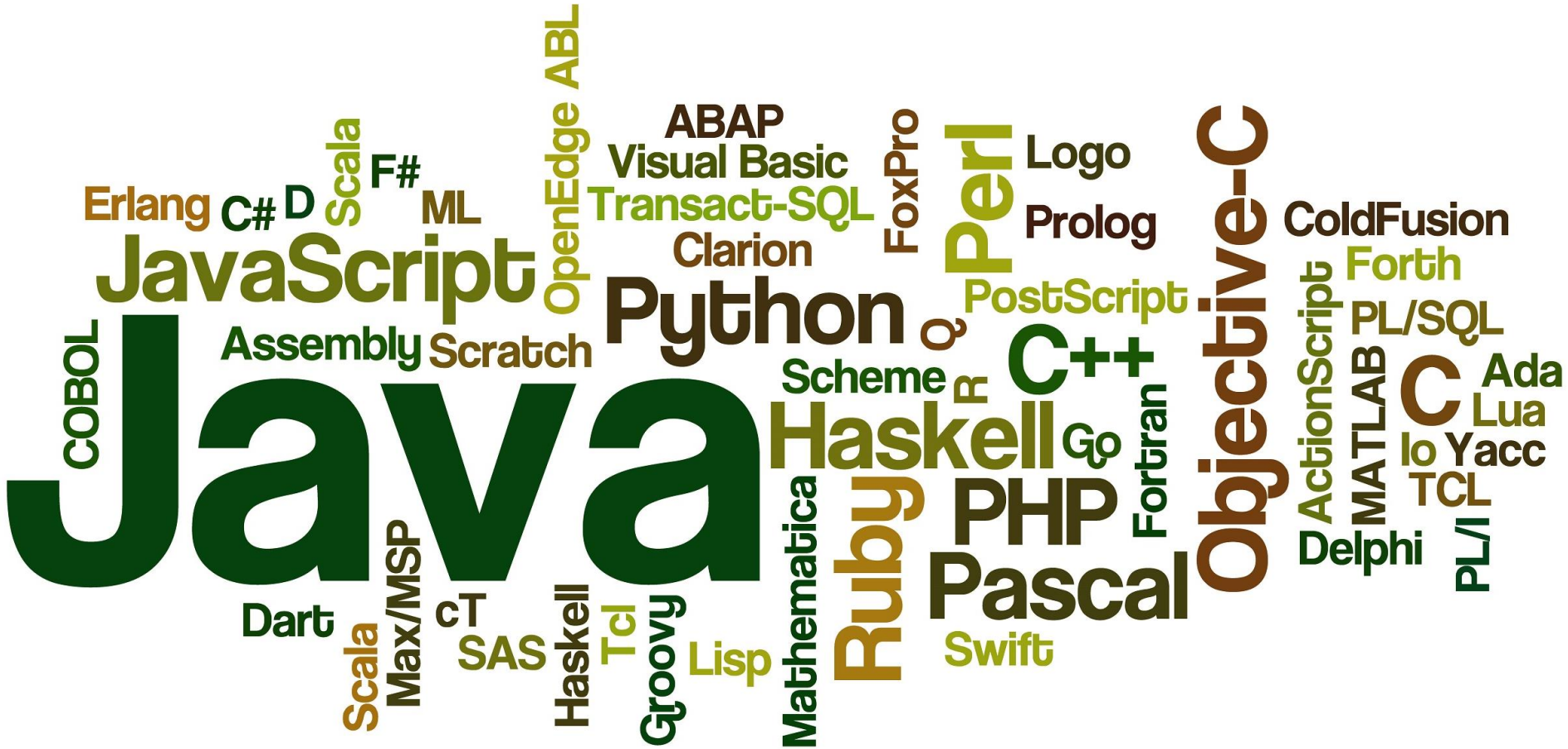
8 čas

Programski jezici i Android OS


Java
JavaScript
Python
C++
Haskell
Ruby
PHP
Pascal
Objective-C
C
Perl
Assembly
Scratch
Scheme
Fortran
Swift
Lisp
Groovy
Tcl
Haskell
Mathematica
Dart
Max/MSP
SAS
cT
Scala
COBOL
Erlang
C#
D
Scala
F#
ML
OpenEdge ABL
ABAP
Visual Basic
Transact-SQL
Clarion
FoxPro
Logo
Prolog
ColdFusion
Forth
PL/SQL
Ada
Lua
Yacc
TCL
PL/I
Delphi
MATLAB
ActionScript
PostScript
Q
R
Go
lo

Mašinski jezik, asembler, viši programski jezici

Softver - pojam

- Softver (*software*) je skup nematerijalnih komponenti računarskog sistema
- Softver „oživljava“ hardver
- *Softver čini celokupnu logičku ili programsku podršku jednog računarskog sistema - programski jezici, operativni sistemi, aplikativni programi, softver za dijagnostiku i zaštitu, poslovni softver itd...*
- Bez softvera, korišćenje hardvera nema smisla i obrnuto
- Hardver i softver moraju biti kompatibilni (usklađeni) da bi računarski sistem funkcionisao ispravno

„Softver je ono što vaš računar čini da se ponaša i izgleda pametniji od vas“ (Winn Sehworten, 1994)

Program i programiranje

- Računarski program, softverski program ili program (*computer program, software program, program*) predstavlja spisak naredbi (programski kôd) napisan u programskom jeziku namenjen za određenu računarsku platformu

- Programi se pišu u programskom jeziku i razumljivi su čoveku, ali ne i računaru pa se programski kôd pomoću kompajlera mora prevestii u binarni kôd da bi ga računar razumeo i izvršavao

- Čovek pomoću programa „uči“ računar kako da reši zadatke koje mu postavlja

- Programiranje (*programming development*) je poseban pristup rešavanju zadataka u računarskom sistemu i nije samo prosta upotreba sintakse programskog jezika ili "pisanja" programa ili prosto razumevanje ili "čitanje" programa; Potrebno je više od toga....

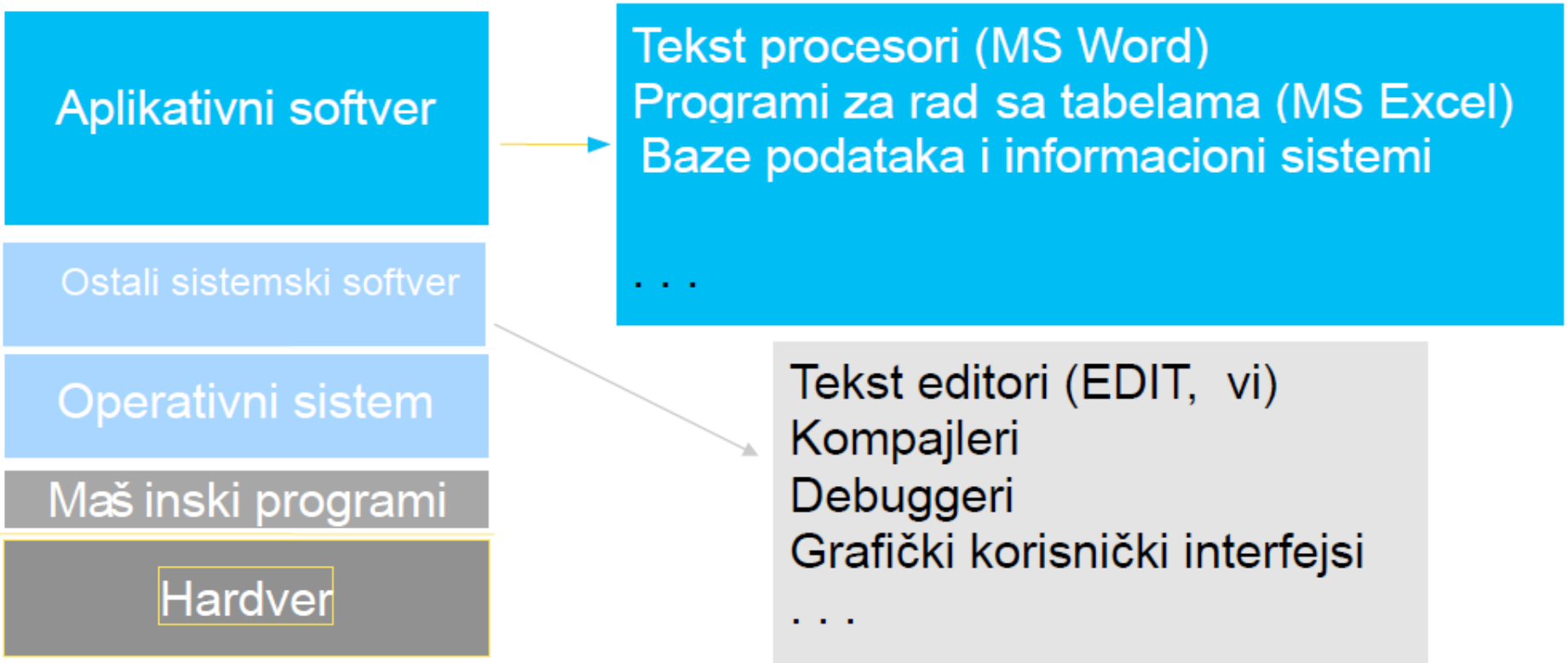
Program i programiranje

- *Programski kôd se učitava iz radne memoriji izvršava u procesoru instrukciju po instrukciju, onako kako je napisan*
- Naredba se prvo prebacuje u procesor (*fetch*), zatim se prevodi i potom izvršava

Adresa	Oznaka	Naredba	Binarni kod
		.begin	
		.org 2048	
	a_start	.equ 3000	
2048		ld [length],&r1	11000010 00000000 00101000 00101100
2052		ld [address],%r2	11000100 00000000 00101000 00110000
2056		addcc %r3,%r0,%r3	10000110 10001000 11000000 00000000
2060	loop:	addcc %r1,%r1,%r0	10000000 10001000 01000000 00000001
2064		be done	00000010 10000000 00000000 00000110

Višeslojni model softvera

- Softver se definiše slojevitom strukturom (*layers*)
- Svaki sloj definiše jednu funkcionalnost softvera
- Na najnižem sloju nalazi se hardver računarskog sistema



Tipovi softvera

Dve osnovne vrste softvera u računarskim sistemima:

1) SISTEMSKI SOFTVER (programi koji se razvijaju da obezbede funkcionisanje hardvera i softvera računarskog sistema); koordinira aktivnosti hardvera i programa i projektuje se uvek za specifičan procesor i specifičnu klasu hardvera; Vrste sistemskog softvera su operativni sistemi (OS) i pomoćni programi
2) APLIKATIVNI SOFTVER (programi koji se razvijaju za rešavanje konkretnih potreba korisnika računarskih sistema); aplikativni softver pomaže krajnjim korisnicima da uspešnije obavljaju svoj posao

- Korisnici softvera su pojedinci, radne grupe ili organizacije

Sistemski softver

- Sistemski softver čine:

a) BIOS (*Basic Input/Output System*) – startni program za računar

b) Operativni sistem (OS) – skup programa za upravljanje radom hardvera i ostalog softvera računara

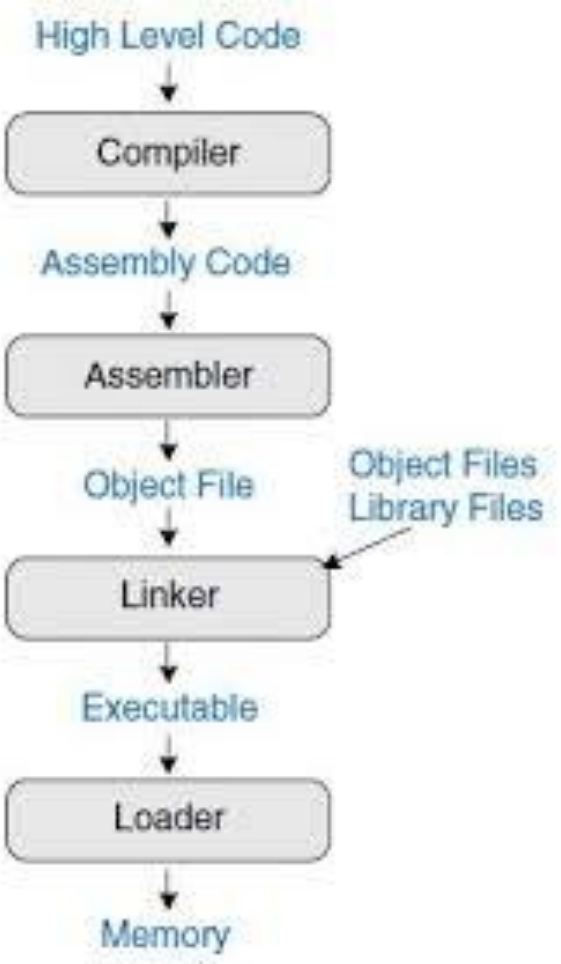
c) Programski alati: asembleri (*assemblers*), kompajleri (*compilers*), dibageri (*debuggers*) i korisnički alati (utilities)...

Sistemski softver

Sistemski softver čine:

- Prevodilac ili kompajler (*compiler*); To je program koji prevodi programe napisane u nekom višem programskom jeziku i vrši povezivanje naredbi izvornog kôda programa u izvršni kôd na mašinskom jeziku; Kompajleri pomažu pri pisanju programa po pravilima sintakse i pomažu pri otkrivanju grešaka u sintaksi tako što generišu poruke o greškama
- Linker je program za povezivanje, tj. povezuje programske module u jednu celinu
- Loader je program za „punjenje“ koji prenosi program u glavnu memoriju pre početka izvršavanja
- Asembler je program koji prevodi programe napisane u simboličkom jeziku na mašinski jezik

Prevođenje programskog kôda



- *Kompajler* je program koji na svom ulazu prihvata program napisan na nekom od viših programskih jezika nazvan izvorni program, a na svom izlazu generiše ekvivalentan program na mašinskom kôdu nazvan objektni program, tj. program u formi mašinskog kôda koji se može direktno izvršavati na hardveru računara (*Executable*). Objektni program koji se generiše na izlazu kompajlera upisuje se u sekundarnu memoriju
- Korisnički program poznat kao punilac (*loader*) smešta program iz sekundarne meorije u glavnu čime je objektni program spreman za izvršenje
- *Interpreter* uzima jednu instrukciju iz programa, analizira je i uslovljava da se sa istim efektom izvrši niz instrukcija nižeg nivoa. Ovaj proces se nastavlja sve dok se ne izvrši kompletan program

Generacije programskih jezika

Razlikujemo četiri klase računarskih jezika:

Generacija	Opis
prva generacija	mašinski jezik
druga generacija	asemblerski jezik
treća generacija	viši programski jezici (HLL)
četvrta generacija	novi jezici 4GL

Mašinski jezik

- Procesor i logički sklopovi računara imaju svoj vlastiti programski jezik koji se naziva mašinski jezik
- Mašinski jezik se sastoji od nizova binarnih reči (0 ili 1) koje predstavljaju instrukcije logičkim sklopovima i podatke koje treba obraditi; Program napisan u mašinskom jeziku nazivamo izvršni program (izvršni kôd) jer ga računar može neposredno izvršiti
- Mašinski jezik je određen arhitekturom računara, a izvršni program je mašinski zavisan, što znači da se kôd napisan na jednom računaru može izvršavati jedino na računarima istog tipa
- Instrukcije su numeričke u formi binarnih 0 i 1, programiranje je naporno i podložno velikom broju grešaka, efikasnost programiranja je niska, programi su nerazumljivi korisniku

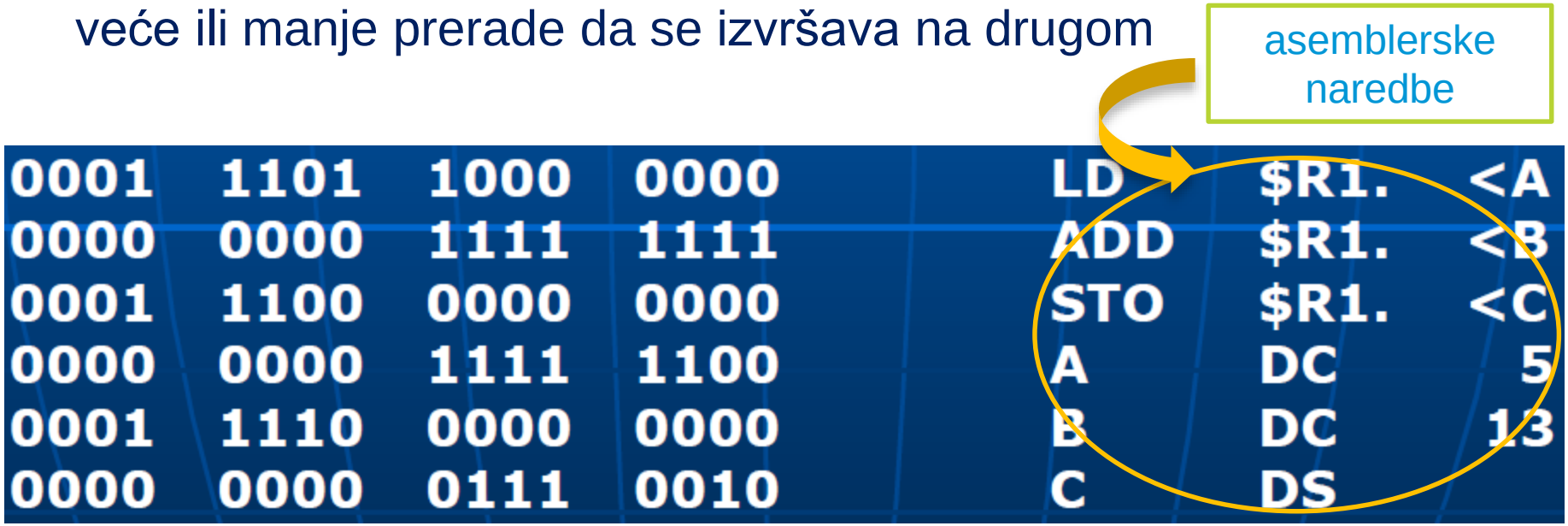
Mašinski jezik

- Prednosti mašinskog jezika: direktno se pristupa resursima mašine, veća je brzina izvršenja programa i efikasnije korišćenje memorije
- Uvek je potrebno poznavati skup naredbi datog računara tj. konkretni mašinski jezik; Program napisan za Intel računar neće moći da se izvrši na AMD računaru
- Sastavljanje i pisanje programa, kao i testiranje programa i podataka u memoriji računara zahteva detaljno poznavanje arhitekture računara i procesorske arhitekture

```
0001  1101  1000  0000
0000  0000  1111  1111
0001  1100  0000  0000
0000  0000  1111  1100
0001  1110  0000  0000
0000  0000  0111  0010
```

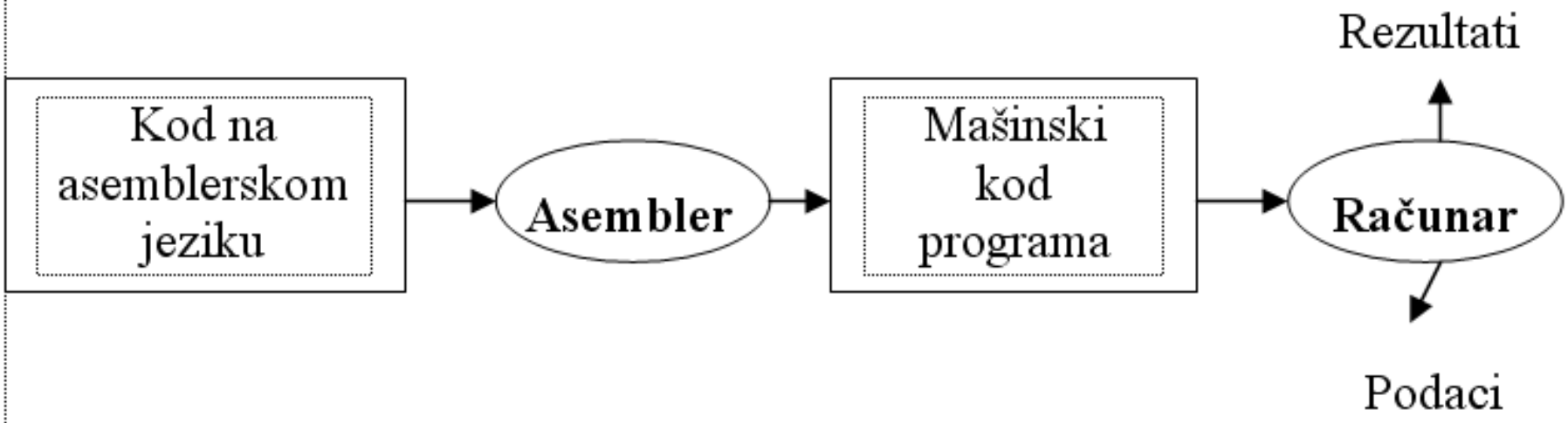
Asemblerski jezik

- Programiranje na simboličkom ili asemblerskom jeziku ima niz nedostataka: potrebno je vršiti detaljizaciju algoritma tako da elementarnim algoritamskim koracima odgovaraju dejstva asemblerskih naredbi,
- Raznovrsnost računara dovela je do raznovrsnosti asemblera, tako da svaki konkretni računar ili familija računara ima svoj poseban asemblerski jezik
- Program napisan na asembleru za jedan računar ne može bez veće ili manje prerade da se izvršava na drugom



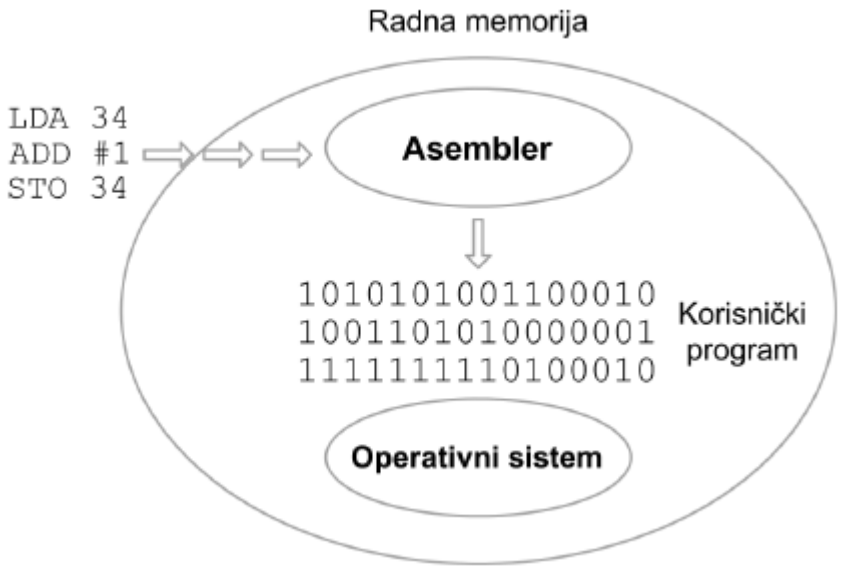
Asemblerski jezik

0001	1101	1000	0000	LD	$R1.	<A
0000	0000	1111	1111	ADD	$R1.	<B
0001	1100	0000	0000	STO	$R1.	<C
0000	0000	1111	1100	A	DC	5
0001	1110	0000	0000	B	DC	13
0000	0000	0111	0010	C	DS	



Asemblerski jezik

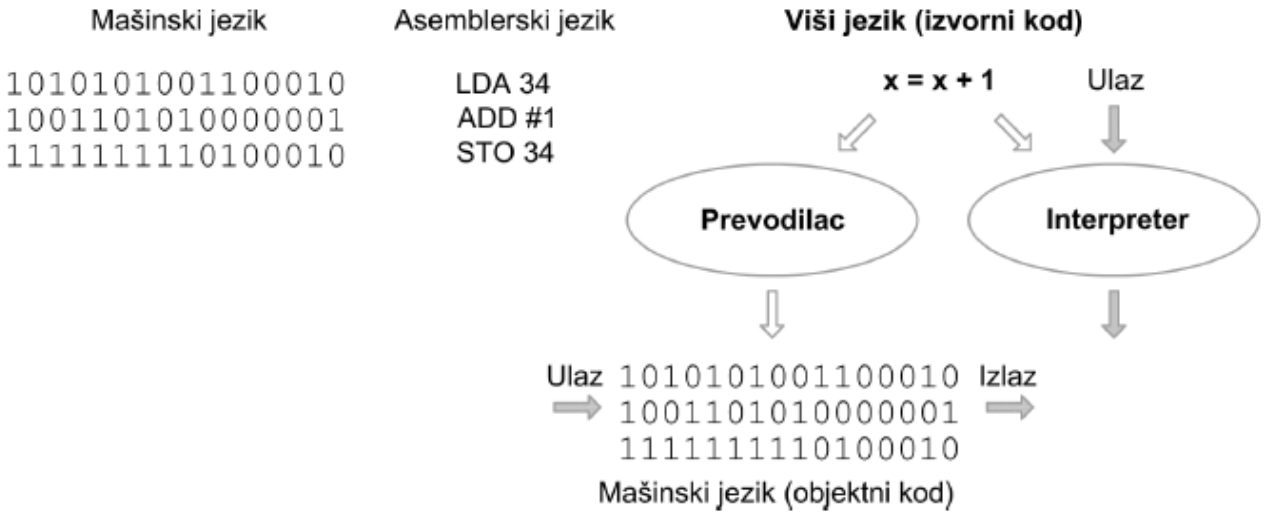
Program koji uvećava sadržaj memorijske lokacije sa adresom 34 za 1. Prvo se u akumulator procesora učitava vrednost sa željene lokacije (LDA adresa), na nju dodaje broj 1 (ADD #1), pa se rezultat upisuje na isto mesto u memoriji (STO 34).



U želji da se poveća izražajnost, kao i da se omogući što veća nezavisnost od tipa i organizacije sistema, uvodi se novi nivo apstrakcije: viši programski jezik.

Viši programski jezici (C, C++, Java, Pajton i drugi) pribiližavaju se prirodnom jeziku tako što uvode čitljive opise za složenije operacije i koncepte. Ovi opisi se transformišu u niz elementarnih operacija za ciljni računarski sistem.

Asemblerski jezik



Program napisan u višem programskom jeziku označava se kao izvorni kod. Izvorni kod je lako razumljiv za čoveka, ali se mora prevesti u mašinski jezik, uz pomoć drugog programa – prevodioca. U procesu prevođenja, naredbe višeg programskog jezika zamenjuju se brojnim mašinskim instrukcijama koje realizuju željeni efekat, a dobijeni rezultat često se zove i objektni kod.

Drugi pristup u izvršavanju izvornog koda, umesto prevodioca, koristi program interpreter. Interpreter čita naredbe višeg jezika i izvršava ih jednu po jednu, koristeći sopstveni skup operacija. On preuzima ulogu kontrolora koji, u ime programa, zahteva unos podataka sa ulaza i prikazuje rezultate na izlazu

Jezici višeg nivoa HLL

- Karakteristike višig programskih jezika HLL (*High-Level Languages*) su:
  - Kompajler prevodi programske iskaze u odgovarajuće sekvence instrukcija na mašinskom nivou
  - Programiranje je znatno olakšano jer su ovi jezici bliski čovekovom pisanom jeziku, skraćeno je vreme obuke da se nauči programiranje
  - Jedan iskaz na HLL-u se prevodi u više instrukcija na mašinskom (asemblerskom) jeziku
  - Programiranje je jednostavnije i efikasnije, programi su duži
  - Ispravljanje grešaka je lakše
  - Nema direktnog pristupa resursima mašine, neefikasno je iskorišćenje memorije

Jezici 4GL i primena

- Novi tipovi računarskih jezika se karakterišu sledećim osobinama:
  - implementiraju veštačku inteligenciju (primer je LISP jezik)
  - jezici za pristup bazama podataka (primer je SQL)
  - objektno-orijentisani jezici (primeri su C++, Java i dr.)
- Prema oblasti primene programski jezici služe za: naučne aplikacije, poslovnu obradu, veštačku inteligenciju, projektovanje sistemskog softvera, projektovanje pomoću računara (CAD), upravljanje pomoću računara (CAM), opis hardvera računara, simboličko programiranje, linearno programiranje, simulacije, računarske komunikacije, prostorno planiranje, stono izdavaštvo, obradu teksta (tekst procesori)...

OS za mobilne uređaje

OS za mobilne uređaje

- Zbog velike popularnosti pametnih telefona i tablet uređaja očekuje se da će vrednost tržišta mobilnih aplikacija u narednim godinama iznositi preko 100 milijardi dolara
- Broj preuzetih aplikacija raste iz godine u godinu; U 2008. godini bilo je nešto više od 2 milijarde preuzimanja a doo 2010. taj broj je porastao za 300%, (na 8.2 milijardi), a procene pokazuju da će do 2019. godine broj preuzetih mobilnih aplikacija godišnje iznositi 70 milijardi
- Znanje razvoja aplikacija za mobilne uređaje jako je traženo i finansijski isplativo bilo da samostalno prodajemo vlastitu aplikaciju, ili koristimo svoje znanje za realizaciju tuđih ideja i izradu aplikacija u okviru neke IT kompanije

OS za mobilne uređaje

- Najpopularniji operativni sistemi za mobilne telefone su: Android, Apple (iOS), Windows Phone, BlackBerry, Symbian
- **Android** je OS za mobilne telefone kompanije Google i trenutno je najbrže rastući operativni sistem na tržištu; Android je postao vodeća softverska platforma sa najvećim udelom na tržištu mobilnih telefona, smatra se da će do 2018. imati oko 85% udela na tržištu
- **Windows Phone** (nekada Windows Mobile) su operativni sistemi kompanije Microsoft koji je posebno optimizovan za mobilne telefone novije generacije (video-snimanje u 4K rezoluciji, Continuum mogućnost povezivanja sa TV uređajima...) ali na tržištu više nema bitniju ulogu jer je dalji razvoj prekinut.

OS za mobilne uređaje

- Za razliku od Androida, **iOS** je zatvoreni operativni sistem koji je vlasništvo američke kompanije Apple i namenjen je isključivo za njihove uređaje, a pokretanje na hardveru drugih proizvođača je onemogućeno
- Broj modela uređaja koji pokreće ovaj operativni sistem je relativno mali, što doprinosi dobroj optimizaciji i boljoj iskorišćenosti hardverskih resursa, a samim tim i boljem korisničkom doživljaju
- Prva verzija iOS-a 2007. godine za prvu generaciju iPhone uređaja
- Trenutno je aktuelna verzija iOS 12.1.1 (decembar 2018.)

OS za mobilne uređaje

- **BlackBerry OS** je operativni sistem za mobilne telefone bivše kompanije RIM (*Research in Motion*). Blackberry se razlikuje od ostalih operativnih sistema i po dizajnu i po korisničkom interfejsu jer ovi uređaji najčešće poseduju hardversku tastaturu (trackball, trackwheel i trackpad)
- Blackberry Enterprise Server (BES) je softverski paket koji se koristi za integraciju mobilnih uređaja kompanije sa korporacijskim e-mail sistemom. Svaki Blackberry uređaj se identifikuje uz pomoć BES-a sa individualnim i jedinstvenim Blackberry PIN kodom
- BlackBerry 7.1 (2012) je poslednja verzija ovog OS i dostupna je na telefonima BlackBerry Curve serije (9310/9315/9220/9320) i BlackBerry 9720.

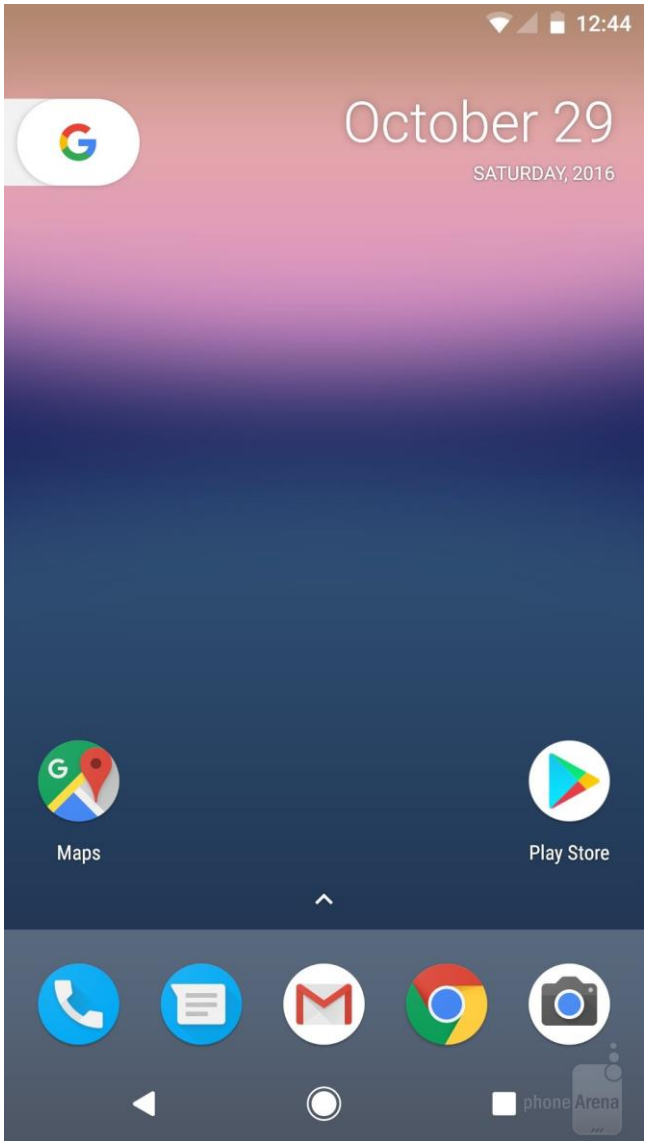
OS za mobilne uređaje

- **Symbian** je operativni sistem koji su koristile kompanije Nokia, Samsung, Motorola, SonyEricsson na svojim mobilnim telefonima
- Najveća snaga ovog operativnog sistema za mobilne telefone ogledala se u njegovoj funkcionalnosti, kao i u mogunosti preuzimanja aplikacija
- Sam Symbian OS nema svoj interfejs na mobilnim telefonima, a ono što se koristilo u te svrhe su bile softverske platforme kao što su S60, UIQ ili MOAP
- Nokijina Symbian platforma pod nazivom S60 ubedljivo je bila najrasprostranjenija, ali ovaj OS danas nije aktuelan, naročito posle kupovine Nokia sektora za mobilne telefone od strane kompanije Microsoft i ukidanja istog

OS za mobilne uređaje

- **WebOS** je operativni sistem za mobilne uređaje zasnovan na Linuxu, inicijalno kreiran od strane kompanije Palm, kasnije preuzet od strane kompanija HP i LG Electronics
- Palm je objavila WebOS u januaru 2009, i tada se nazivao Palm WebOS; Postojale su različite verzije ovog OS-a kreirane za različite uređaje, pre svega za Pre, Pixi, i Veer telefone i HP TouchPad tablet
- Poslednja verzija 3.0.5, objavljena je početkom 2012.
- **Bada OS** je operativni sistem za mobilne uređaje kompanije Samsung, kreiran 2010 i prvi put primenjen na uređaju Wave S8500; Nije imao veći uspeh i uglavnom se koristi za Samsungove uređaje; Nova verzija je Bada 2.0 a iz nje je nastala verzija pod nazivom **Tizen**

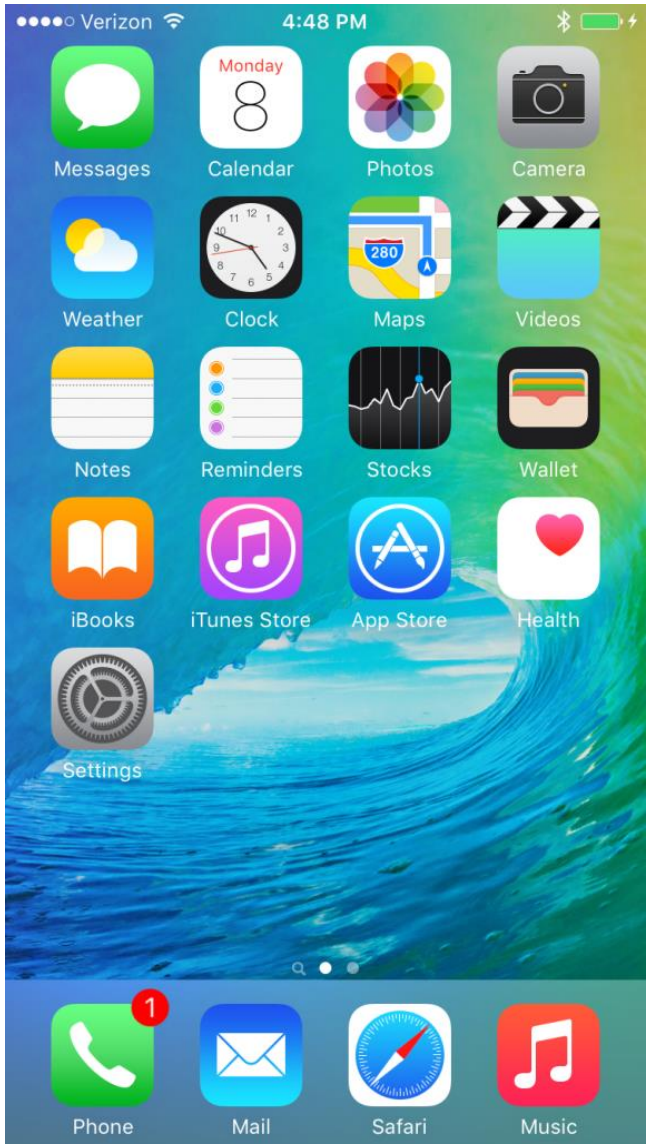
OS za mobilne uređaje



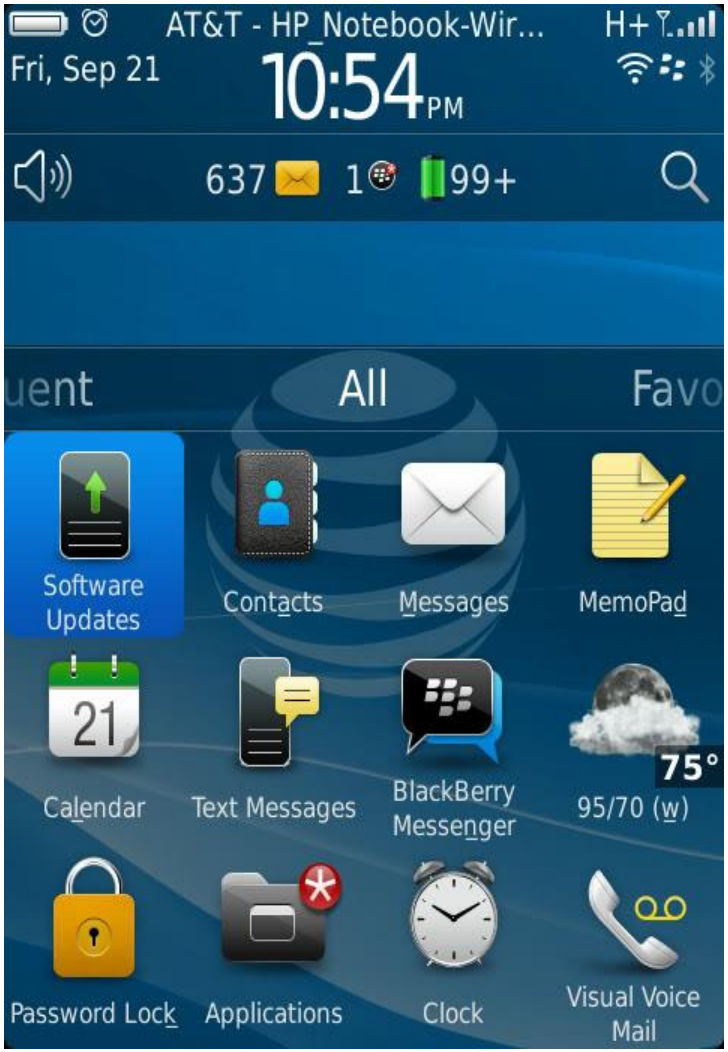
Android OS

Windows 10 Mobile OS

OS za mobilne uređaje



Apple iOS 10



BlackBarry Mobile 10 OS

Programski jezici

Pascal

PASCAL

- Programski jezik Pascal razvio je 1970. švajcarski naučnik Niklaus Virt, na temeljima jezika ALGOL, radi prevazilaženja teškoća koje su se javljale programiranjem u programskim jezicima sekvencijalnog karaktera (FORTRAN, COBOL, ALGOL i BASIC)
- Pascal uvodi paradigmu strukturnog i proceduralnog programiranja, što znači da se programski kod razlaže na samostalne strukture - podatke i podprograme, koji se ne izvršavaju u istom redosledu u kom su navedeni, kao što je slučaj kod naredbi sekvencijalnih programskih jezika, nego se po potrebi pozivaju i izvode
- Uvođenjem podprograma izbegava se nužnost da program sekvencijalno, to jest redosledno, izvrši sve blokove koda; Za tadašnje doba vrlo skupih memorija i početka ubrzanog razvoja softvera u odnosu na hardver, Pascal je imao veliki značaj
- 70-ih godina Pascal stiče veliku popularnost pa se u velikoj meri koristi za aplikacijsko i sistemsko programiranje, ali se danas sve ređe koristi jer nije objektno-orijentisan jezik

C

- Programski jezik **C** je proceduralni programski jezik, nastao 1972. godine. Autor jezika je Denis Riči, a nastao je u istraživačkom centru Bell Laboratories u Nju Džersiju za potrebe operativnog sistema UNIX
- C može funkcionisati na veoma hardverski „slabim“ računarima
- U C-u se puno koriste funkcije, čak je i glavni program jedna velika funkcija
- C je strukturni je jezik (definišu se strukture podataka)
- Dostupno je programiranje na niskom nivou (sa bitovima)
- Izvanredna primena pokazivača (*pointers*) za memoriju, niz, strukture i funkcije
- Iako C nije objektno-orijentisan programski jezik, danas je još uvek mnogo u upotrebi jer hardverski nije zahtevan, brzo se izvršava i ima sjajne načine upravljanja memorijom; Veliku primenu ima kod sistema za ugradnju (*embedded sistemi*)

C++

- C++ je viši programski jezik koji je prvobitno razvijen u Bell Labs tokom 80-ih kao objektno-orijentisano proširenje C programskog jezika
- Programski jezik C++ podržava objektno-orijentisano programiranje što je programerima omogućilo da lakše definišu kompleksnije programe. Jezik C++ bio je jedan od prvih sa podrškom za klase (način na koji se definišu objekti)
- C++ je jezik "srednjeg nivoa". Zadržava se kontrola nad računarom kao kod asemblera i zadržavaju se dobre strane jezika "visokog nivoa". C++ je portabilan ili prenosiv u smislu da se može izvršiti, sa minimalnim promenama, na više različitih računarskih sistema bez modifikacija
- C++ jezik je jedan od najpopularnijih programskih jezika korišćenih za razvoj računarskih aplikacija
- Objektno orijentisano programiranje i C ++ kao odgovor daju: apstraktne tipove podataka, enkapsulaciju, nasleđivanje, polimorfizam itd.

HTML5

- Za kreiranje front-end web aplikacija za mobilne uređaje, skoro je neminovna upotreba jezika HTML5 (*Hypertext Markup Language*)
- Za razliku od programskih jezika koji koriste instrukcije koje se prevode u mašinski kod, HTML je jezik sastavljen od "tagova" i "atributa" čijim se prevođenjem prikazuje jasan izgled neke internet stranice
- Kako je za prikaz internet stranica najbitniji tekst, kao veoma jednostavno rešenje se nametnulo korišćenje "markup-a" odnosno opisnih parametara uz tekst koji definišu način prikaza teksta: podebljan tekst, iskošen, podvučen, centriran, pasusi, paragrafi i sl.

** tekst **

ovo će u browser-u biti prikazano kao boldovan tekst

tekst

HTML5

- Problem HTML-a je nedostatak “dinamičnosti” kao i klasičnih “programerskih” zahteva za razvoj ozbiljnih aplikacija
- Problem je rešen korišćenjem programskih jezika, pre svega *Java* i *JavaScript* koji su uveli mogućnost “programiranja” internet stranica, a *CSS* (*Cascading Style Sheets*) je uveo nove mogućnosti u sfere dizajniranja internet stranica, dok je PHP kao tehnologija uveo revoluciju u kreiranju dinamičkih back-end stranica
- HTML 5 uvodi novine po pitanju prikazivanja multimedijalnog sadržaja, ali i pruža jaču podršku ka integraciji sa programskim jezicima pri kreiranju komplikovanih veb aplikacija
- HTML 5 okuplja sve standarde koji se koriste u kreiranju internet prezentacija: HTML, CSS i JavaScript biblioteke i na taj način se lako kontroliše razvoj standarda i pretraživači mogu lakše da prate sve promene
- Zbog ovakvih karakteristika HTML 5 je postajo standard i za mobilne platforme

HTML5 kôd

```
<!DOCTYPE html>
<html>
  <head>
    <title>Canvas-Rotation</title>
    <meta charset="UTF-8" />
    <style>
      #square {
        border: 1px solid black;
        transform: scale(10) rotate(3deg) translateX(0px);
        -moz-transform: scale(10) rotate(3deg) translateX(0px);
      }

      .box {
        transition-duration: 2s;
        transition-property: transform;
        transition-timing-function: linear;
      }
    </style>
  </head>
  <body>
    <canvas id="square" width="200" height="200"></canvas>
    <script>
      var canvas = document.createElement('canvas');
      canvas.width = 200;
      canvas.height = 200;

      var image = new Image();
      image.src = 'images/card.png';
      image.width = 114;
      image.height = 158;
      image.onload = window.setInterval(function() {
        rotation();
      }, 1000/60);
   </script>
  </body>
</html>
```

Java

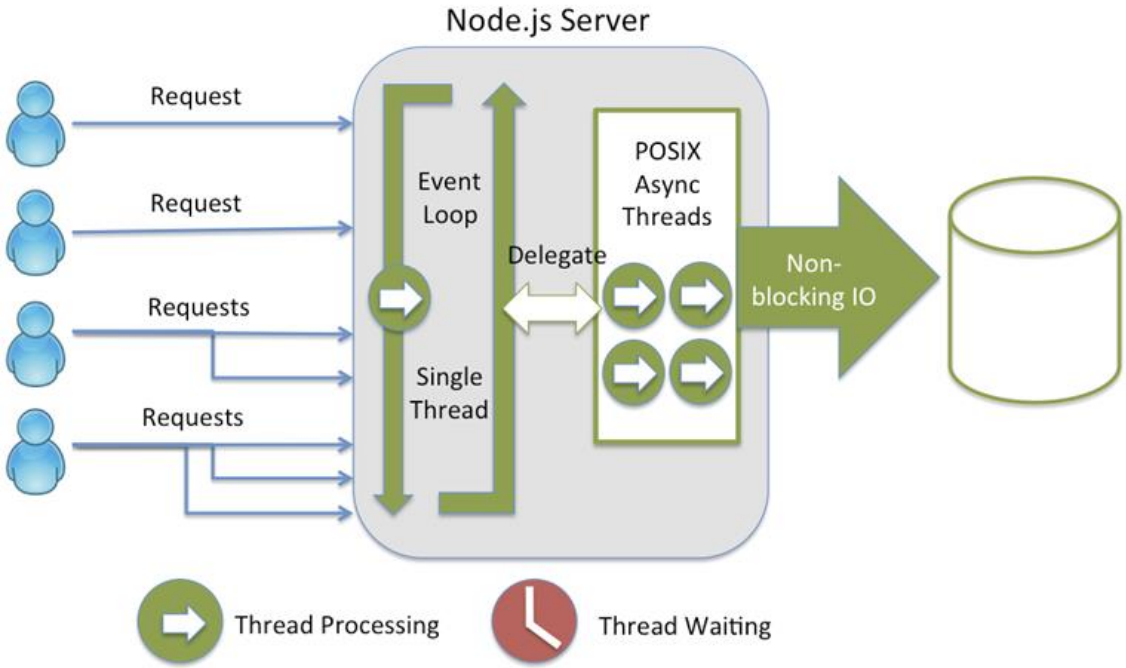
- Softver za mobilne uređaje možemo razvijati i primenom Java, JavaScript, i Enterprise Java Beans jezika. Ovo su tri različite softverske platforme koje se baziraju na jeziku Java
- Java je **objektno-orijententisan programski jezik** razvijen od strane firme Sun Microsystems (koja je sada u vlasništvu korporacije Oracle)
- Java je jedan od najviše korišćenih programskih jezika, i nudi čitav niz servisa i modula
- Java se može izvršavati na dva različita načina:
  - u okviru pretraživača
  - na virtuelnoj mašini koja ne zahteva pretraživač (Android)
- Java je „nezavisna od platforme“ jer omogućava da se njen kod izvodi bez unošenja promena u različitim OS-ima (Windows, Linux, UNIX, …).
- Java je dizajnirana za izvršavanje u distribuiranom okruženju, prvenstveno na internetu

JavaScript

- JavaScript je najpopularniji skriptni jezik kojeg podržavaju svi poznatiji pretraživači (Internet Explorer, Mozilla, Firefox, Opera...)
- *Cilj kreiranja JavaScript jezika je dodati interaktivnost HTML stranicama*
- Skriptni jezici su programski jezici manjih mogućnosti, koji se sastoje od od izvršnog koda, obično direktno ugrađenog u HTML stranice; JavaScript je **interpreter**, što znači da se programska skripta izvršava odmah naredbu po naredbu, bez prethodnog prevođenja (kompajliranja) celog programa i kreiranja izvršne datoteke
- Jezik JavaScript nije isto što i Java jezik !!!
- JavaScript nastavlja da se unapređuje i razvija i kombinacija JavaScripta sa CSS3 i HTML5 jezicima programeru dopuštaju jako velike mogućnosti kreiranja web aplikacija

Node.js

- **Node.js** je open-source softver za pisanja potpuno skalabilnih JavaScript server-side Web aplikacija i omogućava korišćenje jedinstvenog jezika između front-end i back-end strane
- Node.js je se izuzetno brzo izvršava, i koristi asinhroni event-driven model programiranja (upravljanje događajima) za izradu modernih web aplikacija; zbog toga je pogodan za pisanje real-time web aplikacija



Uvek se izvršava samo jedna programska nit (*thread*). Kada se izvršavaju neke spore I/O operacije (npr. čitanje podataka iz baze), program ne čeka, nego ide na izvršavanje sledeće linije koda; Kada se I/O operacija vrati, pokreće se callback funkcija i rezultat se procesuira.

Node.js

- **Node.js** je kreiran je 2009. na osnovu Google V8 JavaScript engine-a i koristi minimalne JavaScript core biblioteke i upravljanje programskim nitima (*threads*)
- Node.js je lakši za korišćenje zbog ugrađene podrška za upravljanje paketima pomoću NPM (*Node Package Manager*) alata koji podrazumevano dolazi sa svakom Node.js instalacijom
- Chat-aplikacija je tipičan primer real-time, multi-user aplikacije i najbolji primer gde Node.js dolazi do izražaja

Ruby / Ruby on Rails

- **Ruby** je dinamički, objektno-orjentisani jezik prvenstveno za funkcionalno programranje Web aplikacija, koji nudi mogućnost za razvoj *streamlined* procesa. Ruby je višeplatformski jezik otvorenog koda. Postao jako popularan dolaskom *Ruby on Rails* framework-a za web aplikacije zasnovane na bazama podataka
- ***Ruby on Rails*** je dizajniran za kreiranje web aplikacija od postojećih šema baza podataka. On dodaje ključne reči u Ruby da bi napravio web aplikacije koje su lakše za konfigurisanje
- Ruby sintaksa je slična jezicima Perl i Python a postoje sličnosti sa Smalltalk-om, i najviše ga koriste Java i PHP programeri
- Ruby on Rails radi sa web serverima kao što je Apache, i sa bazama podataka uključujući MySQL, PostgreSQL, SQLite, Oracle database, SQL Server i DB2

Objektni C

- Mada je C++ za razvoj softvera najkorišćeniji programski jezik na svetu, Apple koristio sopstveni jezik **Objective-C** kao svoj primarnim programski jezik; Objective-C je bio potpuno integrisan u sve ranije Apple iOS i OSX sisteme
- Objective-C se više ne koristi kao primarni jezik za Apple i zamenio ga je Swift
- Objective-C podržava skoro sve kao jezici C i C++, i ima veliki broj funkcija - mogućnost rada sa grafikom, I/O sistemima i funkcijama grafike; iPhone i iOS sistemi bili su programirani na njegovoj bazi preko Apple XCode integrisanog razvojnog okruženja (IDE)
- Korišćenjem Objective-C kompajlera mogu se kompajlirati C datoteke , a u klasama se potpuno slobodno može koristiti C kod
- Objektna sintaksa ovog jezika je derivat iz Smalltalk-a

Swift

- Swift je jezik razvijen u kompaniji Apple sa namerom da postane glavni jezik za razvoj softvera na iOS i OS X platformama
- Swift je potpuno nov jezik razvijen od nule, ali većini koda napisanog u Objective-C može se pristupiti iz Swifta. Pošto Swift ima mogućnost korišćenja bilo koje biblioteke napisane u Objective-C jeziku, to znači da su poslednji API-i (Cocoa i Cocoa Touch) potpuno dostupni iz Swifta
- Swift se koristi za kreiranje kompletnih iOS i OS X aplikacija
- Sintaksa Swift jezika je slična sintaksi savremenih jezika kao što su Java, C# ili JavaScript, što znači da je poznata i bliska većini programera

Microsoft C#

- **C#** (C-*sharp*) je objektno orijentisan programski jezik opšte namene, s obaveznom proverom tipova podataka. Koristi se za kreiranje konzolnih, Windows i Web aplikacija
- Spada u kategoriju najpopularnijih programskih jezika i odlikuje ga produktivnost, jednostavnost i efikasanost programiranja
- C# je neutralan u pogledu platforme, ali je napisan za Microsoft .NET Framework okruženje . Verzija C# 5.0 razvijena je za .NET Framework 4.5.
- Sadrži 70% jezika Java, 10% C++, 5% Visual Basic-a, 15% je sopstveni kod

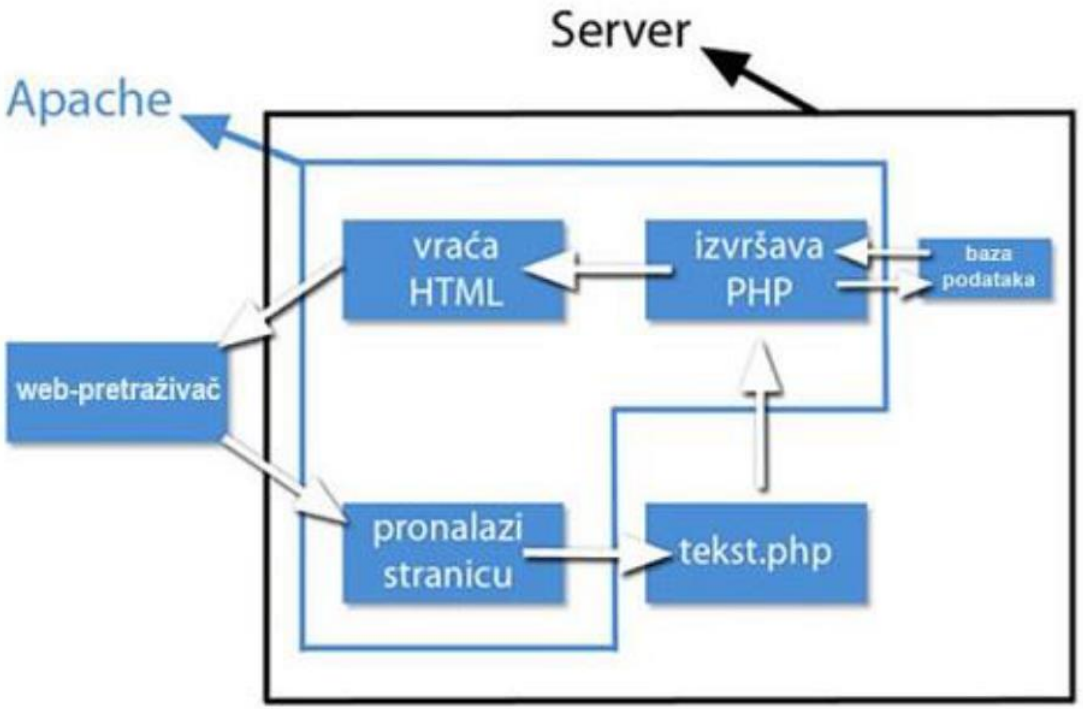
PHP

- PHP predstavlja skraćenicu od izraza **Hypertext Processor**.
- PHP je skriptni jezik namenjen upotrebi na webu i koristi se za server-side aplikacije
- Može se koristiti na većini web servera i u gotovo svim operativnim sistemima i platformama jer je open-source
- PHP se ugrađuje u HTML stranice i njegovom primenom dobijamo generisanu stranicu u čistom HTML-u. Pogodan je za razvoj naprednih web aplikacija koje obuhvataju izradu web shop-ova, internet portala, foruma, sajtova specijalizovanih za internet socijalne mreže (Facebook, Twitter, Google+) i izradu CMS-ova (Content Management System - Joomla, WordPress, i sl.).

PHP

- Pretraživač šalje zahtev serveru da pronađe željenu stranicu na serveru, koji hostuje taj web-sajt
- Pošto ima nastavak .php, server će procesuirati celokupan php kod na koji nailazi u toj datoteci

Tokom procesiranja PHP tražiti da se izvuku podaci iz baze podataka (SQL, MySQL), nešto ubaci u nju, osveže neki podaci...
Nakon što je ceo PHP kod izvršen, nazad se šalje HTML rezultat (u .php formatu) koji vidimo na našem web pretraživaču.



PERL

- Perl je interpretirani jezik opšte namene, (izuzetno) visokog nivoa. Koristi se uglavnom za izdvajanje i obradu podataka iz tekstualnih datoteka, formatiranje i ispis rezultata tih operacija.
- Pravi prodor, međutim, doživeo je tek sa popularizacijom HTML-a i Web-a. Perl je besplatan programski jezik
- Obrada podataka i njihov formatiran ispis je sama suština dinamičkog kreiranja HTML stranica, a za te oblasti Perl spada u vodeće alate
- Osnovne karakteristike Perl-a su: sintaksa po uzoru na C, veoma slaba tipizacija promenljivih, razvijene mogućnosti procesiranja teksta, moćna podrška za rad sa listama i asocijativnim nizovima, jednostavan pristup datotekama, bazama podataka i TCP/IP socket-ima, mogućnost modularnog i/ili objektnog programiranja i POSIX kompatibilnost

Python

- Pajton je osmislio holandski programer Guido van Rossum krajem ‘80-ih godina prošlog veka
- Pajton je open-source interpretirani, objektno-orijentisani jezik visokog nivoa; Kôd se ne prevodi već se izvršava direktno u intepreteru
- Naročito veliku primenu poslednjih godina Pajton ima kod implementacije mašinskog učenja (machine learning), razvoja neuronskih mreža i nauke o podacima (Data Sciences)
- Pravila jezika su jednostavna, a izvorni kod je čitak i često dosta kraći nego u slučaju Java ili C/C++ ekvivalenta
- Dostupan je za različite operativne sisteme poput Windows-a, Linux-a i Mac OS-a
- Trenutna verzija je Python 3.7.1 (decembar 2018)

Python

- Izuzetno čitljiv i jednostavan kod, pa se preporučuje kao prvi programski jezik za učenje

"Hello, World"

- C

```
#include <stdio.h>

int main(int argc, char ** argv)
{
   printf("Hello, World!\n");
}
```

- Java

```
public class Hello
{
   public static void main(String argv[])
   {
      System.out.println("Hello, World!");
   }
}
```

- now in Python

```
print "Hello, World!"
```

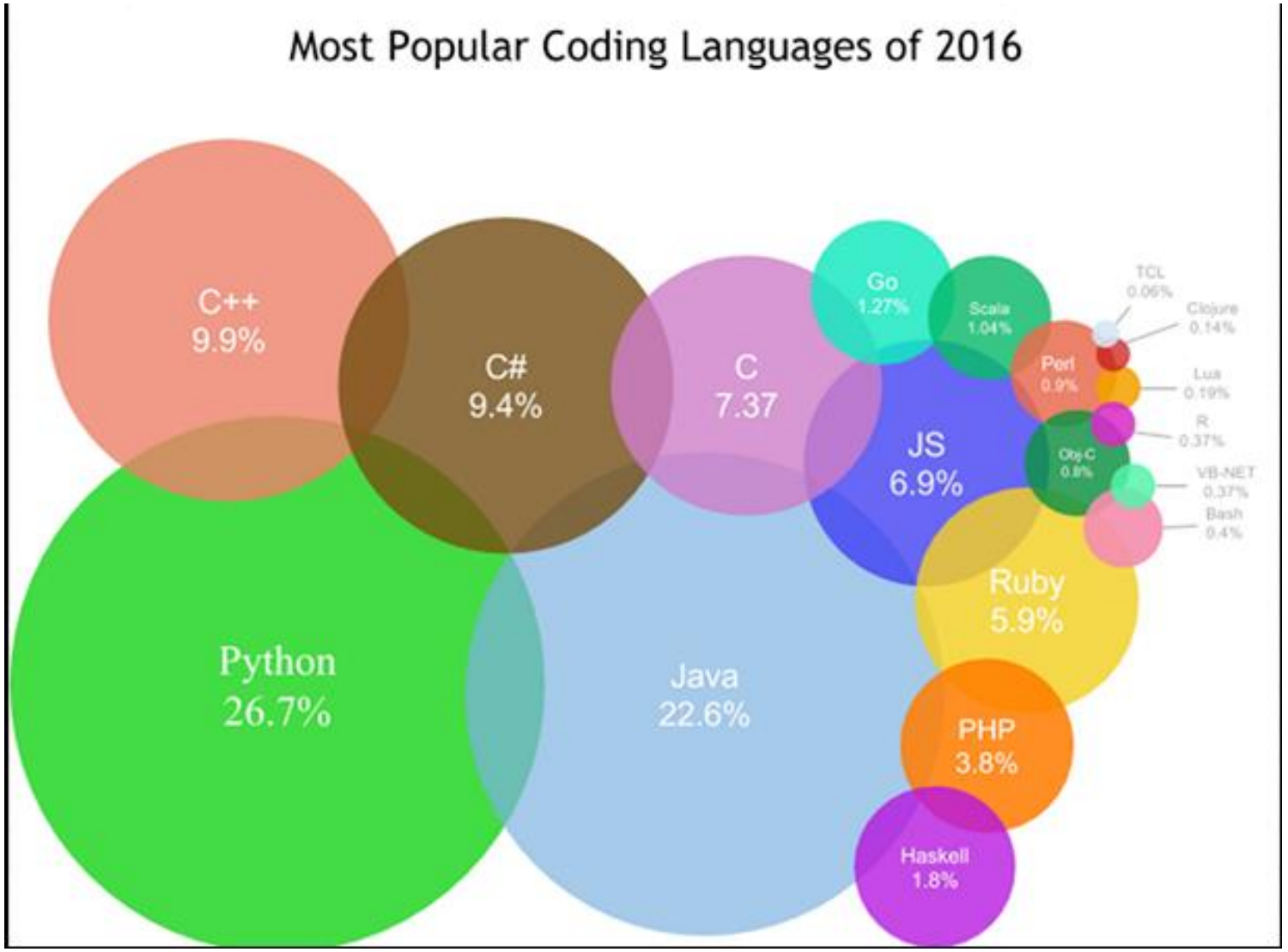
Python

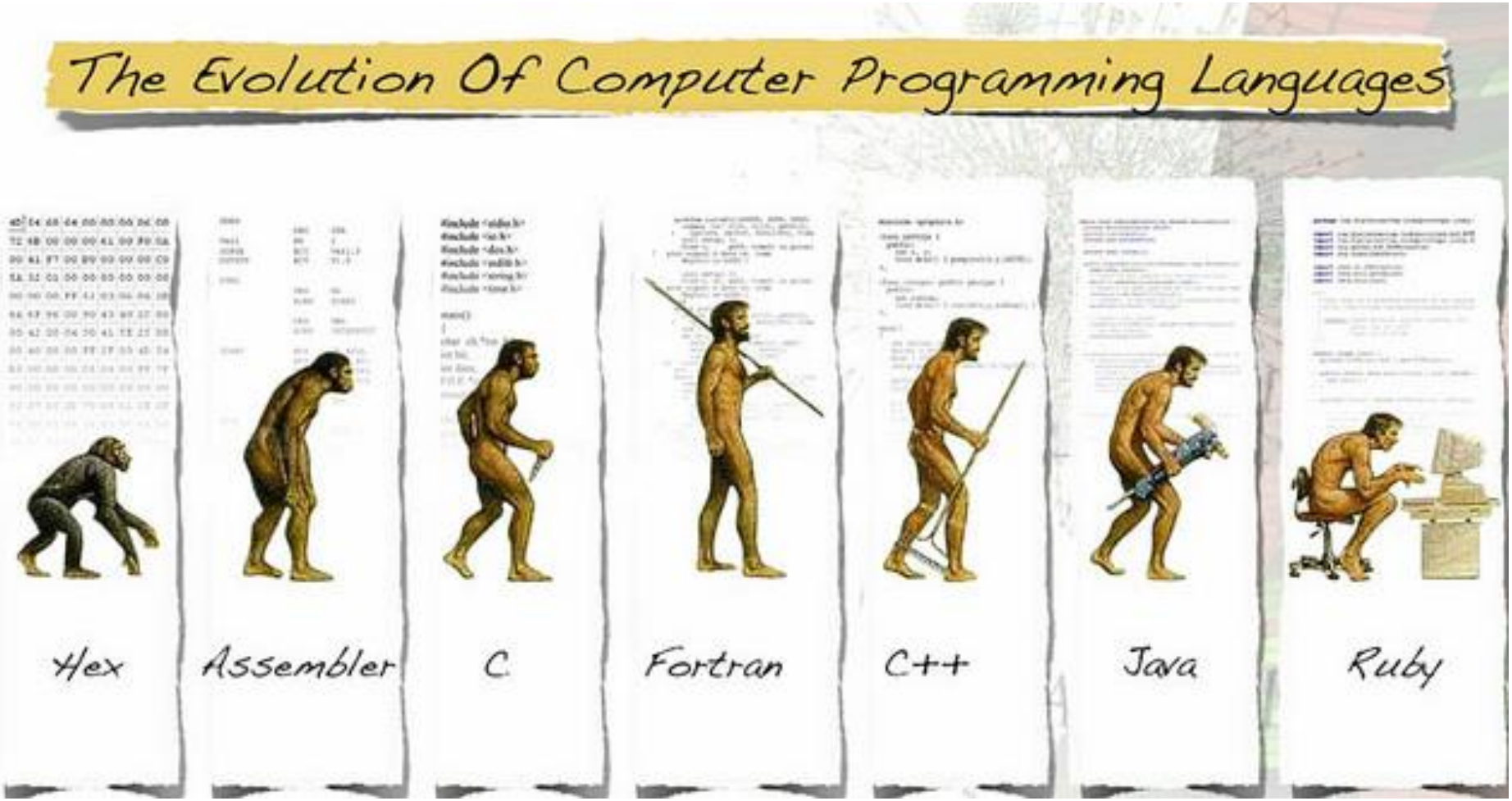
- Najosnovnije korišćenje Pajtona je kao jezik skriptovanja i automatizacije; Pajton nije samo zamena za skriptove ili batch fajlove, već se koristi za automatizaciju interakcija sa web pretraživačima ili GUI-jevima
- Ogromna većina biblioteka korišćenih za nauku o podacima ili mašinsko učenje ima Pajton interfejs, čineći ovaj jezik najpopularnijim komandnim interfejsom visokog nivoa za biblioteke mašinskog učenja i druge numeričke algoritme
- Pajton se koristi za web usluge i REST API-eve. Matične biblioteke Python-a i web-okruženja nezavisnih proizvođača pružaju brze i praktične načine za kreiranje svega, od jednostavnih REST API-ja u nekoliko redova koda, do potpuno obrađenih lokacija upravljanih podacima
- Pajton se koristi za metaprogramiranje. U jeziku Pajton sve je objekat, uključujući i Pajton module i same biblioteke. Ovo omogućava Pajton-u da radi kao visoko efikasan generator koda, omogućavajući pisanje aplikacija koje manipulišu sopstvenim funkcijama i imaju vrstu proširivosti koja bi bila teško ili nikako izvodljiva u drugim jezicima.

TIOBE Index 2017

Nov 2017	Nov 2016	Change	Programming Language	Ratings	Change
1	1		Java	13.231%	-5.52%
2	2		C	9.293%	+0.09%
3	3		C++	5.343%	-0.07%
4	5	↑	Python	4.482%	+0.91%
5	4	↓	C#	3.012%	-0.65%
6	8	↑	JavaScript	2.972%	+0.27%
7	6	↓	Visual Basic .NET	2.909%	-0.26%
8	7	↓	PHP	1.897%	-1.23%
9	16	⇈	Delphi/Object Pascal	1.744%	-0.21%
10	9	↓	Assembly language	1.722%	-0.72%

Popularnost programskih jezika 2016




The Evolution Of Computer Programming Languages
Hex
Assembler
C
Fortran
C++
Java
Ruby

Android OS

Poreklo Android OS

- Android je mobilni operativni sistem koji je zasnovan na modifikovanoj verziji Linux operativnog sistema. Originalno ga je razvila „Android, Inc.“ kompanija
- U toku 2005. godine, kao deo strategije za pristupanje tržištu mobilnih uređaja, „Google“ je kupio Android i preuzeo odgovornost za njegov dalji razvoj (kao i za razvojni tim)
- Programski kod Android OS-a je javno dostupan i funkcioniše pod **Apache License** programom
- Dizajniran za mobilne uređaje kao što su smart telefoni i tablet računari
- Razvijen od strane Google-a u saradnji sa OHA (*Open Handset Alliance*) koju je činilo 86 kompanija koje su kreirale otvorene standarde za mobilne uređaje
- Inicijalnu verziju je razvila kompanija Android Inc. (2003.)
- Prvi Android OS je objavljen 2007.godine

Prednosti Android OS

- Projekat AOSP (*Android Open Source Project*) bazira se na održavanju i daljem razvoju
- Pored Google-a, Android ima zajednicu sa velikim brojem samostalnih programera koji menjaju sam kod operativnog sistema (Custom ROM) kao što je npr. XDA-Developers
- Sve modifikacije se pišu pomoću **programskog jezika Java** u prilagođenoj verziji za Android OS
- Za Android postoji veliki broj aplikacija (Apps) koje nude različite funkcionalnosti
- Sve aplikacije su dostupne putem Google Play servisa

Karakteristike Android OS

Unificirani pristup razvoju aplikacija je jedna od glavnih prednosti Android operativnog sistema

- Sva softverska rešenja se razvijaju pod Android platformom i izvršavaju na uređajima koje pokreće neka od verzija Android OS
- Android je open-source softver, alati i tehnologije za prilagođavanje i unapređenje, kao i aplikacija koje se izvršavaju pod Androidom dostupni su potpuno besplatno

Karakteristike Android OS

- Android je softverska platforma dizajnirana i namenjena za mobilne uređaje
- Android je više od samog operativnog sistema, to je softverski stek koji čine:
  - ✓ operativni sistem
  - ✓ osnovne biblioteke i izvršno okruženje
  - ✓ radni okvir za aplikacije (application framework)
  - ✓ aplikacije
- Android softverska platforma je dizajnirana tako da:
  - ✓uzima u obzir ograničenja koja nameću mobilne platforme: procesorska moć, memorija, baterija…
  - ✓služi potrebama širokog spektra mobilnih uređaja

Android OS verzije

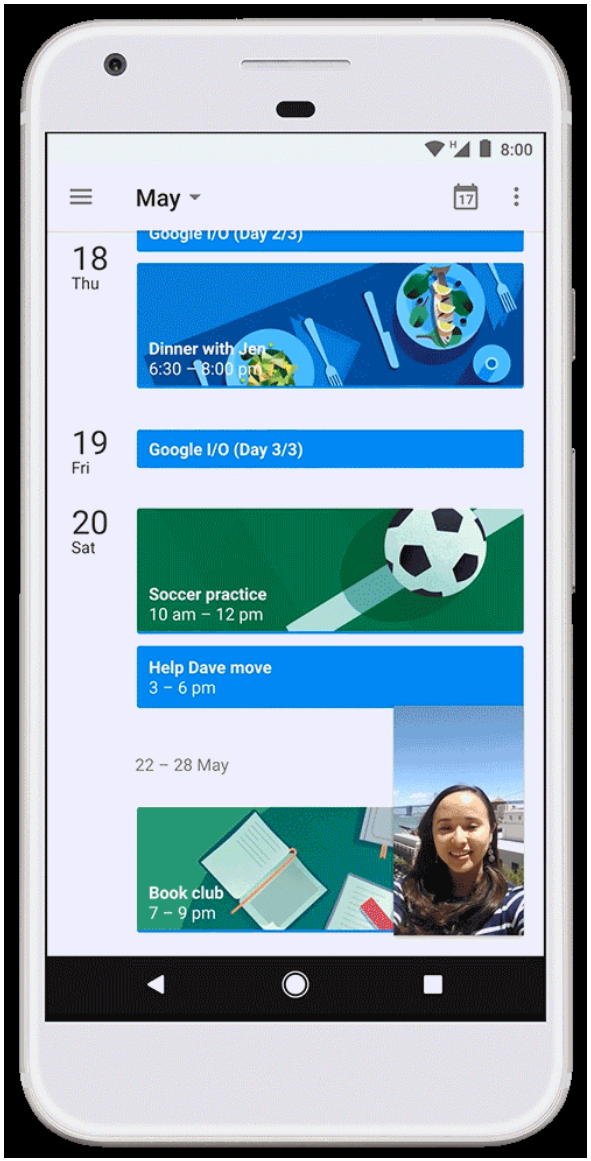
- Android 1.0 (septembar 2008), API 1
- Android 1.5 Cupcake (april 2009), API 3
- Android 1.6 Donut (septembar 2009), API 4
- Android 2.0 Eclair (oktobar 2009), API 5
- Android 2.2-2.2.3 Froyo (maj 2010), API 8
- Android 2.3-2.3.2 Gingerbread (decembar 2010), API 9
- Android 3.0 Honeycomb (februar 2011), API 11
- Android 4.0-4.0.2 Ice Cream Sandwich (oktobar 2011), API 14
- Android 4.1-4.1.2 Jelly Bean (jul 2012), API 16
- Android 4.4-4.4.4 KitKat (oktobar 2013), API 19
- Android 5.0-5.0.2 Lillipop (novembar 2014), API 21
- Android 6.0-6.0.1 Marshmallow (oktobar 2015), API 23
- Android 7.0-7.1.1 Nougat (avgust 2016), API 24 i 25
- Android 8.0-8.1 Oreo (avgust 2017), API 26 i 27
- Android 9.0 Pie (avgust 2018), API 28

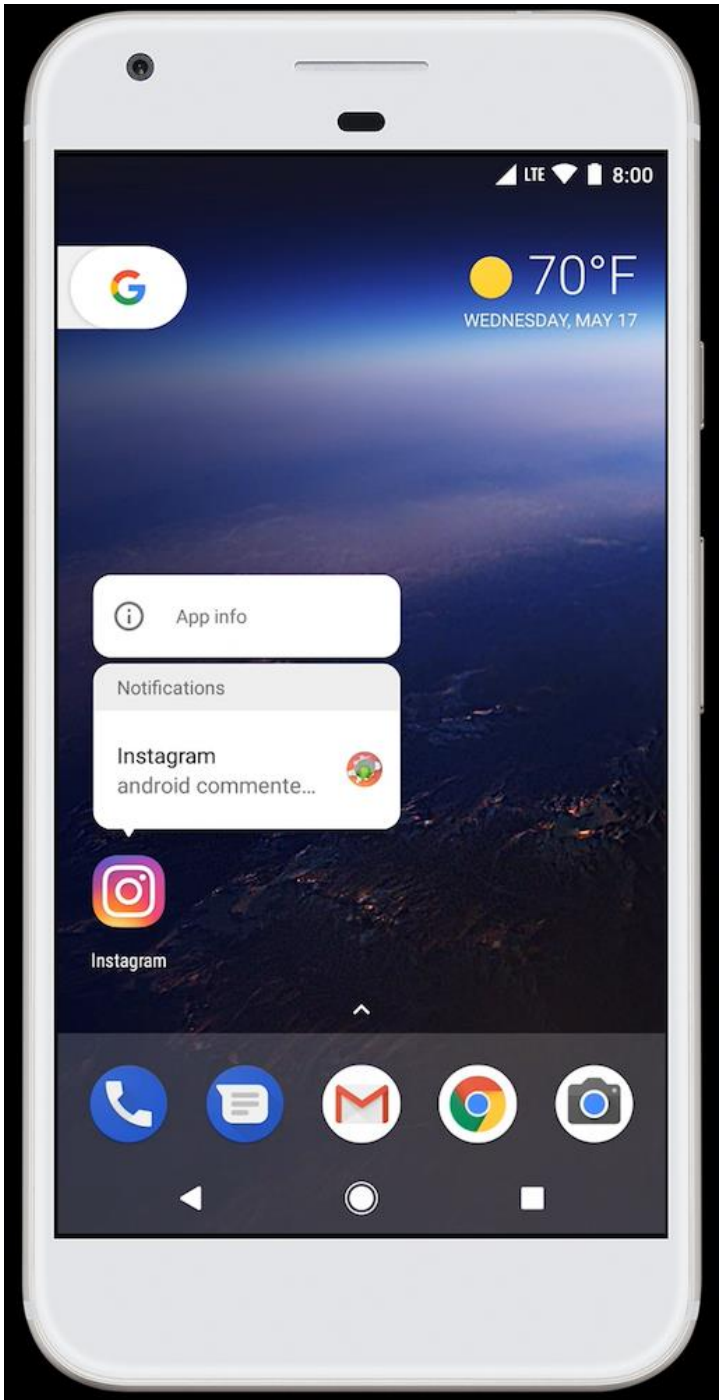
Android 8.0 Oreo karakteristike

Ova verzija Androida uvodi sledeća poboljšanja:

- Novi Picture-in-Picture mod ili ”Slika-u-Slici” (poseban mod koji je podvrsta ”multi-window” moda koji se uglavnom koristi za video playback)
- Android O može ”naučiti” o tome što korisnik preferira i koje su korisnikove preference korišćenja softvera
- Napredne opcije uklanjanja keš memorije i oslobađanja dodatnog prostora za aplikacije
- Wi-Fi Aware ili *Neighbor Awareness Networking* (NAN) ili svesnost o susednom korišćenju mreže; Uređaji koji imaju potreban hardver će sami moći otkriti i u isto vreme se spojiti na vlastite Wi-Fi Aware mreže
- Napredni rad sa aplikacijama i procesima u pozadini i napredni način uštede baterije
- Korišćenje Bluetooth 5 hardvera i naprednije upravljanje zvukom

Android Oreo





Karakteristike Android OS

Ne postoje posebno određene hardversko-softverske konfiguracije za Android ali moraju biti ispoštovani sledeći zahtevi:

1) skladištenje podataka putem **SQLite** relacione baze podataka
2) pristupanje mobilnim mrežama putem svih poznatih mobilnih standarda (GSM, GPRS, EDGE, UMTS, LTE) kao i umrežavanje putem Bluetooth, NFC, WiFi, WiMax i drugih poznatih standarda
3) slanje tekstualnih SMS i multimedijalnih MMS poruka
4) web čitač zasnovan na V8 JavaScript okruženju koje koristi Chrome
5) hardverska podrška za akcelerometar, GPS, kameru i razne senzore
6) podrška za ekrane osetljive na dodir
7) multi-tasking rad izvršavanja više zadataka istovremeno
8) Identifikacija otiskom prsta ili skeniranjem oka ili lica
9) deljenje Internet konekcije itd...

Android telefoni

Google Pixel 2

LG V30

Android wear

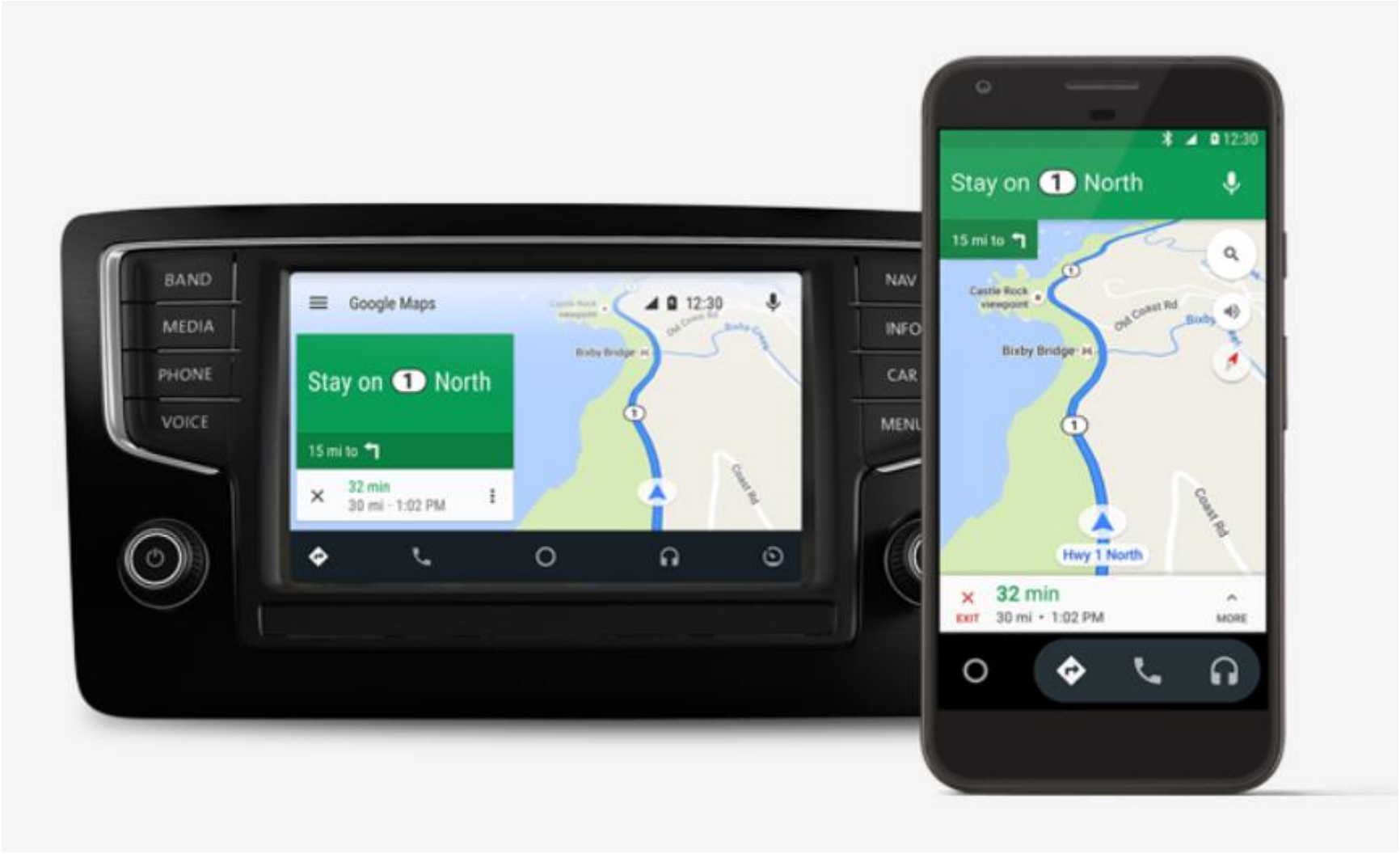
Android tablets

Samsung Galaxy Tab S2 8.0

Sony Xperia Z4 Tablet

Android TV and gaming

Android vehicles (automobil)



Verzije Androida i API Level

Svaka verzija Androida ima i broj koji se zove API level

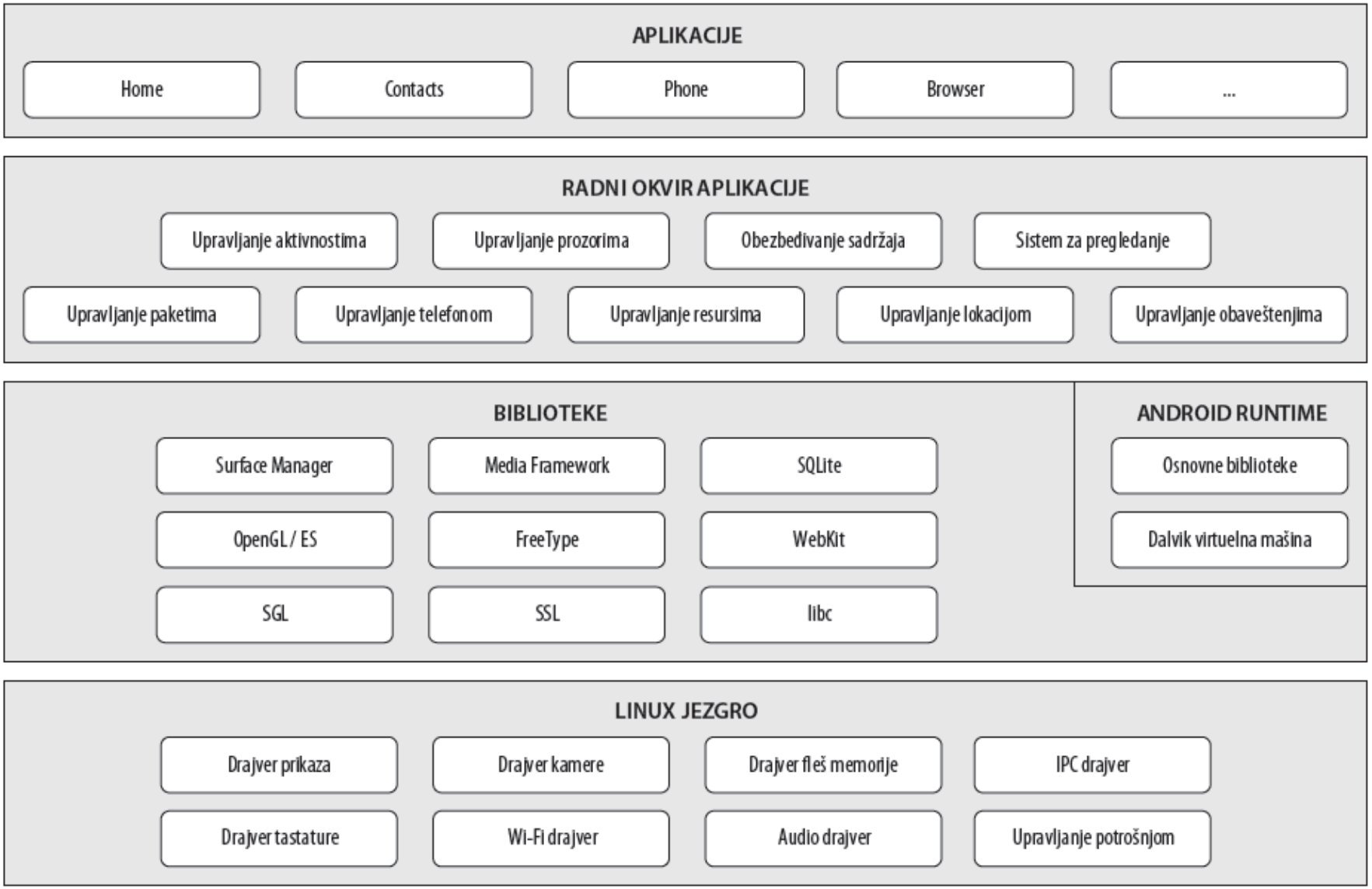
- API level označava verziju Android API-a, odnosno verziju biblioteka i klasa koji su dostupni programerima
- Više verzija Android-a može imati isti API level, na primer Marshmallow ima 6.0 - 6.0.1 imaju isti API level 23
- to znači da verzije 6.0. do 6.0.1 nisu imale promene u API-u koje su vidljive programerima, već samo unapređenja i ispravke bug-ova

Primer verzije API-a:

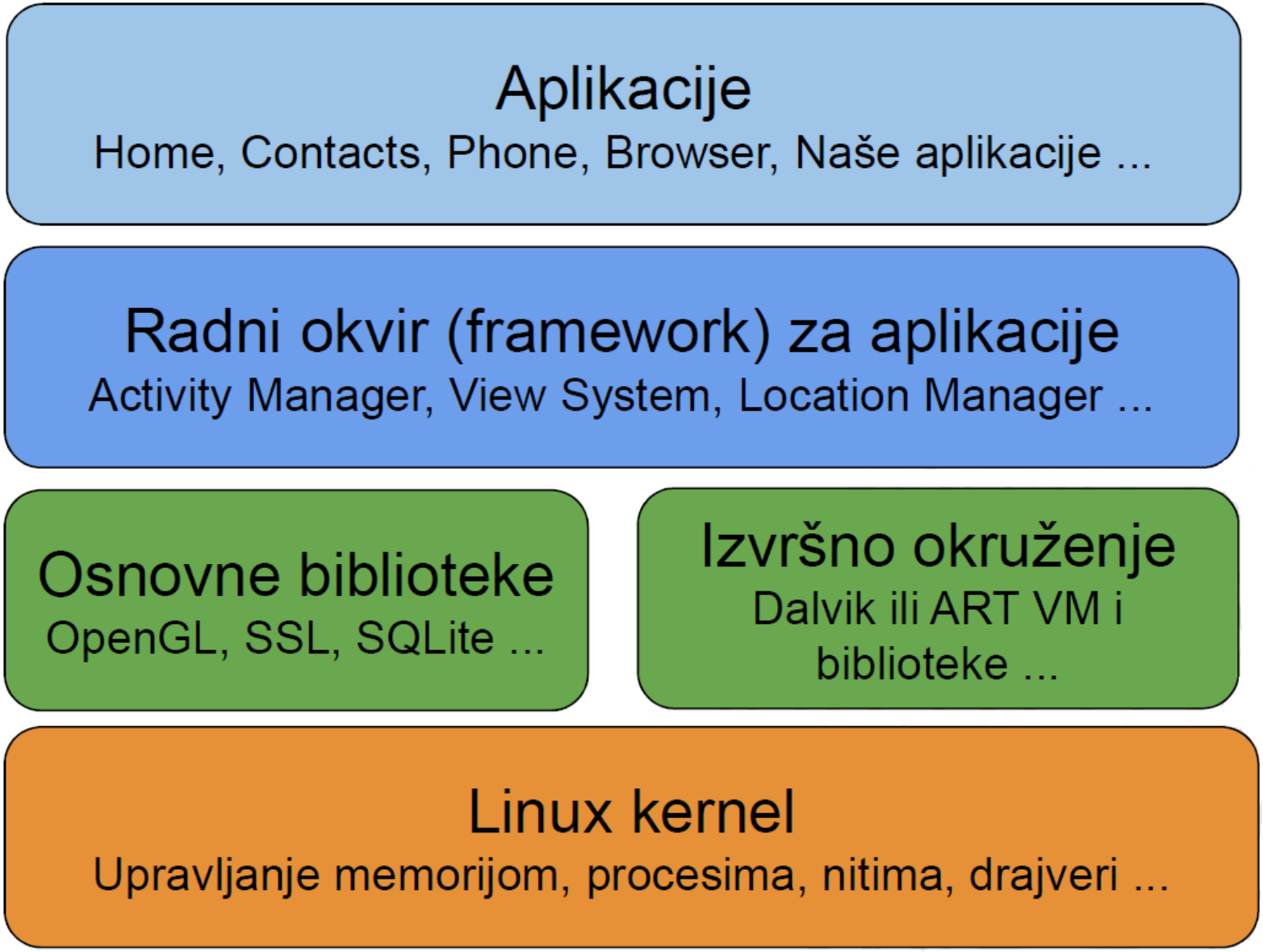
```
//API level 1
class GPS{
    public long geoSirina(){ ... };
    public long geoDuzina(){ ... };
}
```

```
//API level 2
class GPS{
    public long geoSirina(){ ... };
    public long geoDuzina(){ ... };
    public long nadmorskaVisina() { ... };
}
```

Android arhitektura



Android arhitektura – softverski stek



Android – arhitektura

Android OS je podeljen na pet sekcija i četiri osnovna sloja:

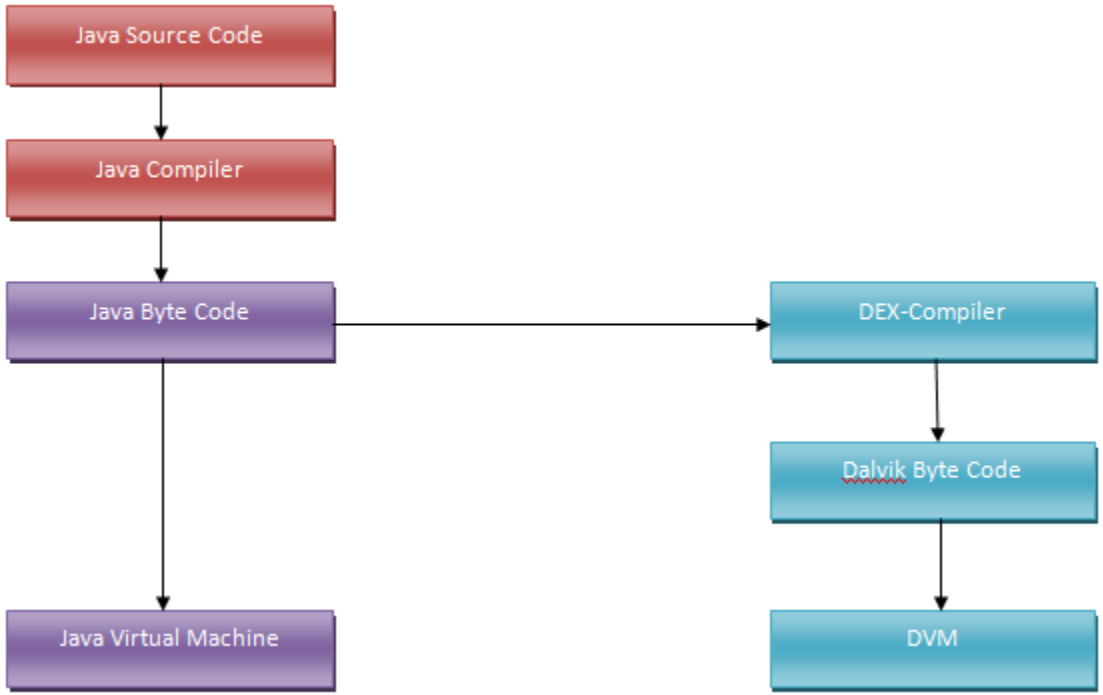
- **Linux jezgro** – To je jezgro na kome je Android zasnovan i ovaj sloj sadrži sve drajvere hardverskih komponenti od kojih se sastoji Android uređaj, definisanih na niskom nivou
- **Biblioteke** – Sadrže sav kod koji obezbeđuje osnovne funkcije Android operativnog sistema (npr. SQLite biblioteka obezbeđuje podršku za korišćenje baza podataka, tako da aplikacija može da je koristi za skladištenje podataka, WebKit biblioteka obezbeđuje funkcije koje se odnose na korišćenje web pretraživača...) Ove biblioteke se jednim imenom zovu Android API i obuhvataju različite nivoe, od onih za definisanje izgleda i funkcionalnosti aplikacija do onih za komunikaciju sa drugim aplikacijama, servisima i hardverom

Android – arhitektura

- **Android runtime okruženje** – Na istom nivou kao i biblioteke, Android okruženje obezbeđuje skup osnovnih biblioteka (arhiva) koje omogućavaju programerima da pišu Android aplikacije korišćenjem Java programskog jezika; Android okruženje sadrži i Dalvik virtuelnu mašinu (DVM) koja omogućava svakoj Android aplikaciji da se izvršava u sopstvenom procesu, sa sopstvenom instancom DVM (Android aplikacije se prevode u Dalvik izvršne datoteke **.dex**). Dalvik je virtuelna mašina, projektovana specijalno za Android i optimizovana za mobilne uređaje koji koriste baterije pri radu i imaju ograničene RAM i CPU resurse
- **Radni okvir aplikacija (Framework)** – Omogućava se korišćenje različitih mogućnosti Android operativnog sistema, tako da programeri mogu da ih koriste u svojim aplikacijama

Android – arhitektura

- **Nivo aplikacija** – Na ovom nivou nalaze se aplikacije koje se isporučuju sa Android uređajima (kao što su Phone, Contacts, Browser i slične), kao i aplikacije koje se preuzimaju i instaliraju korišćenjem Google Play-a. Sve aplikacije koje programeri samostalno kreiraju ili implementiraju nalaze se na ovom nivou



Softver za razvoj Android aplikacija

- Za razvoj Android aplikacija može se koristiti Apple, Windows PC ili Linux OS
- Programski jezik JAVA je osnova razvoja Android aplikacija
- Softver neophodan za kreiranje Android aplikacija je besplatan i može se preuzeti sa Interneta
- Potrebni softverski alati su:
  - Java JDK SE
  - Android SDK
  - Eclipse IDE ili Android Studio IDE
  - ADT (Android Development Tools)
  - AVD (Android Virtual Devices)

Developers

Java JDK

- Android SDK koristi Java SE Development Kit (JDK)
- Ukoliko na računaru nije instaliran JDK, treba ga preuzeti sa stranice koja se nalazi na adresi:

https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

i zatim ga instalirati pre instaliranja Android SDK softverskog paketa

Java SE Development Kit 8u191

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

○ **Accept License Agreement** ◉ **Decline License Agreement**

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	72.97 MB	jdk-8u191-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	69.92 MB	jdk-8u191-linux-arm64-vfp-hflt.tar.gz
Linux x86	170.89 MB	jdk-8u191-linux-i586.rpm
Linux x86	185.69 MB	jdk-8u191-linux-i586.tar.gz
Linux x64	167.99 MB	jdk-8u191-linux-x64.rpm
Linux x64	182.87 MB	jdk-8u191-linux-x64.tar.gz
Mac OS X x64	245.92 MB	jdk-8u191-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	133.04 MB	jdk-8u191-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	94.28 MB	jdk-8u191-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	134.04 MB	jdk-8u191-solaris-x64.tar.Z
Solaris x64	92.13 MB	jdk-8u191-solaris-x64.tar.gz
Windows x86	197.34 MB	jdk-8u191-windows-i586.exe
Windows x64	207.22 MB	jdk-8u191-windows-x64.exe

Java JDK

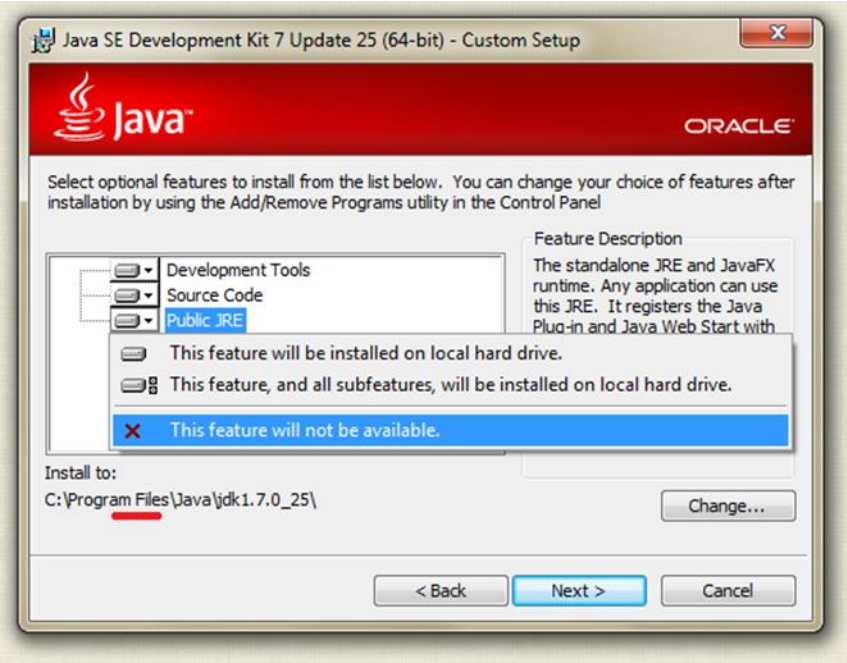
- JAVA Development Kit (JDK) predstavlja implementaciju JAVA platforme predstavljene od strane kompanije Oracle u formi paketa binarnih datoteka namenjenih programerima za razvoj JAVA softverskih rešenja na različitim hardversko-softverskim platformama.
- JDK obuhvata i JAVA virtuelnu mašinu (JVM) i sve prateće resurse kojima je omogućeno razvijanje i izvršavanje JAVA softvera
- Od 2007. godine JAVA se distribuira kroz GNU General Public Licence (GPL), što znači da je omogućeno potpuno besplatno nabavljanje JAVA tehnologije i odgovarajućih razvojnih alata
- Od 1995. godine, kada je predstavljena prva verzija programskog jezika JAVA, objavljeno je osam generacija ovog programskog jezika zajedno sa tekućim ispravkama

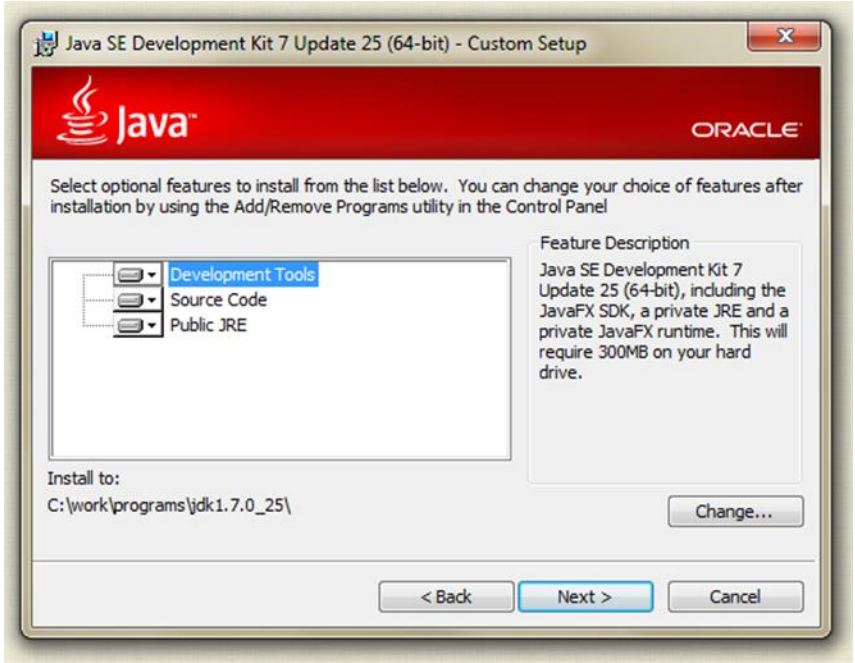
Java JDK

- Neophodno je izabrati i instalirati verziju JDK koja odgovara verziji OS instaliranog na računaru
- Java JDK i njegova dokumentacija su potpuno nezavisni i instaliraju se posebno
- Komplet JDK za operativni sistem Windows postoji u dve verzije – kao web instalacija u kojoj se datoteke preuzimaju redom i kao kompletno preuzimanje .exe datoteke čijim se aktiviranjem pokreće celokupna instalacija

Java JDK

- Neophodno je izabrati lokaciju i opcione alate za instalaciju
- Tokom procesa instalacije JDK biće neophodno izvršiti određena podešavanja, poput izbora lokacije na kojoj će biti JDK snimljen, kao i izbora opcionih JDK alata

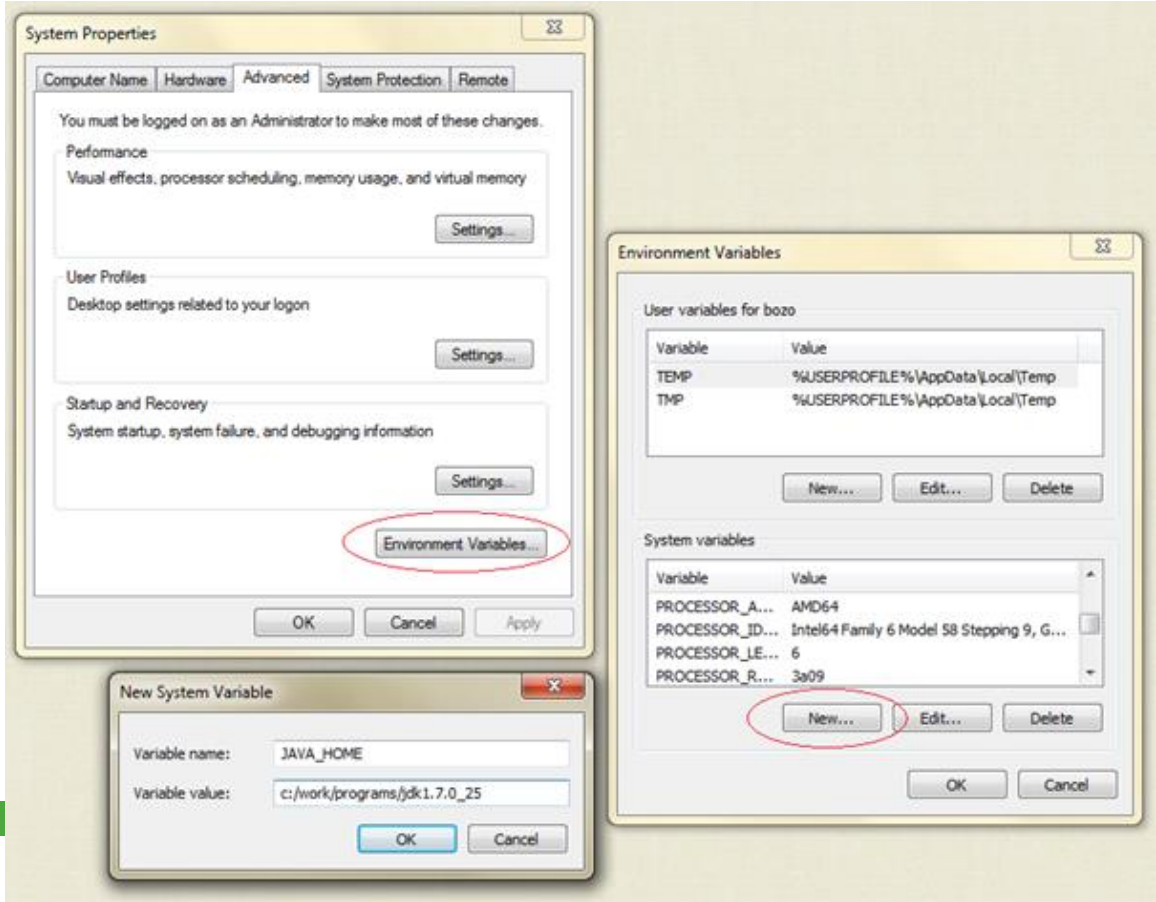




Java JDK (važan deo)!!

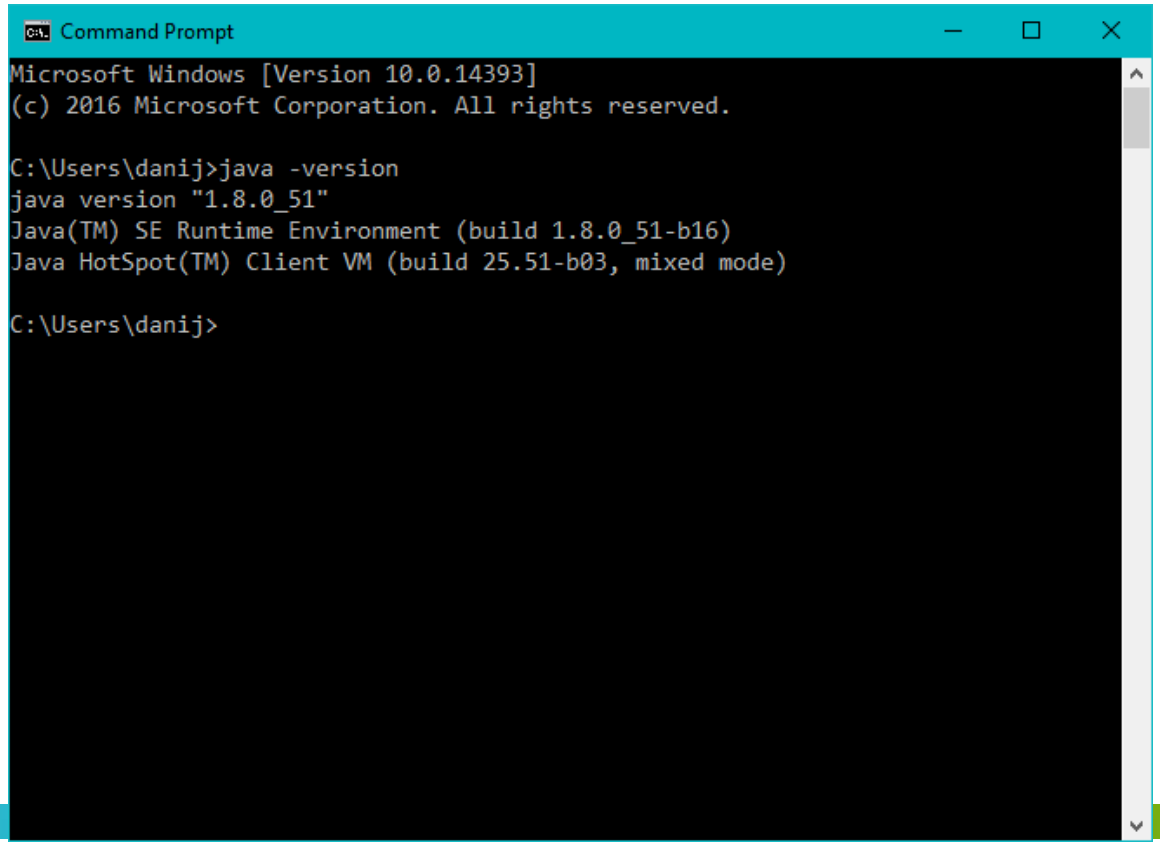
- Neophodno je JAVU obezbediti dostupnom iz bilo kojeg direktorijuma na računaru; To se postiže podešavanjem sistemskih varijabli. Klikom na **My Computer**>**Properties**>Advanced System Settings>Enviroment Variables, otvara se prozor u kojem će biti izvršena navedena podešavanja

Definiše se jedna sistemska promenljiva po imenu **JAVA_HOME** čija će vrednost biti putanja do JDK foldera. Nakon toga iz liste postojećih varijabli, bira se varijabla PATH u okviru koje je neophodno dodati sledeći string ;%JAVA_HOME%/bin kojim je određena putanja do JAVA prevodioca



Java JDK (važan deo)!!

- U MS DOS-u se proverava verzija instalacije JAVA paketa
- Otvaranjem DOS Command Prompta i kucanjem instrukcije **java –version** vrši se provera; Ukoliko se na ekranu ispiše verzija, Java je uspešno instalirana; u suprotnom biće ispisan komentar *'java' is not recognized as an internal or external command, operable program or batch file.*



Android SDK

- Zatim je neophodno preuzeti je Android SDK (*Software Development Kit*); trenutna verzija je SDK v26.1.1 (sep. 2017)
- SDK sadrži debager, biblioteke, emulator, dokumentaciju, primere koda i uputstva za
- Android SDK može se preuzeti kao .zip datoteka sa stranice koja se nalazi na adresi:

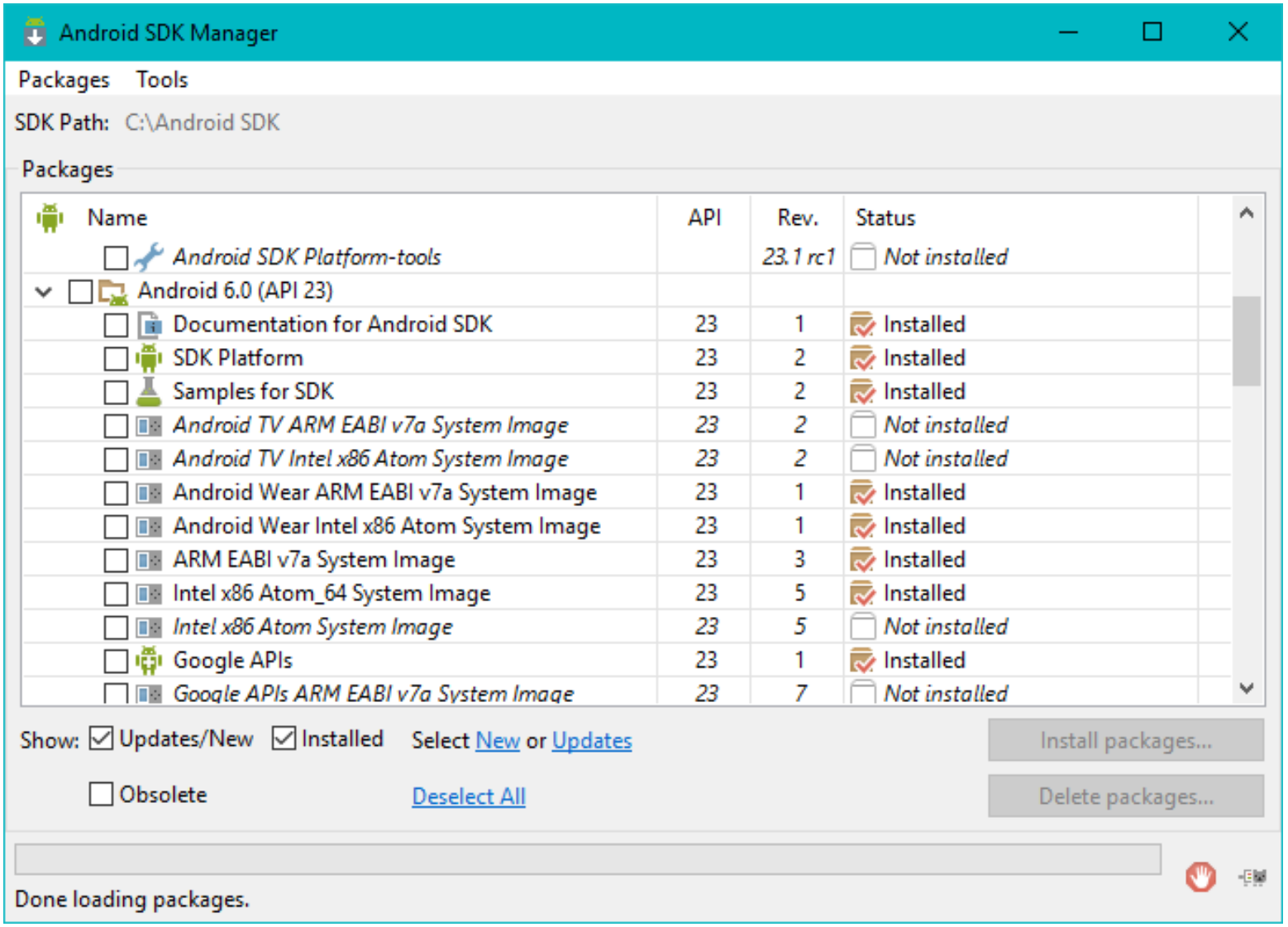
https://developer.android.com

Platform	SDK tools package	Size
Windows	sdk-tools-windows-4333796.zip	149 MB
Mac	sdk-tools-darwin-4333796.zip	98 MB
Linux	sdk-tools-linux-4333796.zip	147 MB

Android SDK

- Android SDK biće potreban kod instalacije ADT za Eclipse IDE
- Android SDK Manager upravlja različitim verzijama Android SDK koje su trenutno instalirane na računaru. Nakon pokretanja SDK Managera, prikazuje se lista stavki
- Selektovati relevantne alate, dokumentaciju i platforme koje želimo koristiti u projektu
- Nakon selektovanja stavki koje želimo, kliknuti na **Install** taster da bismo ih preuzeli. Pošto je neophodno određeno vreme da se sve selektovane stavke preuzmu na računar sa Google servera, dobra ideja je da se preuzme samo ono što je zaista neophodno da bi se započeo rad, a ostatak preuzeti kasnije

Android SDK Manager



Android SDK - konfigurisanje

- Svaka verzija Android OS identifikovana je API brojem nivoa (npr. Android 2.3.3 je nivo 10 (API 10), dok Android 6.0 predstavlja nivo 23 (API 23) itd).
- Za svaki nivo postoje dve platforme. Na primer, nivo 14 pruža sledeće:

SDK platformu

Google API interfejse kompanije „Google“

- Ključna razlika između ova dve platforme ogleda se u činjenici da Google API platforma sadrži dodatne API interfejse koje je obezbedio Google (npr. Google Maps biblioteka). Zbog toga, ukoliko aplikacija koju kreiramo zahteva Google Maps, neophodno je da kreirate AVD korišćenjem Google API platforme

Eclipse IDE for Java Developers 2018-09

Eclipse IDE for Java Developers

Package Description

The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Mylyn, Maven and Gradle integration

This package includes:

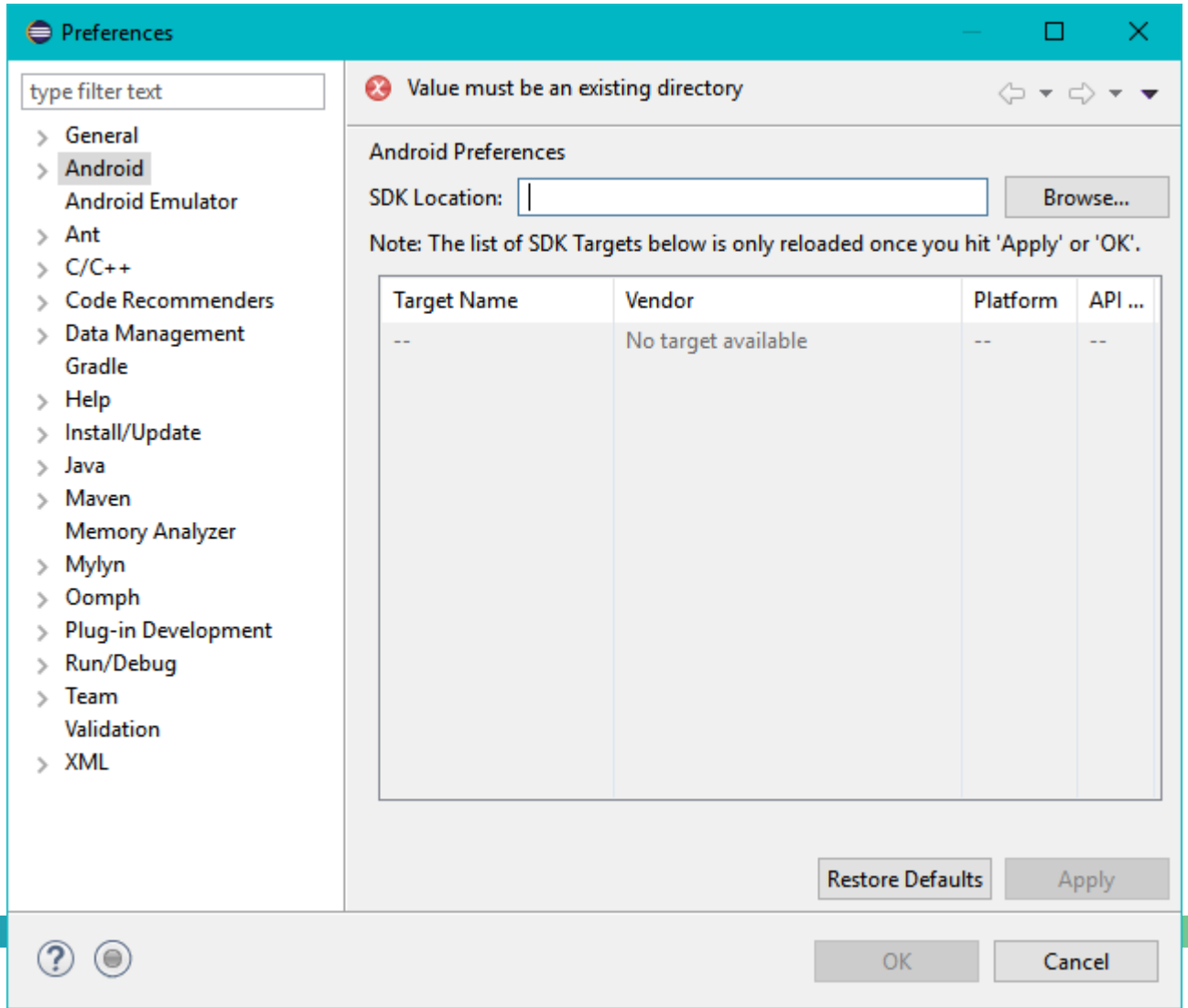
- Git integration for Eclipse
- Eclipse Java Development Tools
- Maven Integration for Eclipse
- Mylyn Task List
- Code Recommenders Tools for Java Developers
- Eclipse XML Editors and Tools

Eclipse IDE

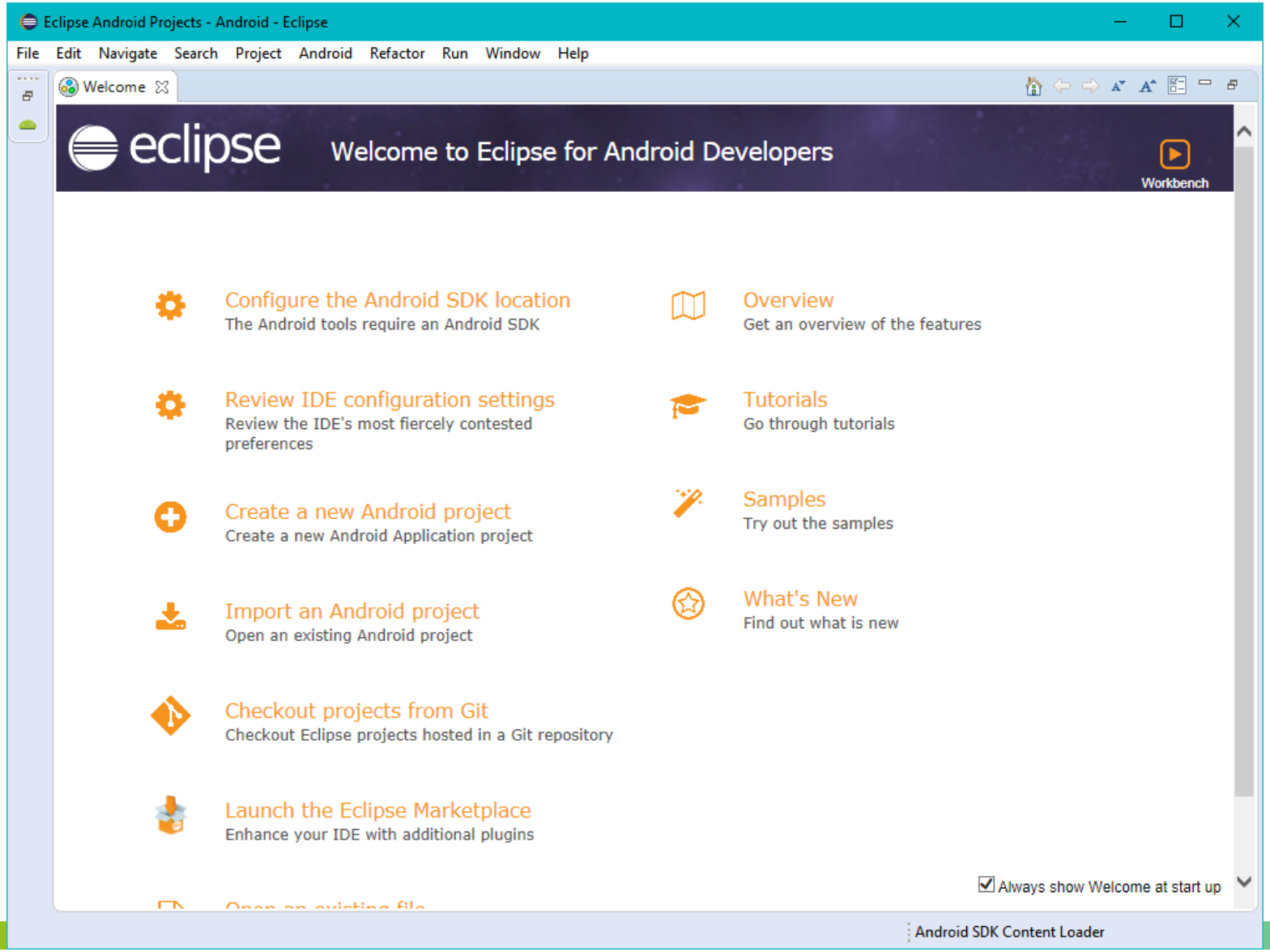
- Kada je reč o Android aplikacijama, preporučuje se korišćenje **Eclipse** integrisanog razvojnog okruženja IDE (*Integrated Development Environment*), koje se može koristiti za razvoj programa u različitim programskim jezicima i koje predstavlja sistem proširiv različitim dodacima.
- Eclipse se može koristiti za razvoj različitih tipova aplikacija korišćenjem programskih jezika, kao što su Java, Ada, C, C++, COBOL, Python i drugi
- Za potrebe razvoja Android aplikacija treba preuzeti Eclipse IDE for Java Developers sa zvanične internet adrese **www.eclipse.org/downloads/packages**

Eclipse proces instalacije

Potrebno je uneti putanju do SDK koji je prethodno preuzet



Eclipse IDE


Eclipse Android Projects - Android - Eclipse
File Edit Navigate Search Project Android Refactor Run Window Help
Welcome
eclipse
Welcome to Eclipse for Android Developers
Workbench
Configure the Android SDK location
The Android tools require an Android SDK
Review IDE configuration settings
Review the IDE's most fiercely contested preferences
Create a new Android project
Create a new Android Application project
Import an Android project
Open an existing Android project
Checkout projects from Git
Checkout Eclipse projects hosted in a Git repository
Launch the Eclipse Marketplace
Enhance your IDE with additional plugins
Overview
Get an overview of the features
Tutorials
Go through tutorials
Samples
Try out the samples
What's New
Find out what is new
Always show Welcome at start up
Android SDK Content Loader

ADT (Android Development Tools)

- Nakon startovanja Eclipse integrisanog razvojnog okruženja, selektuje se Help → Install New Software kako bi se instalirao ADT dodatak za Eclipse; zatim se unosi adresa

https://dl-ssl.google.com/android/eclipse/

- ADT je proširenje za Eclipse koje podržava kreiranje i identifikovanje grešaka u Android aplikacijama. Korišćenje ADT u Eclipse okruženju omogućava sledeće:

  - kreiranje novih Android aplikacija
  - pristup alatima koji omogućavaju korišćenje Android emulatora i uređaja
  - prevođenje i identifikacija greške u Android aplikacijama
  - eksportovanje Android aplikacije u Android pakete APK (Android Package)
  - kreiranje digitalnih sertifikate za potpisivanje koda APK paketa

ADT (Android Development Tools)

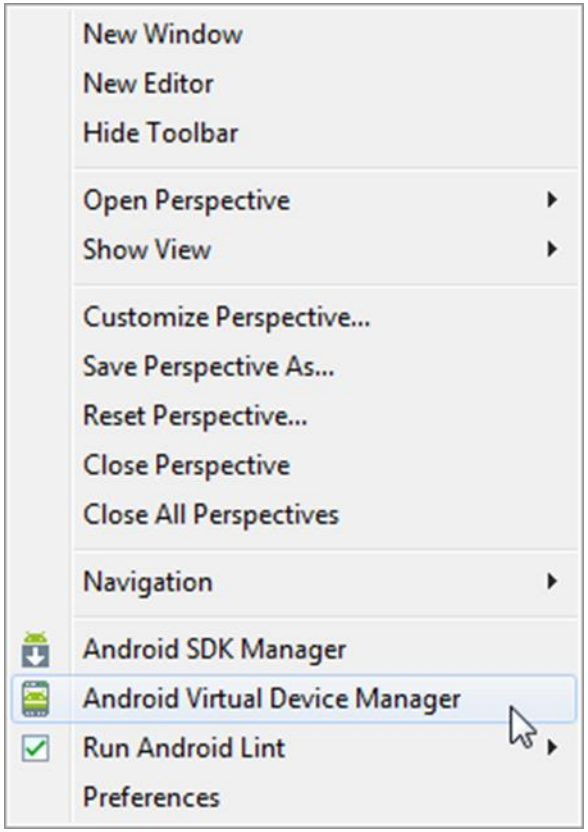
- Nakon instalacije ADT softvera za Eclipse, potrebno je konfigurisati Android SDK tj. navesti putanju do prethodno preuzetog SDK direktorijuma
- U "Welcome to Android Development" prozoru izabrati **Use existing SDKs**
- Uneti putanju Android SDK direktorijuma koji je preuzet i raspakovan; Kliknuti na **Next**.
- Sada je Eclipse IDE okruženje spremno za razvoj Android aplikacija, ali je potrebno dodati poslednje SDK platformske alate i Android API za aplikaciju

AVD (Android Virtual Devices)

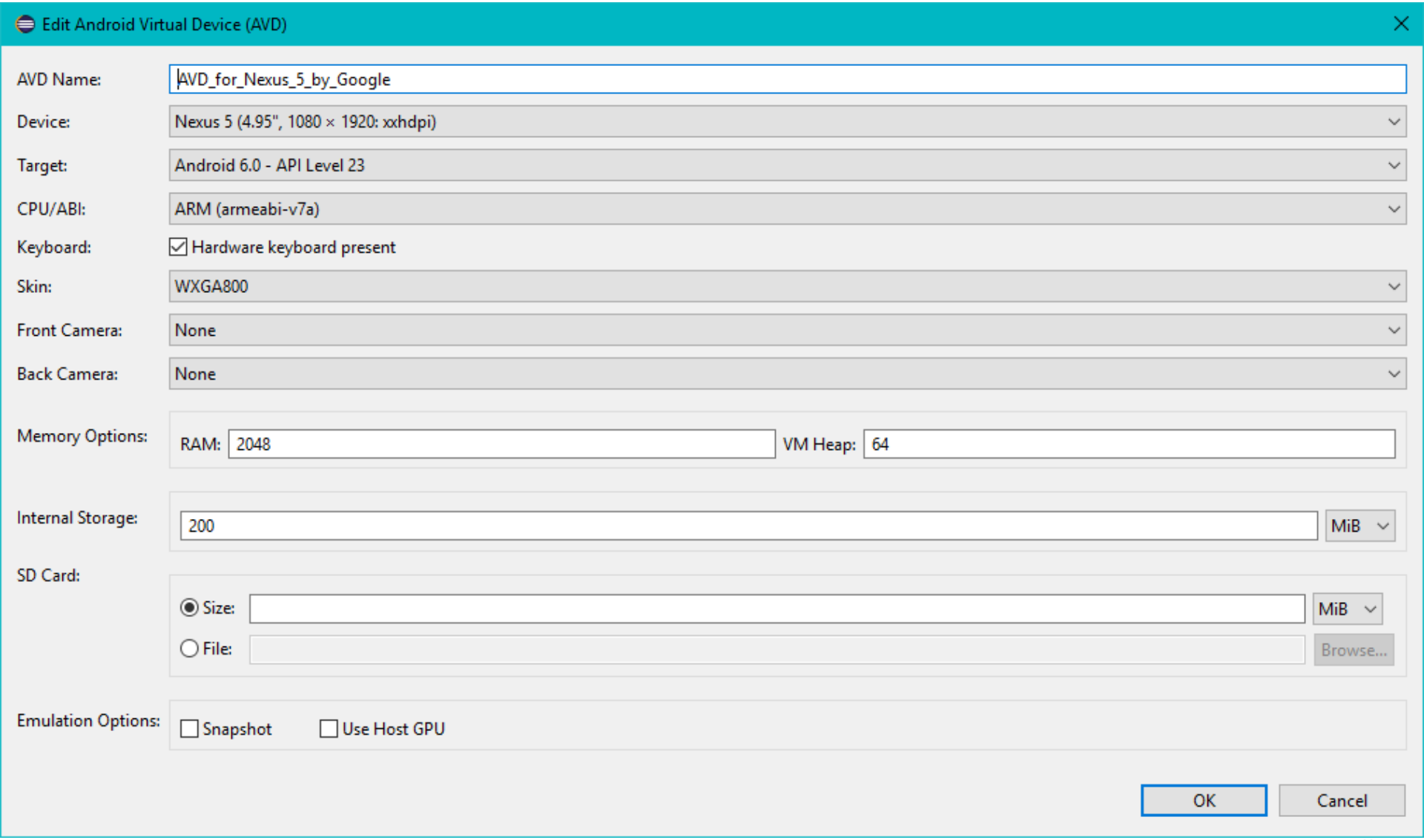
- Android virtuelni uređaj AVD se koristi prilikom testiranja aplikacija
- AVD je instanca emulatora koja omogućava modelovanje realnih uređaja; svaki AVD uređaj sastoji se od hardverskog profila, procedure za mapiranje u sistemsku sliku i emuliranog skladišta, kao što je npr Secure Digital (SD) kartica.
- Može se kreirati neograničeni broj AVD uređaja kako bi se aplikacije testirale na različitim konfiguracijama. Ovo testiranje je veoma značajno zbog provere načina funkcionisanja aplikacije kada se ona izvršava na različitim uređajima koji poseduju različite funkcije
- Android aplikacije se mogu testirati direktno na Android telefonu (ili tabletu), ali osnovna prednost korišcenja AVD je što je moguće raditi testove da bi se proverilo kako se aplikacija ponaša na uređajima sa različitim sposobnostima i na raznim verzijama Androida

AVD

U Eclipse IDE razvojnom okruženju bira se opcija AVD Manager iz menija Window



AVD (primer emulatora za Nexus 5)




Android
Studio

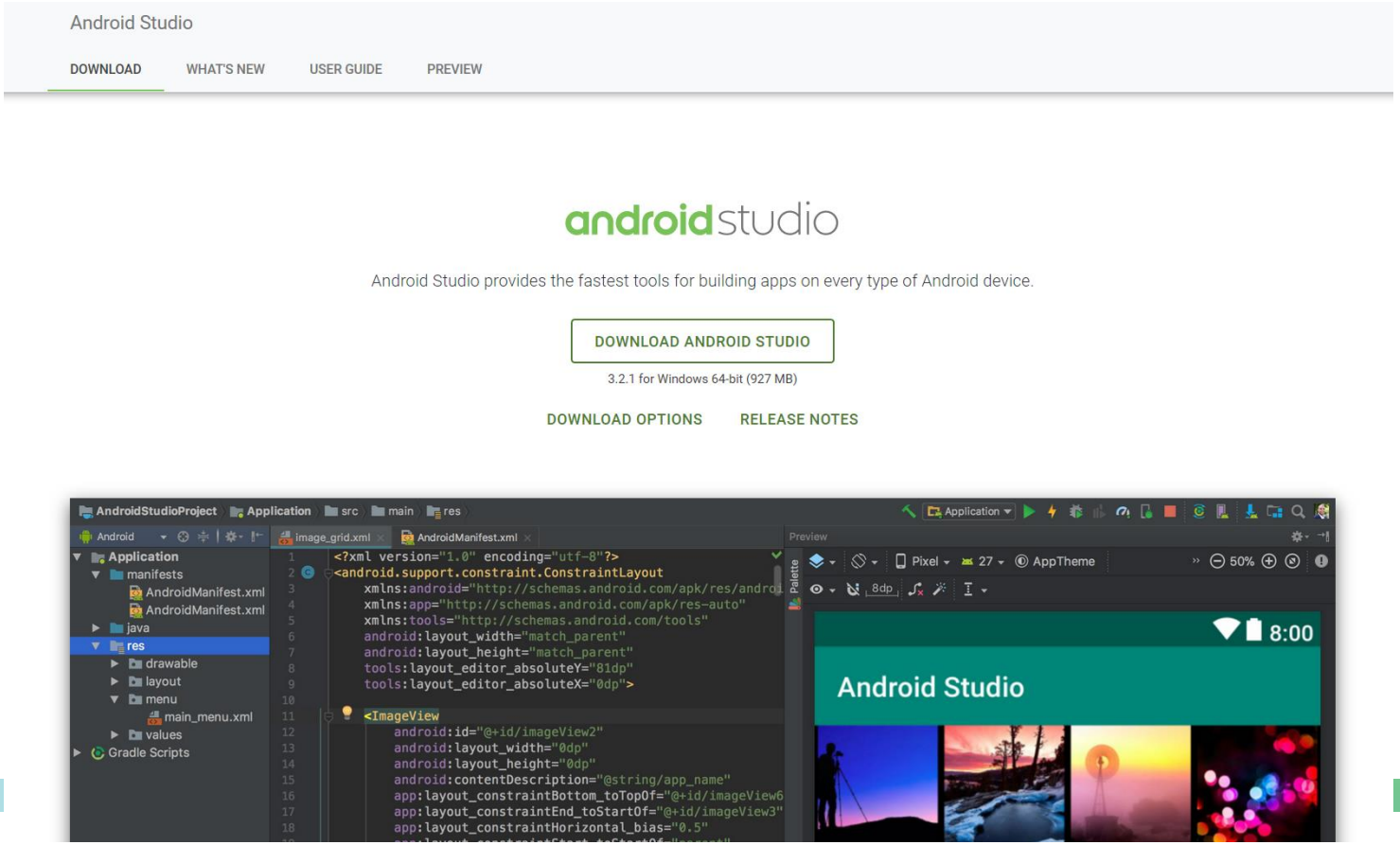
Android Studio

- Za razvoj Android aplikacija Google preporučuje da se koristi Android Studio koji u sebi već sadrži osnovne pakete za razvoj aplikacija, a lako se može dopuniti drugim paketima.
- Android Studio je podržan za Linux, Windows i Mac OS X operativne sisteme
- Zvanično IDE okruženje od 2016
- Zasniva se na IntelliJ IDEA okruženju
- Proširen sa podrškom za Android
- Google ga zvanično podržava
- Po funkcionalnosti odgovara Eclipse + ADT paa čak i naprednije
- Trenutna verzija je 3.2.1

Android Studio 3

- Instalacija Android Studio softvera

https://developer.android.com/studio/index.html



Android Studio sistemski zahtevi

Za Windows OS:

- Microsoft® Windows® 7/8/10 (32- or 64-bit)
- 4 GB RAM minimum, 8 GB RAM preporučeno, plus 1 GB za Android Emulator
- 4 GB slobodnog prostora na disku (minimum) i 4 GB preporučeno(500 MB za IDE + 1.5 GB za Android SDK i emulator system image)
- 1280 x 800 minimum rezolucija monitora
- Za ubrzani emulator: 64-bitni Windows OS i Intel® procesor sa podrškom za Intel® VT-x, Intel® EM64T (Intel® 64), i Execute Disable (XD) Bit funkcionalnost

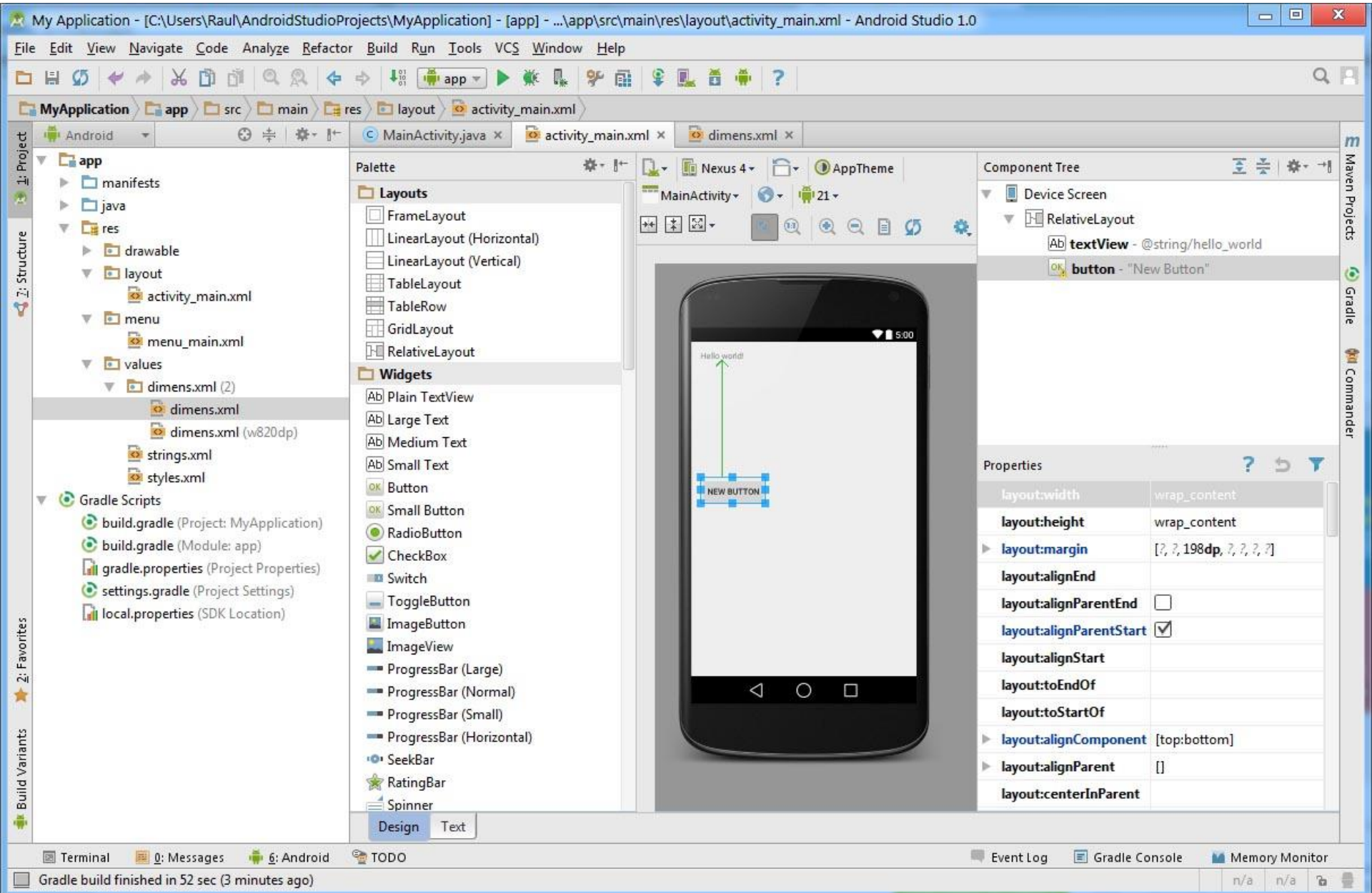
Android Studio GRADLE

- Kompajliranje je proces prevođenja programskog koda u objektni kod
- Kompajleri uz kompajliranje vrše i linkovanje, koje objektni kod pretvarau datoteku koja se može izvršiti u računaru
- **Build** projekta, uz kompajliranje i linkovanje, uključuje i dodatne radnje, npr. izradu installera. Build omogućuje da se izvrše testovi kompajliranog koda, određuje gde će se izvršni program nalaziti itd.
- Procesom build-ovanja moguće je stvarati statičke i dinamičke pomoćne biblioteke koje se kasnije mogu samostalno koristiti.
- Razlika među njima je u tome što kad statičku biblioteku unesemo u neki kod, ona se u celosti linkuje prilikom kompajliranja, dok se kod dinamičke datoteke linkuje samo deo koji će se koristiti u izvršnom programu

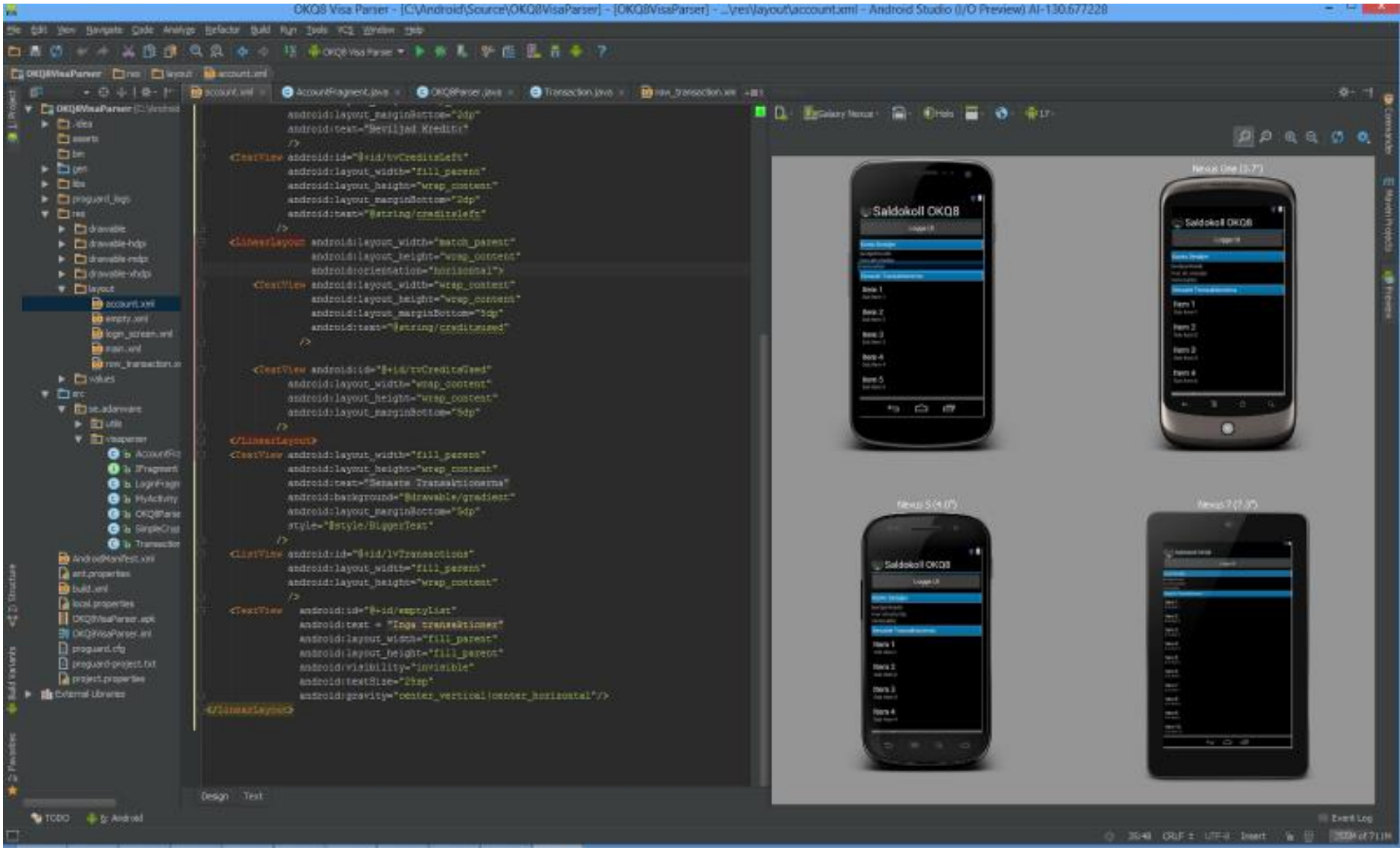
Android Studio GRADLE

- U slučaju razvoja Android aplikacija, build sistem služi da bi uzeo izvorne programske datoteke (.java ili .xml) i na njih primenio neke alate kompajliranja i linkovanja (npr **.java** datoteku pretvori u **.dex**) i grupiše sve te datoteke u jednu komprimovanu **.apk** datoteku s kojom Android sistem zna da radi
- Build sistem koji se koristi u Android Studio naziva se Gradle
- Gradle je razvijen na osnovu drugih operativnih sistema, integrisanjem njihovih najboljih karakteristika
- Gradle je baziran na JVM (*Java Virtual Machine*) build sistemu, što znači da se mogu pisati skripte za build u Java programskom jeziku
- Gradle vrši build projekata pomoću *Apache Ant* i *Apache Maven*, ali uvodi i Groovy DSL (*Domain-Specific Language*) koji omogućava skriptovanje buildova, što na kraju omogućava automatizaciju uploada beta .apk datoteka na TestFlight u svrhu testiranja

Android Studio



Android Studio



Android verzije - API Level

Verzije Androida - API level

- Prvi API level je bio 1, poslednji je 28 (Pie)

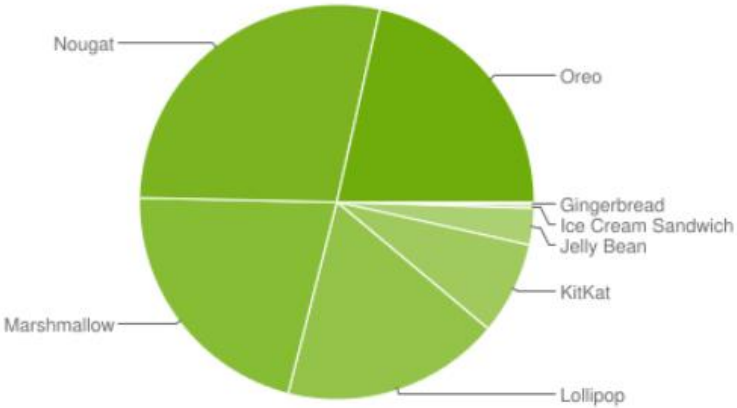
Zašto je programerima bitan API level?!

- Veći API level nudi više funkcionalnosti
- Kada se razvija aplikacija programer mora da izabere i deklariše API level u aplikaciji

- Izbor levela određuje kojim sve uređajima će aplikacija biti dostupna, sa kojim uređajima će biti kompatibilna

Android verzije - API Level zastupljenost

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.2%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.1%
4.2.x		17	1.5%
4.3		18	0.4%
4.4	KitKat	19	7.6%
5.0	Lollipop	21	3.5%
5.1		22	14.4%
6.0	Marshmallow	23	21.3%
7.0	Nougat	24	18.1%
7.1		25	10.1%
8.0	Oreo	26	14.0%
8.1		27	7.5%



Android verzije - API Level

- U *manifestu* (XML opisni fajl) aplikacije, programer može da definiše tri API levela:

1) minSdkVersion - minimalna verzija na kojoj aplikacija može da se izvršava, Android neće dozvoliti instalaciju na uređaju koji ima manju API verziju od ove; poželjno je da se definiše, u protivnom Android će pretpostaviti API verziju **1**

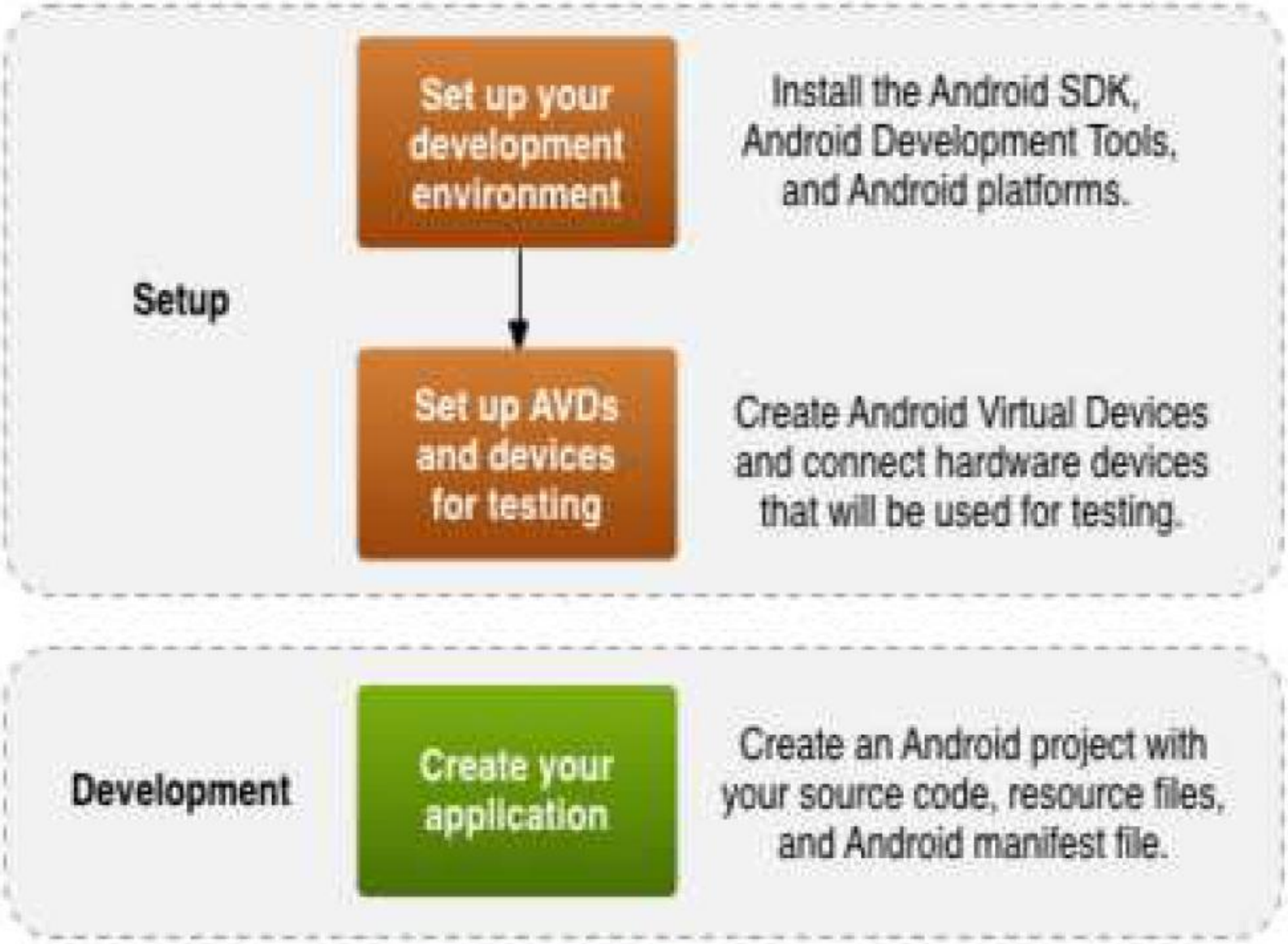
2) targetSdkVersion - API verzija na kojoj je aplikacija testirana, ako se ne definiše Android podrazumeva da je jednak minSdkVersion

3) maxSdkVersion - nije neophodno, često nije ni poželjno da se definiše, ograničava maksimalnu verziju API-a

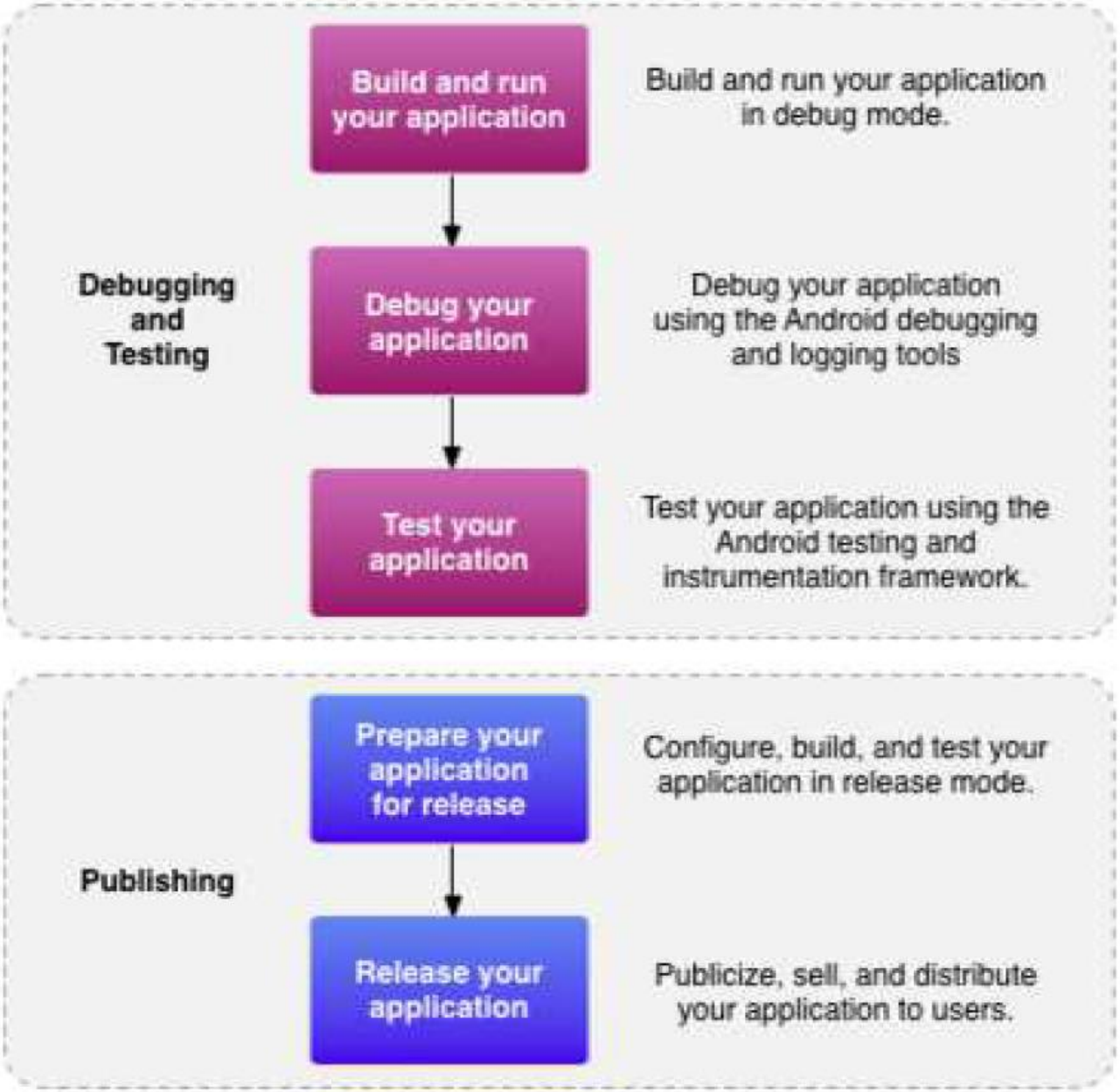
- Šta raditi kada želimo funkcionalnost iz novijeg API levela na uređaju sa starijom verzijom Androida - Android support biblioteke

  ○ Eksterne biblioteke koje se dodaju u projekat i obezbeđuju željeni API i ponašanje

Razvoj Android aplikacije – koraci 1 i 2



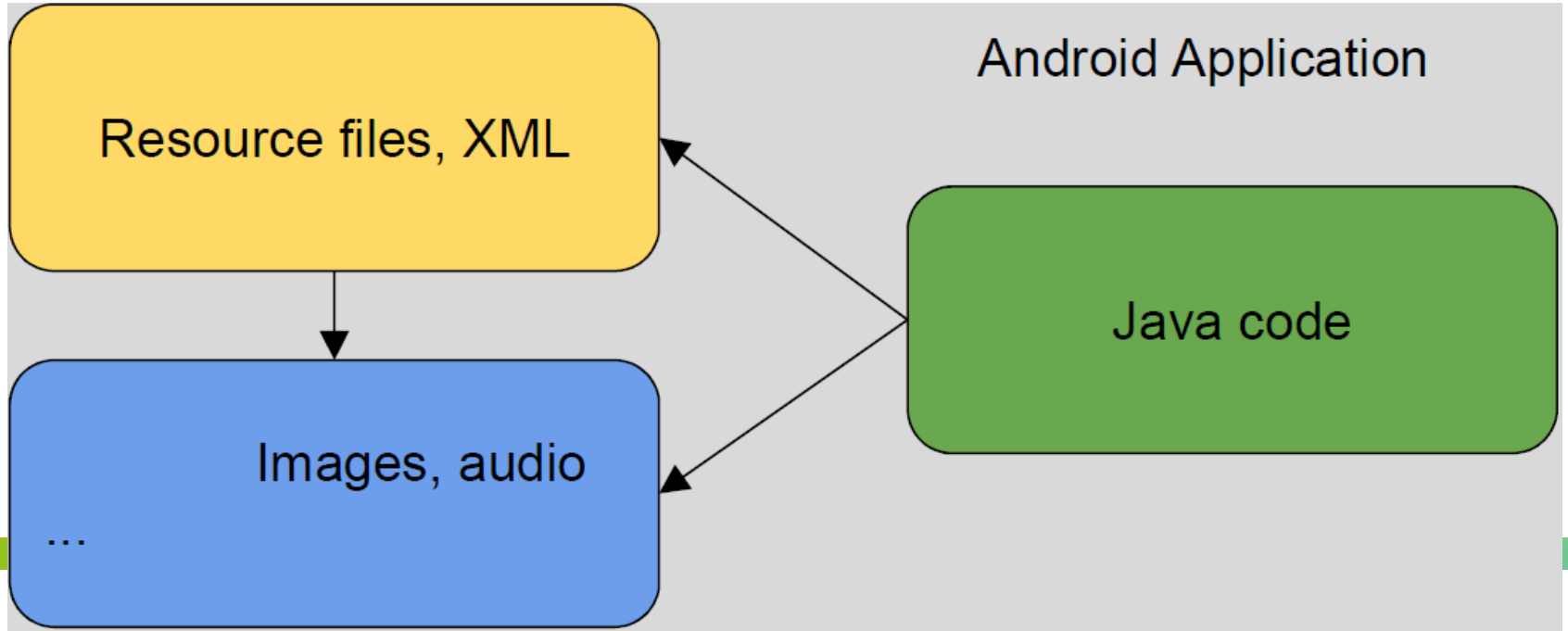
Razvoj Android aplikacije – koraci 3 i 4



Struktura Android aplikacije

Tipična Android aplikacija sastoji se od:

- programskog koda
- ***resource*** fajlova
  - ✓ XML fajlovi
  - ✓ slike i drugi dodatni fajlovi (audio zapisi itd.)



Struktura Android aplikacije

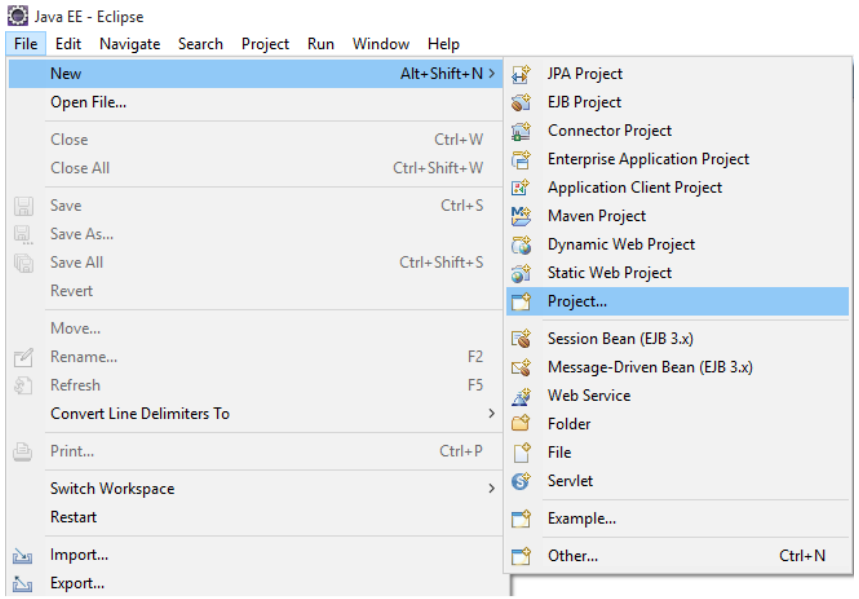
- Filozofija razvoja Android aplikacija jeste da se jasno razdvoje programski kod i resursni fajlovi
- ***Resource*** fajlovi su **XML** fajlovi
- U XML fajlovima se definišu: rasporedi, poruke, dimenzije, fontovi, teme, boje …
- Programski kod se piše u Java programskom jeziku korišćenjem Android API biblioteka, kao i nekih default Java biblioteka
- Android omogućava programeru da u programskom kodu pristupa elementima definisanim u *resource* fajlovima, da utiče na i menja prikaz, reaguje na događaje itd.

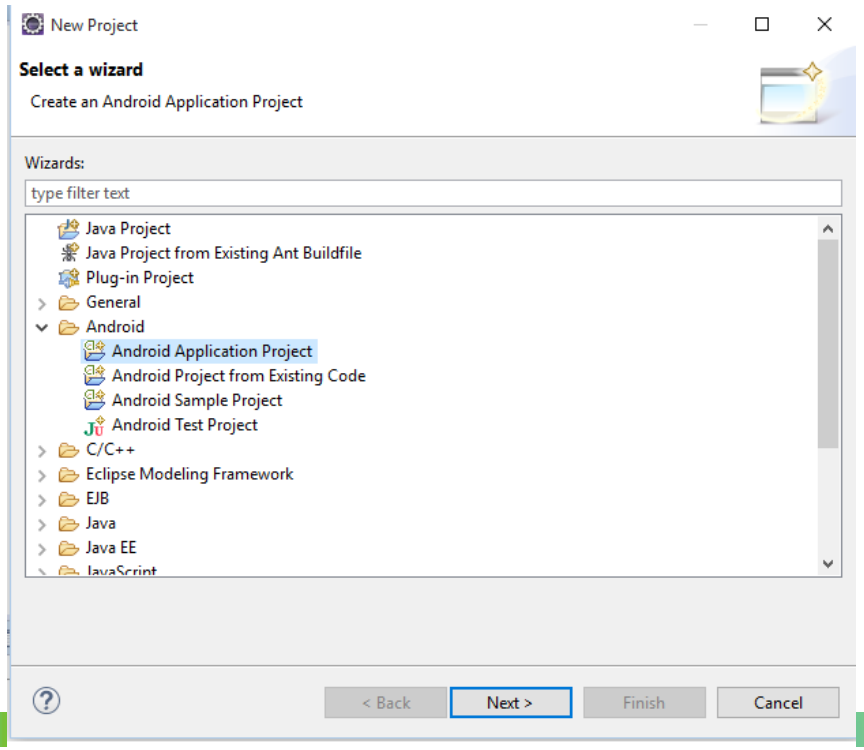
Početak kreiranja Android aplikacije

Android aplikacije imaju formu projekata.

Razvoj svake Android aplikacije započinje pokretanjem Eclipse IDE razvojnog pkruženja. Nakon toka iz menija *File,* bira se opcija *New*, za zatim *Project.*

Nakon klika na opciju Project otvara se prozor u kojem se, u meniju Android, bira opcija Android Project.





Početak kreiranja Android aplikacije

Svi fajlovi čuvaju se u folderu projekta.

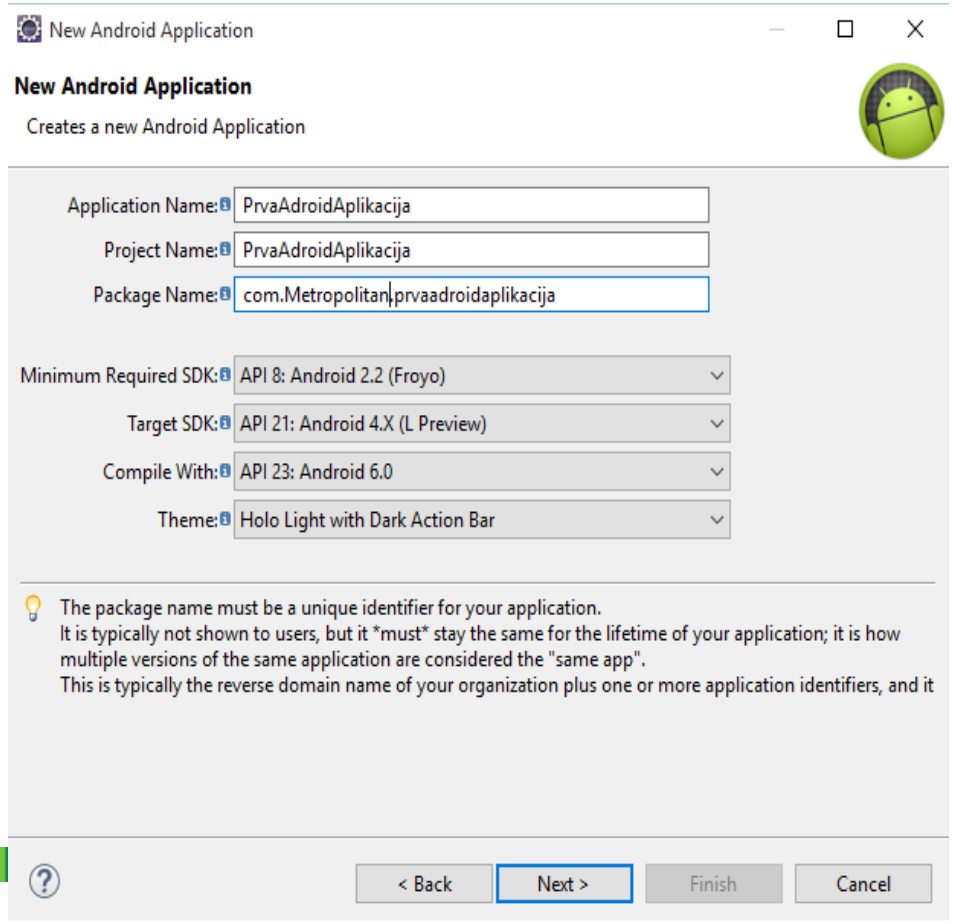
U sledećem koraku neophodno je dodeliti naziv projektu, aplikaciji i paketu koji će čuvati dokumentaciju projekta. Treba izabrati API koji odgovara verziji Android operativnog sistema za koju se aplikacija razvija. U konkretnom slučaju ciljni API je verzija Android 4.4 KitKat, kompajliranje će biti obavljeno najnovijom verzijom za Android 6.0 Ključne informacije o aplikaciji su sledeće:

Naziv projekta: PrvaAdroidAplikacija;

Naziv aplikacije: PrvaAdroidAplikacija;

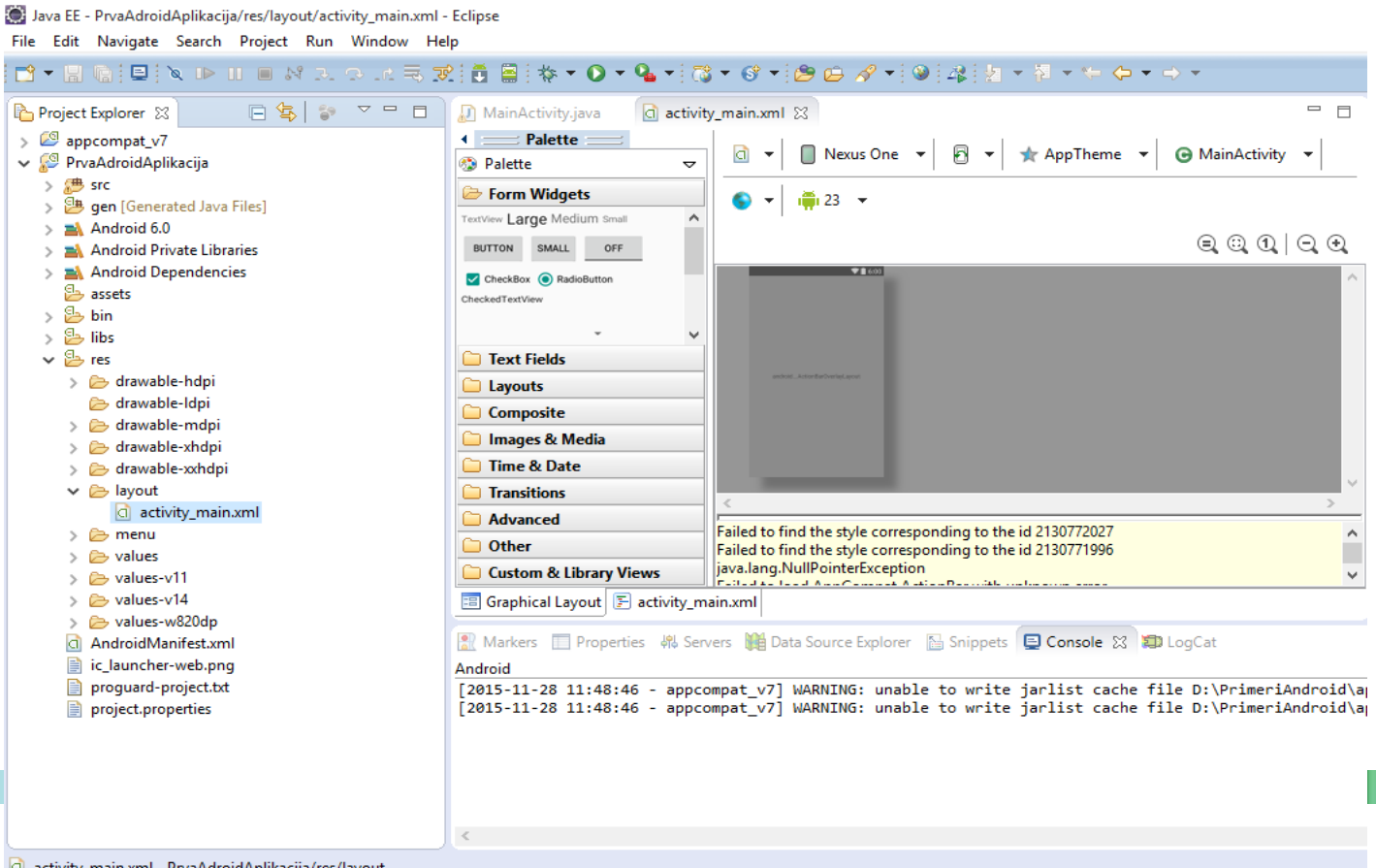
Naziv paketa: com.Metropolitan.prvaadroidaplikacija.

Klikom na Finish završavaju se inicijalna podešavanja i Eclipse IDE je spreman za razvoj prve Android aplikacije



Activity_main.xml datoteka

Korisnički interfejs aplikacije definisan je activity_main.xml datotekom. Pogledati panel pod nazivom Project Explorer U Eclipse integrisanom razvojnom okruženju. U **res/layout** folderu dvostrukim klikom bira se datoteka pod imenom **activity_main.xml**. Ova datoteka definiše korisnički interfejs aplikacije.



Activity_main.xml datoteka - primer

Inicijalni kod datoteka moguće je korigovati dodavanjem novih komponenata korisničkog interfejsa.

Inicijalni kod datoteke activity_main.xml sledi ispod:

MainActivity.java | *activity_main.xml | strings.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/app_name" />

</LinearLayout>
```

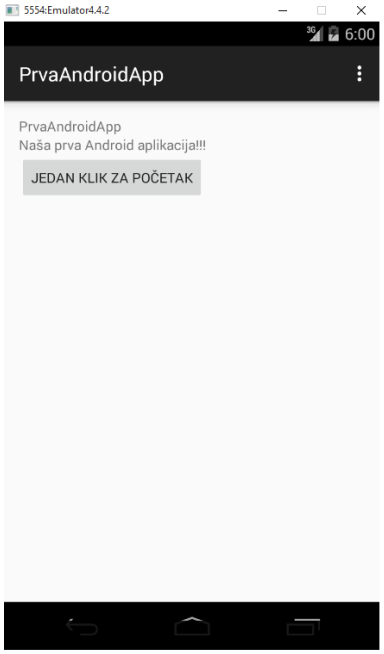
Sada je moguće ubaciti i neke vlastite korekcije, npr. još malo teksta i jedno dugme. Ispod prvog xml taga <TextView…/>, a pre završnog taga </LinearLayout>, može se ubaciti sledeći kod:

```
<TextView
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="Naša prva Android aplikacija!!!" />
<Button
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="Jedan klik za početak" />
```

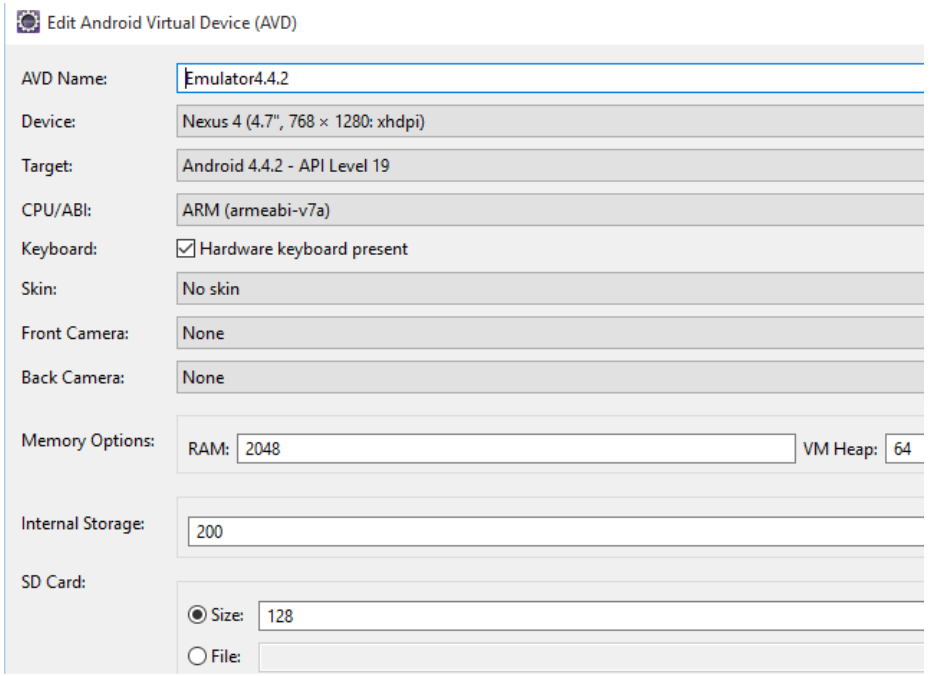
Prevođenje i demonstracija primera

Nakon snimanja projekta, izborom Run As (Android Application) emulatorom se startuje kreirana aplikacija.

Sada je moguće pristupiti prevođenju i testiranju kreirane aplikacije. Prvo je neophodno snimiti projekat, npr sa CTRL+S, za zatim desnim klikom na naziv projekta selektovati Run As, pa Android Application.



Emulator koji je kreiran simulira rad uređaja na KitKat Androidu i definisan je na način prikazan sledećom slikom.



Anatomija Android aplikacije

Android aplikacija sadrži različite datoteke prikazane u Package Explorer panelu u Eclipse okruženju; Postoje sledeći folderi i u njima odgovarajuće datoteke:

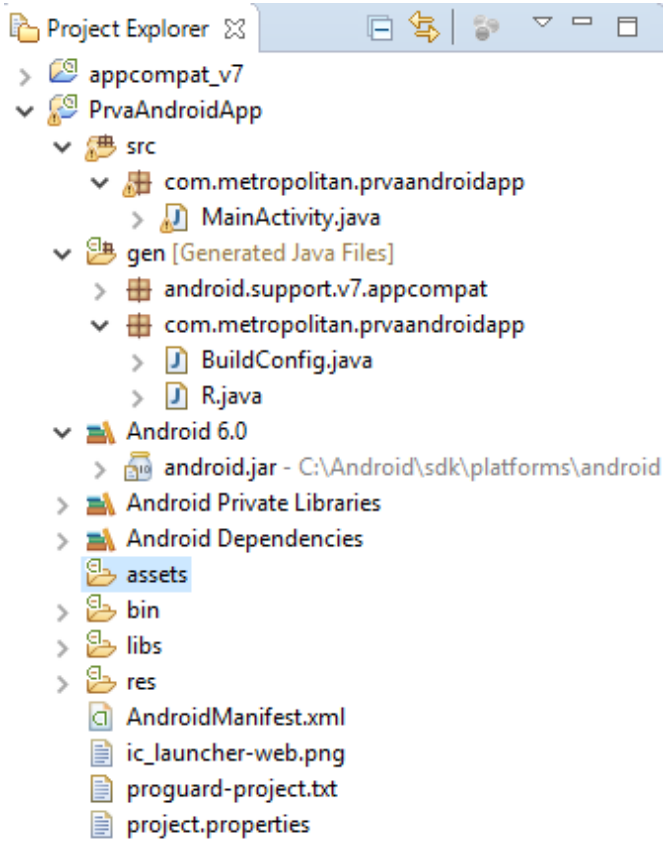
- **src** – sadrži **.java** izvorne datoteke u projektu. Java datoteka je prikazana u okviru naziva paketa projekta
- **gen** – sadrži **R.java** datoteku, koju je generisao prevodilac, a koja referencira sve resurse u projektu; nju ne treba menjati; Svi resursi u projektu se automatski prevode u ovu klasu i referenciraju se njenim korišćenjem
- **Android library** – sadrži samo jednu datoteku - **android.jar**, koja sadrži sve biblioteke klasa neophodne za jednu Android aplikaciju

Anatomija Android aplikacije

- **assets** – ovaj folder sadrži sva sredstva koja koristi aplikacija, kao što su HTML, tekstualne datoteke, baze podataka itd.
- **bin** – ovaj folder sadrži datoteke koje je kreirao ADT u procesu prevođenja; generiše se **.apk** datoteka (*Android Package*) datoteka; datoteka sa ekstenzijom .apk je binarni kod Android aplikacije i ona sadrži sve što je neophodno za izvršavanje jedne Android aplikacije
- **res** – ovaj folder sadrži sve resurse koji se koriste u aplikaciji; sadrži i druge potfoldere: drawable-<resolution>, layout i values
- **AndroidManifest.xml** – ovo je manifest datoteka za Android aplikaciju. U njoj se definišu privilegije koje su neophodne za aplikaciju, kao i ostale funkcije (kao što su filter sadržaja, primaoci i slično).

Organizacija Package Explorera

Svi folderi i datoteke aplikacije dostupni su u Package Exploreru,



Neke datoteke projekta koje su od posebnog značaja za kreiranje i funkcionisanje aplikacije:

1. activity_main.xml (u starijim verzijama main.xml) datoteka ima za namenu definisanje korisničkog interfejsa datoteke. Posebno bi trebalo napomenuti da je instrukcijom @string/app_name preuzet string definisan u datoteci strings.xml, a koji odgovara nazivu aplikacije. Upravo je i preporuka da se stringovi, koji se koriste u aplikaciji, čuvaju u navedenoj datoteci i da se na njih vrši referenciranje primenom identifikatora @string/* .

2. AndroidManifest.xml je veoma važna datoteka koja sadrži detaljne informacije o aplikaciji kao što su:

- naziv paketa – u našem slučaju paket ima naziv com.metropolitan.prvaandroidapp;
- identifikator verzije aplikacije;
- minimalnu i ciljanu verziju Android OS kojima je aplikacija namenjena;

Organizacija Package Explorera - nastavak

AndroidManifest.xml je datoteka u kojoj su definisane aktivnosti aplikacije.

- aplikacija koristi sliku ic_launcher.png iz drowable foldera;
- android:name=".MainActivity“ instrukcijom ukazuje se na aktivnost u aplikaciji;
- Posebnu pažnju, u okviru aktivnosti, trebalo bi obratiti na xml tag <intent-filter> … </intent-filter> u okviru kojeg se ukazuje na početnu tačku apliacije (android.intent.action.MAIN), kao i na mogućnost pokretanja aplikacije pomoću launcher ikone(android.intent.category.LAUNCHER).

Sledećim xml kodom prikazan je sadržaj datoteke AndroidManifest.xml sa navedenim informacijama.

MainActivity.java | activity_main.xml | strings.xml | PrvaAndroidApp Manifest

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

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="21" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Manifest | Application | Permissions | Instrumentation | AndroidManifest.xml

R.Java datoteka

R.java je automatski ažurirana od strane Eclipse IDE.

Tokom procesa dodavanja datoteka i foldera u Android projekat, datoteka R.java će automatski biti ažurirana od strane Eclipse IDE razvojnog okruženja i nije predviđeno da programer na bilo koji način modifikuje navedenu datoteku.

Konačno, datoteka MainActivity.java metodom setContentView() povezuje korisnički interfejs sa aktivnošću.

MainActivity.java | activity_main.xml | strings.xml | PrvaAndroidApp Mani... | R.java

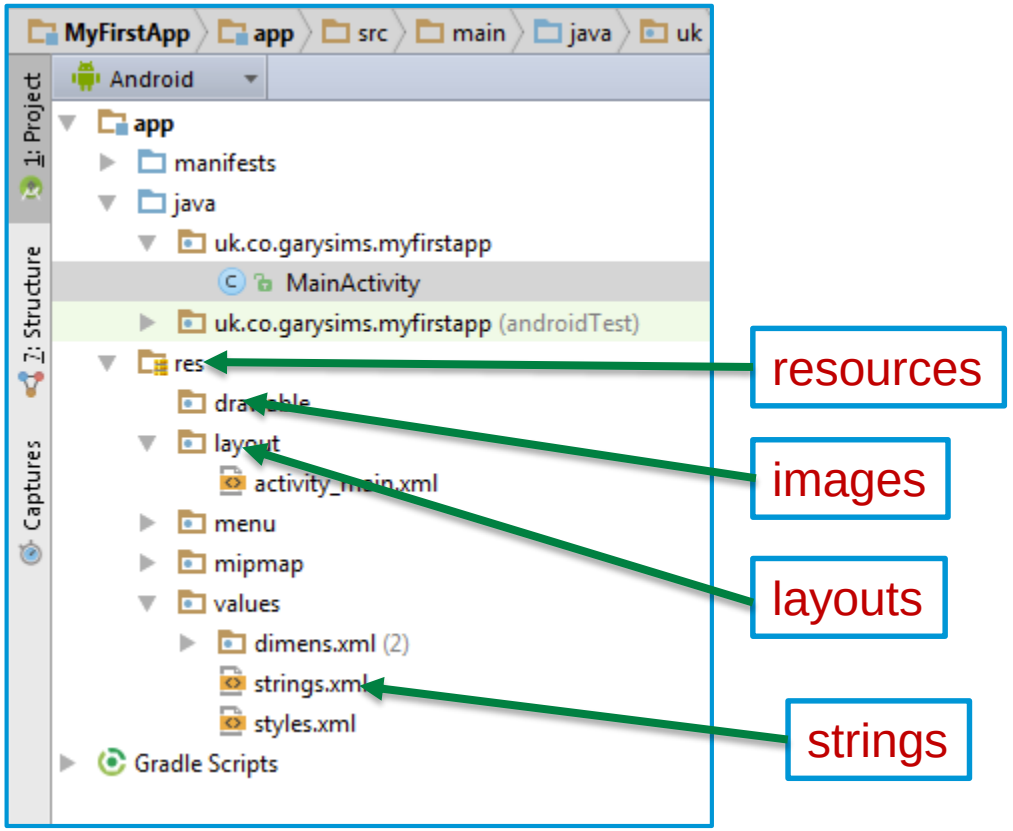
```
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 */

package com.metropolitan.prvaandroidapp;

public final class R {
    public static final class anim {
        public static final int abc_fade_in=0x7f040000;
        public static final int abc_fade_out=0x7f040001;
        public static final int abc_grow_fade_in_from_bottom=0x7f040002;
        public static final int abc_popup_enter=0x7f040003;
        public static final int abc_popup_exit=0x7f040004;
        public static final int abc_shrink_fade_out_from_bottom=0x7f040005;
        public static final int abc_slide_in_bottom=0x7f040006;
        public static final int abc_slide_in_top=0x7f040007;
        public static final int abc_slide_out_bottom=0x7f040008;
        public static final int abc_slide_out_top=0x7f040009;
    }
```

MainActivity.java | strings.xml

```
public class MainActivity extends ActionBarActivity {

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();
        if (id == R.id.action_settings) {
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

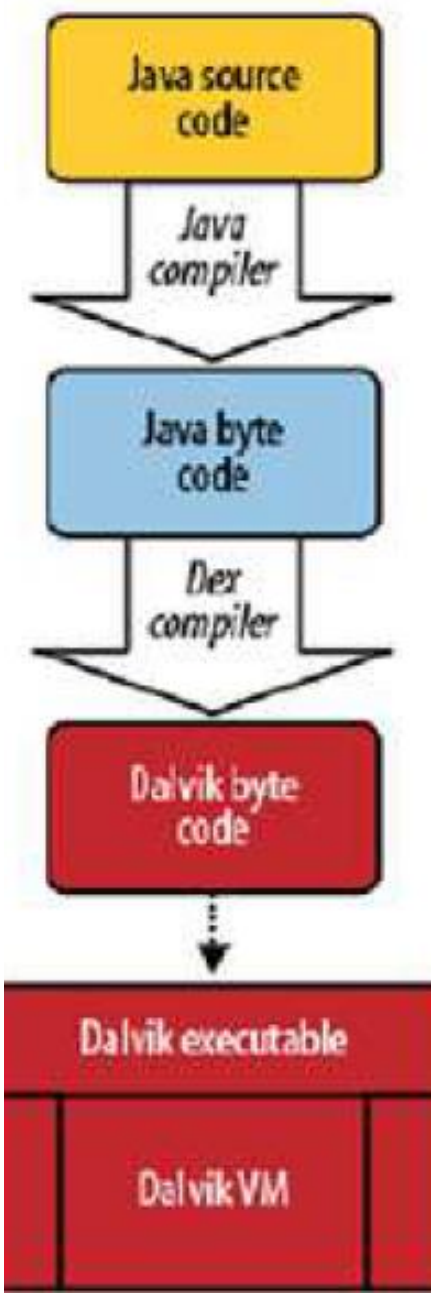
Struktura Android aplikacije

- *Resource* XML fajlovi se nalaze u odvojenom delu projekta, u folderu ***res***



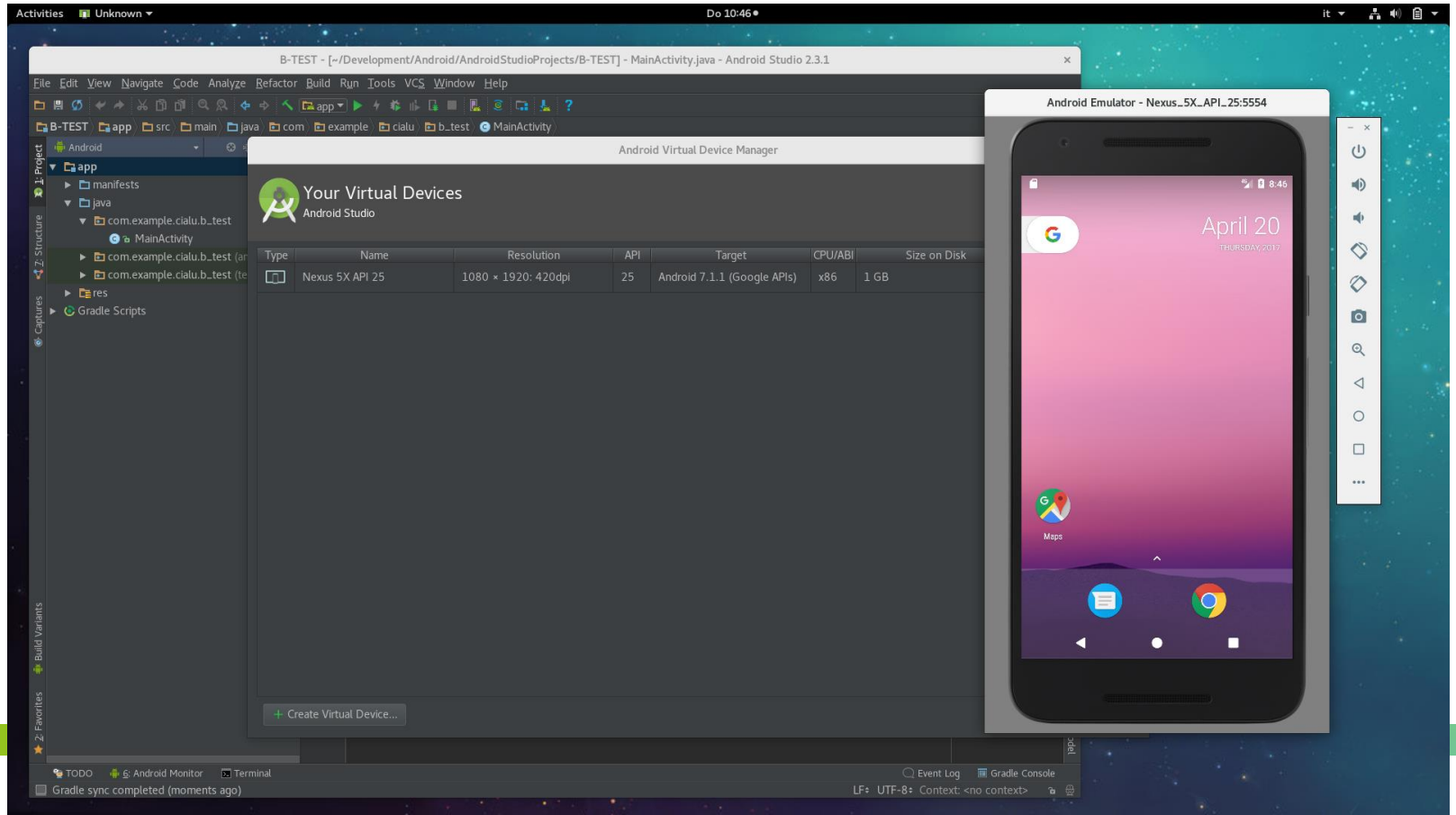
Dalvik Virtuelna mašina

- Projektovana specijalno za Android
- Pošto VM Dalvik zahteva drugačiji bajtkod koji će interpretirati, to znači da ne podržava standardne Java class fajlove, već se oslanja na sopstveni format: Dalvik Executable (DEX)
- Android platforma za programiranje sadrži skup alata za postprocesiranje kompajliranih Java class fajlova u format DEX
- Za razliku od standardnih Java aplikacija koje sadrže više class fajlova, DEX spaja sve class fajlove u jedinstveni DEX fajl



Android Emulator

- Okruženje za razvoj Android aplikacija uključuje skup emulatora virtuelnih mobilnih uređaja. Na ovaj način je omogućen razvoj aplikacija na računaru bez korišćenja pravih uređaja. U zadnjoj fazi obavezno treba testirati aplikaciju na stvarnim uređajima!



Android Emulator

- Služi za testiranje i debagovanje aplikacija
- To je implementacija Dalvik VM; pošto nije vezana za određeni hardver, odlična je za testiranje programa
- Podržava mrežu, podešavanje brzine Interneta, simulira slanje i primanje poziva i SMS poruka
- Ne implementira kameru, LED-ove, akcelerometar, USB povezivanje, snimanje zvuka, indikator baterije

Programski jezik Android aplikacija

- Najveći deo Android aplikacija napisan je u Javi
- Postoje i druge opcije:

– Delovi koda mogu se pisati u jezicima C/C++ da bi se poboljšale performanse, ili ako se koristi OpenGL za 3D animacije

– Delovi aplikacije mogu se pisati kao HTML, CSS i JavaScript, pa se taj materijal pakuje u Android aplikaciju koja se može distribuirati preko Play Store i sl

- Skup Javinih biblioteka za Android najbliži je verziji Java SE
- Najveća razlika je u tome što su Javine biblioteke za korisnički interfejs (AWT i Swing) izostavljene i zamenjene bibliotekama specifičnim za Android

Programski jezik Android aplikacija

- Najveći deo Android aplikacija napisan je u Javi
- Postoje i druge opcije:

– Delovi koda mogu se pisati u jezicima C/C++ da bi se poboljšale performanse, ili ako se koristi OpenGL za 3D animacije

– Delovi aplikacije mogu se pisati kao HTML, CSS i JavaScript, pa se taj materijal pakuje u Android aplikaciju koja se može distribuirati preko Play Store i sl

- Skup Javinih biblioteka za Android najbliži je verziji Java SE
- Najveća razlika je u tome što su Javine biblioteke za korisnički interfejs (AWT i Swing) izostavljene i zamenjene bibliotekama specifičnim za Android