XII – Debagiranje aplikacija

S A D R Ž A J

12.1 Debagiranje Android aplikacija

12.2 Korišćenje prekidnih tačaka

12.3 Interpreterski način rada

12.4 Pregled i modifikovanje promenljivih

12.5 Ostali alati za otkrivanje grešaka

12.1 – Debagiranje Android aplikacija

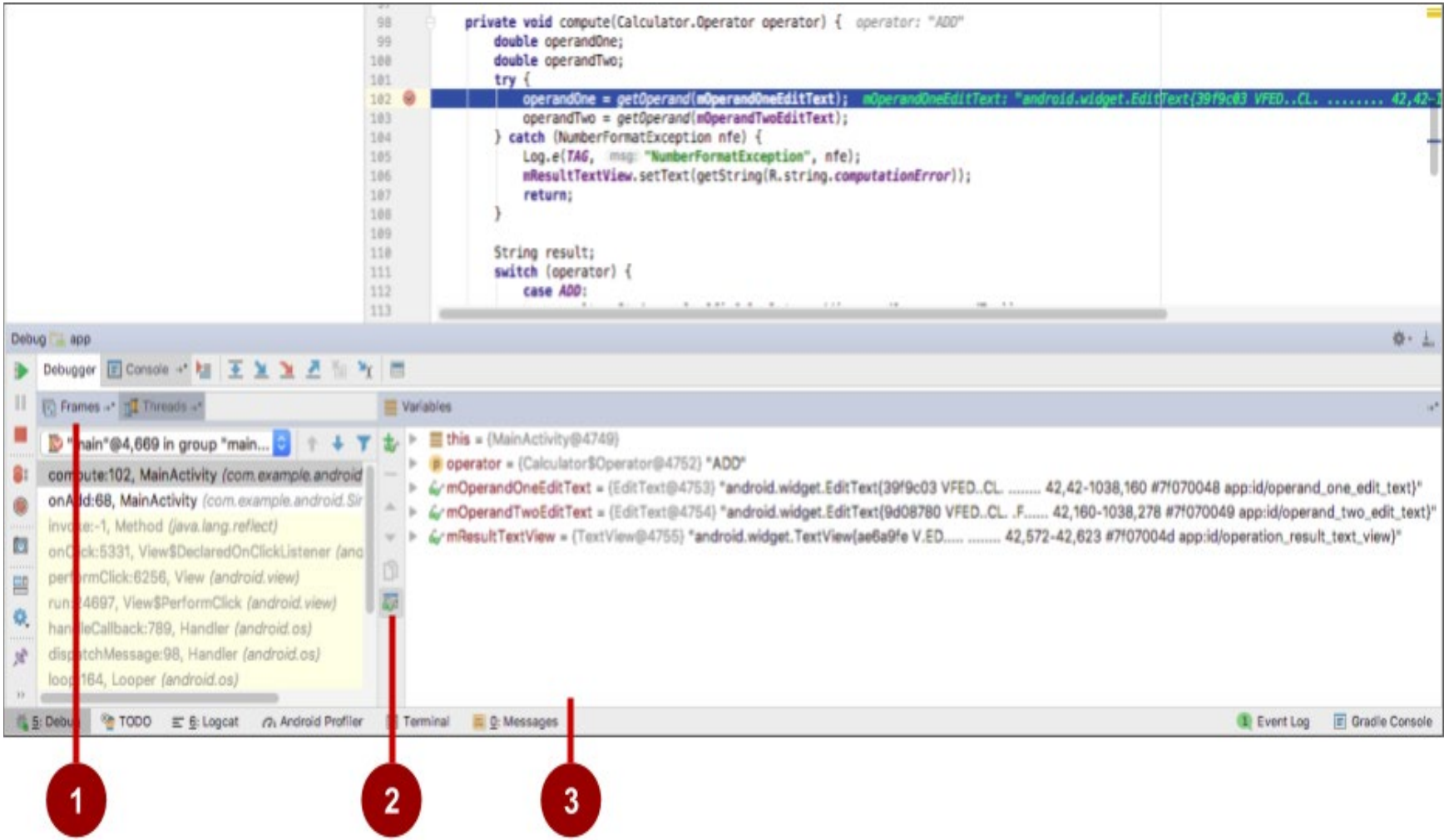
- ***Debugging*** predstavlja proces pronalaženja i ispravljanja grešaka ili neočekivanog ponašanja aplikacije.
- Svi kodovi imaju greške, od pogrešnog ponašanja u vašoj aplikaciji, do ponašanja koje previše troši memoriju ili mrežne resurse, do stvarnog zamrzavanja aplikacije ili pada aplikacije.
- Postoji više razloga koji mogu da budu uzrok greške:
  1. Greške u dizajnu ili implementaciji aplikacije
  2. Ograničenja u okviru Android (greške u korišćenom *framework*-u)
  3. Nedorečeni zahtevi ili pretpostavke kako aplikacija treba da radi
  4. Ograničenja uređaja na kome radi aplikacija
- Android studio sadrži više alata koji nam pomažu da otklonimo greške, testiramo i profilišemo našu aplikaciju:
  1. Prozor **Logcat** u kome se prikazuju događaji vezani za rad aplikacije
  2. Prozor **Debugger** za pregled okvira, niti i promenljivih u aplikaciji
  3. **Debug mod** rada koji omogućava rad aplikacije sa tačkama prekida
  4. Test framework kao što su **JUnit** ili **Espresso**
  5. Dalvik Debug Monitor Server(DDMS) za praćenje korišćenja resursa

12.1 - Pokretanje Debugger-a

- Pokretanje aplikacije u režimu za otklanjanje grešaka slično je pokretanju same aplikacije.
- Aplikaciju možete da pokrenete u režimu **debug** ili da ***debugger*** program aktivirate tj. priključite aplikaciji koja radi
- Aplikaciju pokrećete u debug modu klikom na Debug ikonicu u toolbar
- Android Studio pravi **APK** i instalira ga na izabrani uređaj, zatim ga pokreće i otvara prozor za debug sa karticama **Debugger** i **Console**.
- Kartica **Debugger** prikazuje sledeće funkcije:

1. **Kartica Okviri**: kliknite da biste prikazali prozor **Okviri** sa trenutnim okvirima stack izvršenja za datu nit. Stack za izvršavanje prikazuje svaku klasu i metod koji su pozvani u vašoj aplikaciji i u Android runtime-u, sa najnovijom metodom na vrhu.
2. Klik na karticu Threads aktivira prozor Frames sa prozorom Threads.
3. Dugme **Watches**: izborom prikazujete prozor **Watches** u **Variables**, koji prikazuje vrednosti za bilo koje izabrane promenljive.
4. Prozor **Variables**: prikazuje trenutnu vrednost svake promenljive. Svaka promenljiva u ovom prozoru ima posebnu poziciju/red

12.1 - Pokretanje Debugger-a



12.1 - Pokretanje Debagger-a iz aplikacije

- Ako je vaša aplikacija već pokrenuta na uređaju ili emulatoru, ***debugger*** pokrećemo na sledeći način:

1. Izaberite ***Run>Attach debugger to Android process*** ili klikom na ikonu **Attach**  iz **toolbar**-a (traka sa alatkama).
2. U dijalogu **Choose Process** bira se proces u koji želite da priključite ***debugger*** kako bi mogli da pratite vaš proces na uređaju
3. Kliknite **OK** kako bi prikazali prozor **Debug**

- Da biste nastavili sa izvršavanjem aplikacije nakon što je ispravite, izaberite ***Run>Resume Program*** ili kliknite na ikonu **Resume** 
- Da biste zaustavili otklanjanje grešaka u aplikaciji, izaberite **Run> Stop** ili kliknite na ikonu **Stop**  iz toolbar-a.

12.2 – Korišćenje prekidnih tačaka

- Android Studio podržava nekoliko vrsta prekidnih tačaka koje omogučuju otklanjanje različitih grešaka.
- Najčešći tip prekidne tačka omogučuje pauziranje tj. zaustavljanje izvršavanja aplikacije na određenoj liniji koda.
- Dok je aplikacija zaustavljena, možete ispitati promenljive, izračunati izraze, a zatim nastaviti redosled izvršavanja kako biste utvrdili uzroke grešaka u toku rada aplikacije.
- Prekidnu tačku možete postaviti na bilo koju izvršnu liniju koda.
- Da biste dodali prekidnu tačku u vaš programski kod potrebno je da uradite sledeće:

1. Pronađite liniju koda gde želite da zaustavite izvršenje programa.
2. Kliknite na levu stranu editor prozora na izabranoj liniji, pored brojeva redova. Na toj liniji se pojavljuje **crvena tačka** koja pokazuje tačku prekida. Crvena tačka sadrži oznaku ako je aplikacija već pokrenuta u debug režimu.

- Alternativno, možete izabrati ***Run> Toggle Line Breakpoint*** ili **Control-F8** da biste postavili ili izbrisali tačku prekida na liniji.

12.2 - Korišćenje prekidnih tačaka

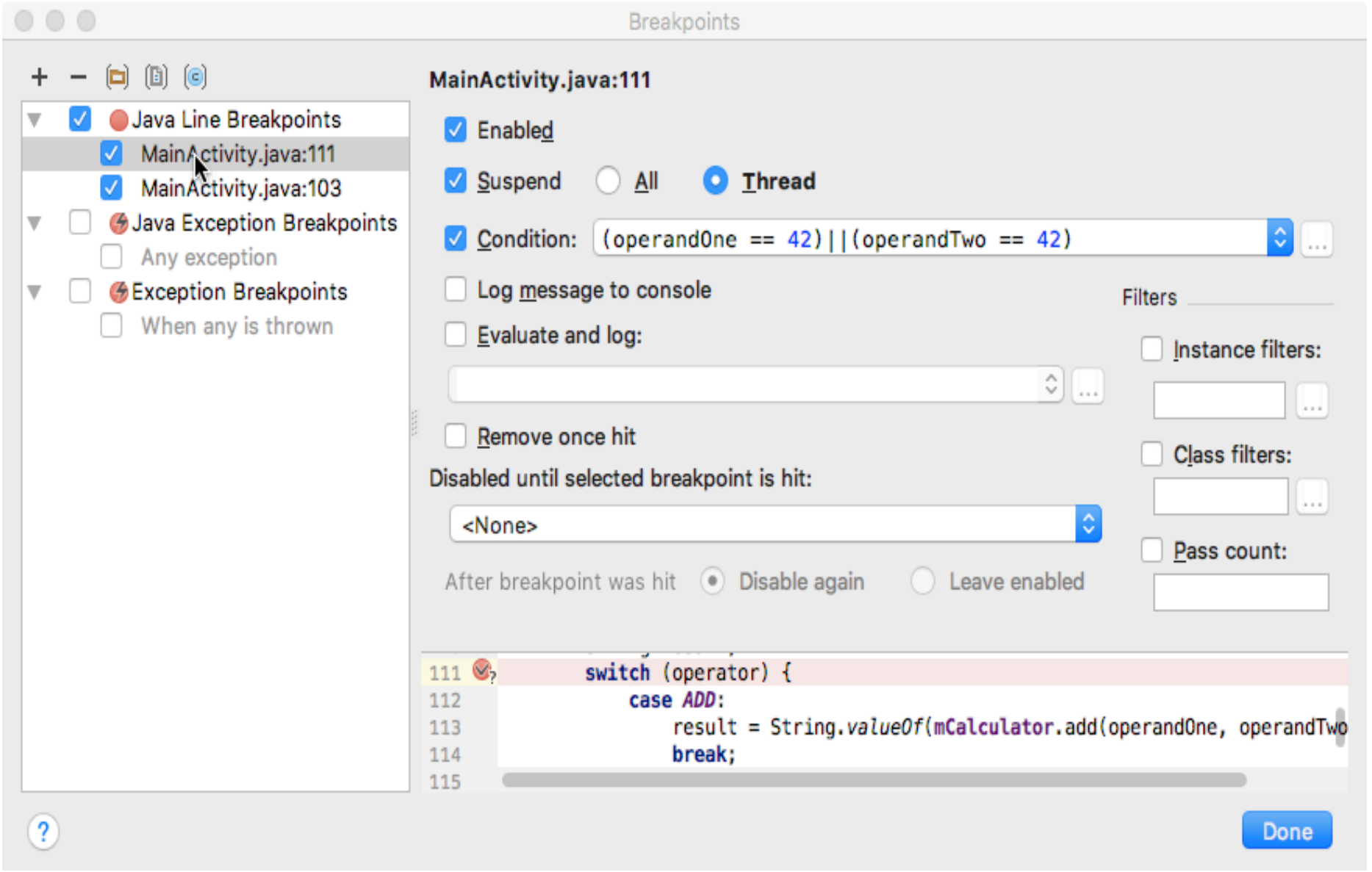
- Ako je vaša aplikacija već pokrenuta, ne morate je ažurirati da biste dodali tačku prekida.
- Ako greškom kliknete na tačku prekida, možete je poništiti klikom na tačku prekida.
- Ako ste kliknuli na liniju koda koja nije izvršna, crvena tačka uključuje "**k**" i pojavljuje se upozorenje da linija koda nije izvršna.
- Kada izvršenje koda dostigne tačku prekida, Android Studio prekida izvršavanje vaše aplikacije.
- Zatim možete da koristite alate u prozoru Debug da biste videli stanje aplikacije i otklonili greške u aplikaciji dok se ona pokreće.

```
98      private void compute(Calculator.Operator operator) {
99          double operandOne;
100         double operandTwo;
101         try {
102             operandOne = getOperand(mOperandOneEditText);
103             operandTwo = getOperand(mOperandTwoEditText);
104         } catch (NumberFormatException nfe) {
105             Log.e(TAG,  msg: "NumberFormatException", nfe);
106             mResultTextView.setText(getString(R.string.computationError));
107             return;
108         }
109
110         String result;
111         switch (operator) {
```

12.2 – Korišćenje prekidnih tačaka

- Da biste videli sve tačke prekida koje ste postavili i konfigurisali kliknite na ikonu **View Breakpoints**  na levoj ivici prozora **Debug**.
- Pojavljuje se prozor **Breakpoints** sa svim prekidnim tačkama koje ste postavili koje sada možemo pojedinačno omogućiti ili onemogućiti.
- Ako je tačka prekida onemogućena, Android Studio ne pauzira vašu aplikaciju kada izvršenje dostigne tačku prekida.
- Izaberite tačku prekida sa liste da biste konfigurisali njene postavke.
- Možete konfigurirati tačku prekida da bude onemogućena na početku i da je sistem omogući nakon što se naiđe na drugu tačku prekida.
- Takođe možete da konfigurišete da li će tačka prekida biti onemogućena nakon što je dostignuta.
- Da biste postavili tačku prekida za bilo koji izuzetak, izaberite **Exception Breakpoints** u listi tačaka prekida.

12.2 - Korišćenje prekidnih tačaka

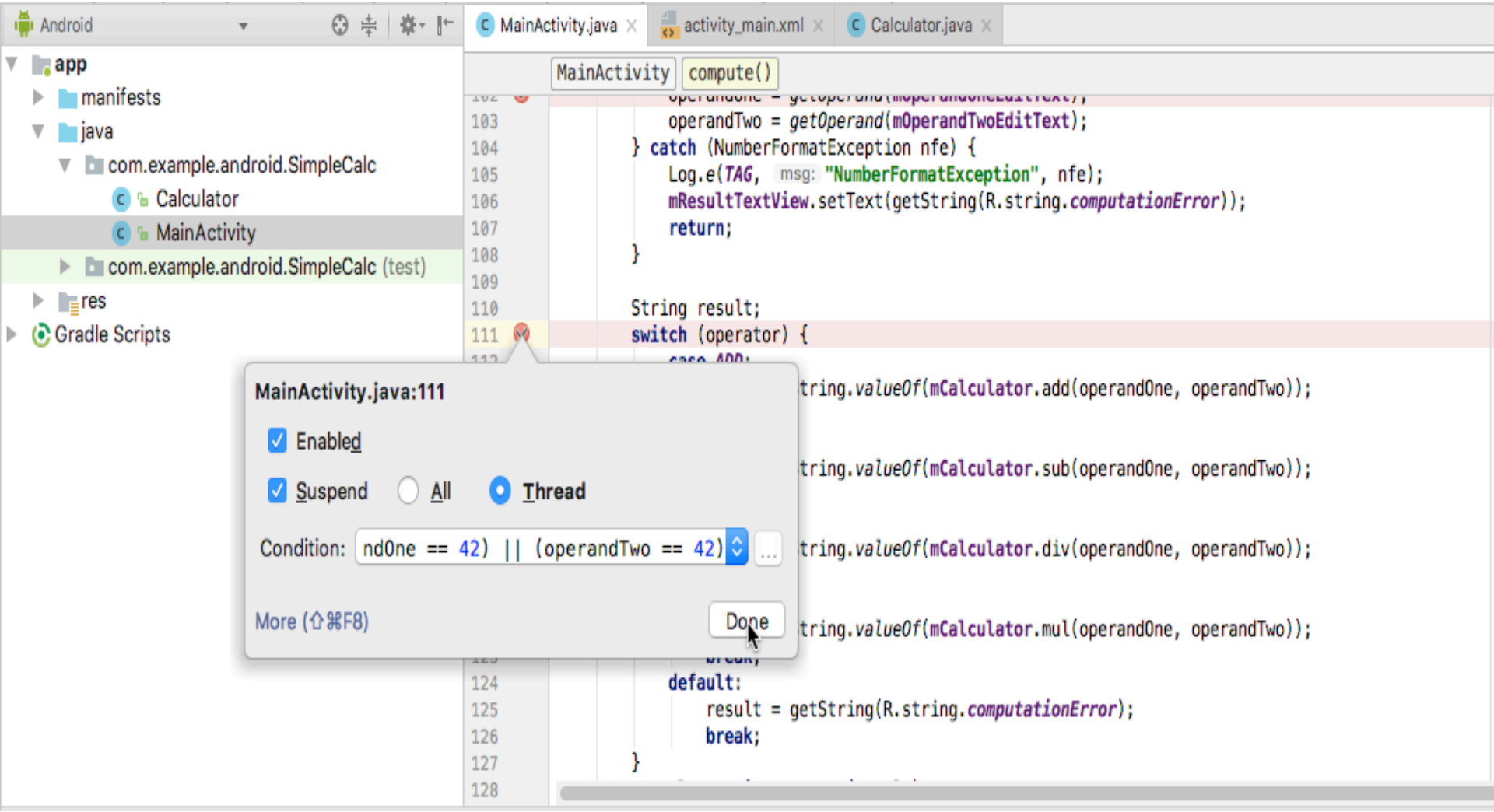


12.2 – Korišćenje prekidnih tačaka

- Onemogućavanje tačke prekida vam omogućava da privremeno onemogućite tu tačku prekida bez da je uklonite iz koda.
- Ako potpuno uklonite tačku prekida, takođe ćete izgubiti sve uslove ili druge karakteristike koje ste kreirali za tu tačku prekida, tako da onemogućavanje može biti bolji izbor.
- Za isključivanje svih tačaka prekida klik na ikonu Mute Breakpoints 
- Ponovni klik na ikonu omogućuje (uključuje) sve tačke prekida.
- Uslovne tačke prekida su tačke prekida koje samo zaustavljaju izvršavanje vaše aplikacije ako se ispuni postavljeni uslov.
- Da biste definisali test za uslovnu tačku prekida, koristite ove korake:

1. Kliknite desnim tasterom miša na (ili **Control-click**) tačku prekida i unesite uslov u polje **Condition**. Uslov koji unesete u ovo polje može biti bilo koji Java izraz sve dok vraća ***boolean*** vrednost. Koristimo imena promenljivih iz alikacije kao deo izraza. Takođe možemo da koristimo prozor **Breakpoints** da bi uneli stanje prekidne tačke.
2. Pokrenite aplikaciju u režimu za otklanjanje grešaka i ako je uslov ispunjen aplikacija se zaustavlja se na uslovnoj tački prekida.

12.2 - Korišćenje prekidnih tačaka



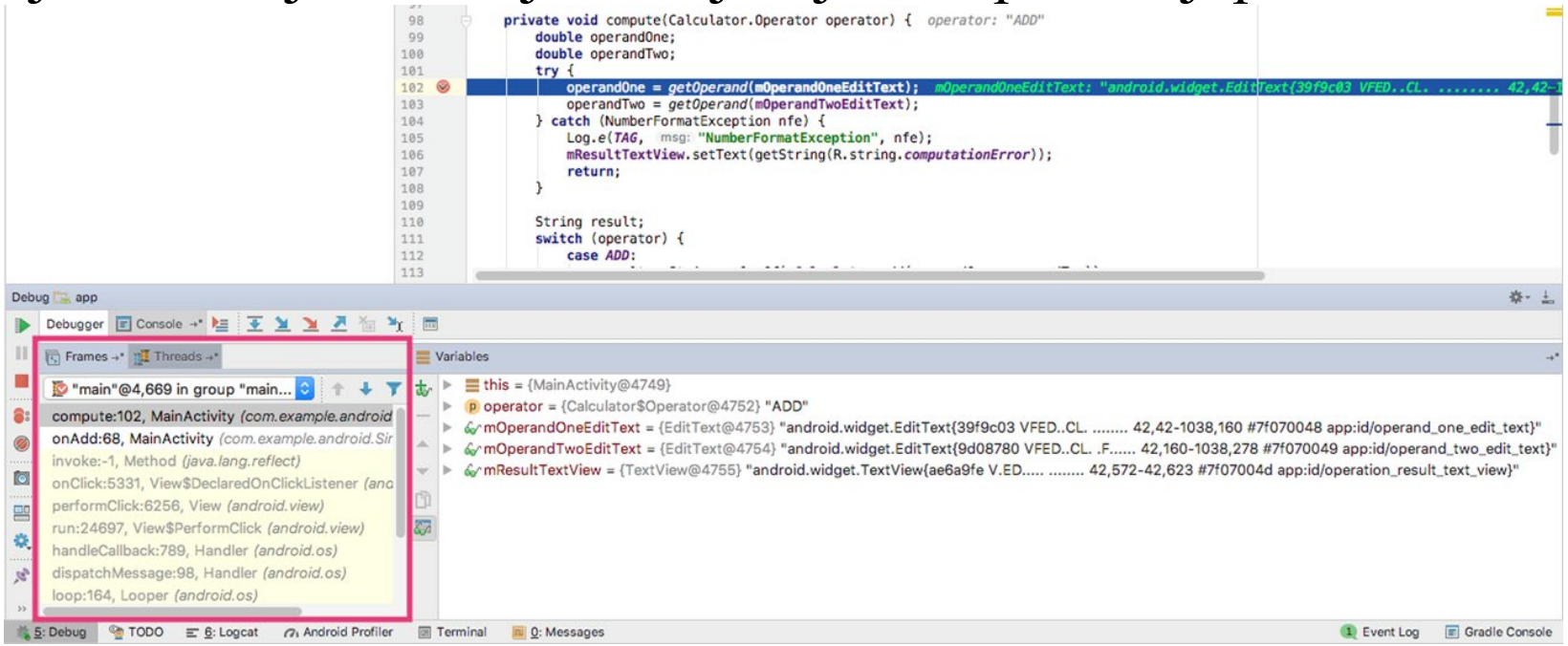
12.3 - Interpreterski način rada

- Predstavlja poseban način izvršavanja program. koda: **korak po korak**
- Kada se izvršavanje aplikacije zaustavi zato što je dostignuta prekidna tačka, za dalje izvršavanje koristimo **Step Over**, **Step Into** i **Step Out**.
- Da bismo koristili bilo koju od ovih funkcija potrebno je:

1. Pauzirajte izvršavanje aplikacije sa tačkom prekida. Prozor Debugger prikazuje stanje aplikacije. Trenutna linija je istaknuta u vašem kodu.
2. Kliknite na ikonu **Step Over**,  izaberite **Run> Step Over** ili **F8**. **Step over** izvršava sledeći red koda u trenutnoj klasi i metodi, izvršavajući sve pozive metoda na tom mestu i ostajući na istoj liniji.
3. Kliknite na ikonu **Step Into**,  izaberite **Run> Step Into** ili pritisnite **F7**. **Step Into** prelazi u izvršenje poziva metode na trenutnoj liniji. Prozor **Frames** ažurira se kako bi prikazao novi okvir (novi metod). Ako je poziv metode sadržan u drugoj klasi, otvara se datoteka za tu klasu i označava trenutna linija njoj.
4. Kliknite na ikonu **Step Out**,  izaberite **Run> Step Out** ili **Shift-F8**. Završava se trenutna metoda i vraća se na mesto odakle je pozvana.
5. Za nastavak izvršavanja, izaberite **Run>Resume Program** ili kliknite na ikonu **Resume** 

12.4 - Pregled izvršavanja aplikacije

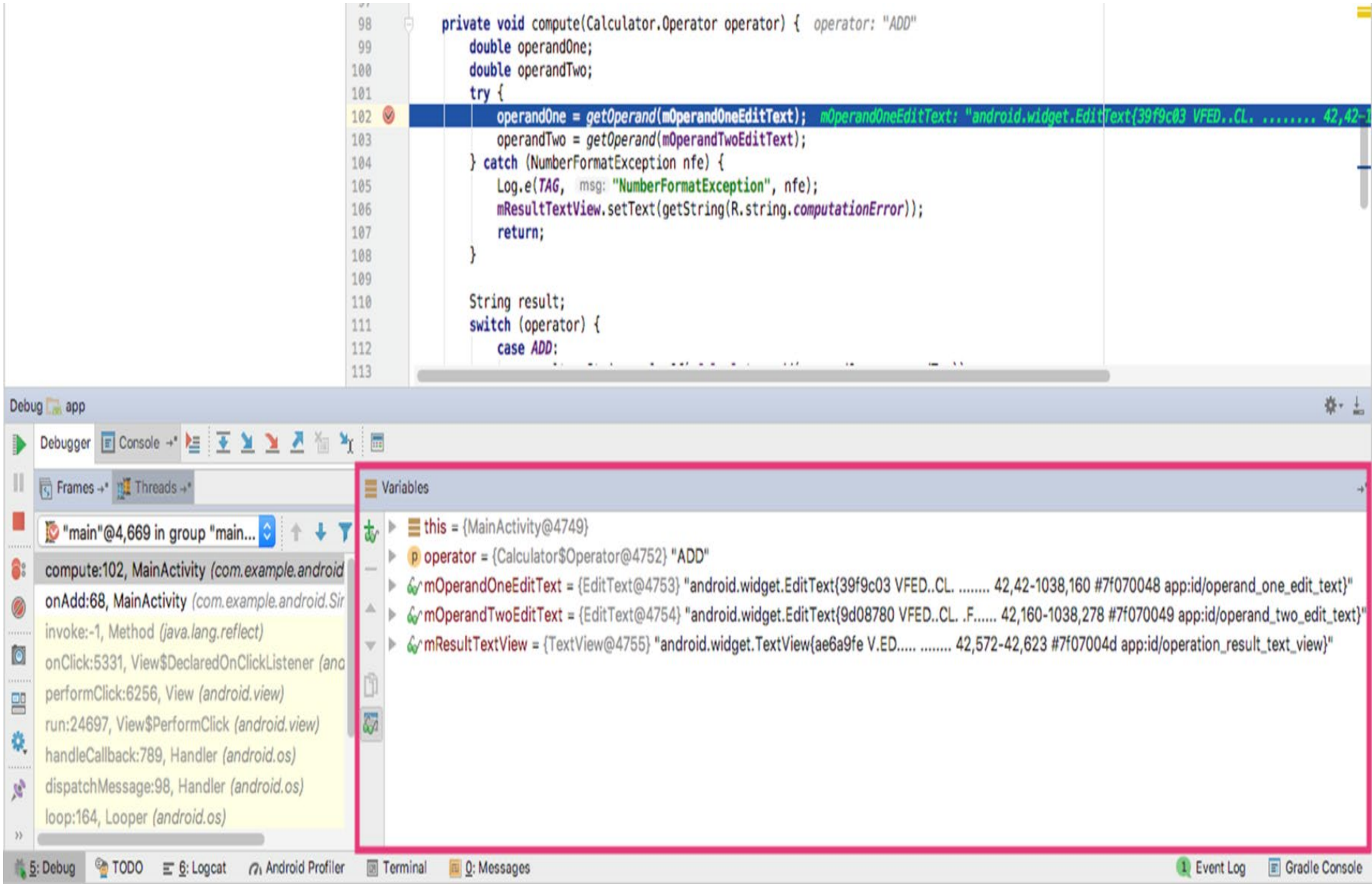
- **Frames** u okviru **Debug** prozora omogućava da vidimo stack izvršenja i određenu metodu koja je izazvala dostizanje trenutne tačke prekida.
- Izvršni **stack** prikazuje sve klase i metode (okvire) koji se izvršavaju do ove tačke u aplikaciji, obrnutim redosledom (najnoviji okvir prvo).
- Kako se izvršavanje nekog okvira završava, taj okvir izlazi iz **stack-a**
- Klikom na liniju za okvir u prozoru **Frames** otvara se pridruženi izvor u editoru i ističe linija u kojoj je taj okvir prvobitno izvršen.
- Prozori **Variables** i **Vatches** takođe se ažuriraju tako da odražavaju stanje okruženja izvršenja kada je taj okvir poslednji put unesen.



12.4 - Pregled i modifikovanje promenljivih

- Prozor **Variables** u **Debugger**-u vam omogućava da pregledate promenljive dostupne u trenutnom okviru **steka** kada sistem zaustavi vašu aplikaciju na tački prekida.
- Promenljive koje sadrže objekte ili kolekcije kao što su polja mogu se proširiti da bi se prikazale njihove komponente.
- Prozor **Variables** vam takođe omogućava da pratite izraze u toku izvršavanja pomoću statičkih metoda ili promenljivih dostupnih unutar izabranog okvira.
- Ako ne vidite prozor **Variables**, kliknite na **Restore Variables View** 
- Da biste promenili promenljive u aplikaciji dok se pokreće:
  1. Kliknite desnim tasterom miša na (ili **Control-click**) bilo koju promenljivu u prozoru **Variables** i izaberite **Set Value**. Takođe možete pritisnuti **F2**.
  2. Unesite novu vrednost za promenljivu i pritisnite **Return**.
- Vrednost koju unesete mora biti odgovarajućeg tipa za tu promenljivu, ili Android Studio vraća grešku "neusklađenost tipa".

12.4 - Pregled i modifikovanje promenljivih



12.4 – Postavljanje Watches

- Prozor **Watches** pruža sličnu funkcionalnost prozoru **Variables**, osim što izrazi dodati u **Watches** i dalje postoje između sesija debugovanja
- Dodajte **Watches** za promenljive i polja kojima često pristupate ili koji obezbeđuju stanje koje je korisno za trenutno otklanjanja grešaka
- Da biste koristili **Watches**:
  1. Počnite sa debagiranjem vaše aplikacije.
  2. Kliknite na ikonu **Show Watches icon**. Pojavljuje se prozor **Watches** pored prozora promenljivih.
  3. U prozoru **Watches** kliknite na dugme plus (+).
  4. U tekstualnom okviru koji se pojavi, unesite ime promenljive ili izraza koji želite da gledate, a zatim pritisnite taster **Enter**.
  5. Uklonite stavku sa liste **Watches** tako što ćete izabrati stavku i zatim kliknuti na dugme minus (-).
- Promena redosleda elemenata u listi prozora **Watches** može se uraditi tako što ćete izabrati stavku i zatim kliknuti ikone nagore ili nadole.

12.4 - Izračunavanje izraza

- Koristite **Evaluate Expression** da biste videli stanje promenljivih i objekata u aplikaciji, uključujući metode pozivanja na tim objektima.
- Da biste procenili izraz:

1. Kliknite na ikonu **Evaluate Expression Evaluate Expression Icon** ili izaberite **Run> Evaluate Expression**.
2. Pojaviće se prozor **Evaluate Code Fragment**. Takođe možete kliknuti desnim tasterom miša na bilo koju promenljivu i izabrati **Evaluate Expression**.
3. Unesite bilo koji Java izraz u gornje polje prozora **Evaluate Code Fragment** i kliknite **Evaluate**. Polje **Result** prikazuje rezultat tog izraza. Imajte na umu da rezultat koji dobijete od procene izraza bazira se na trenutnom stanju aplikacije.
4. U zavisnosti od vrednosti promenljivih u vašoj aplikaciji u vreme kada procenjujete izraze, možete dobiti različite rezultate.
5. Promenom vrednosti promenljivih u vašim izrazima takođe se menja trenutno pokrenuto stanje aplikacije.

12.5–Ostali alati za otkrivanje grešaka

- Android Studio i Android SDK sadrže brojne druge alate koji vam pomažu da pronađete i ispravite probleme u svom kodu:

1. Sistemski dnevnik (Logcat)

- ✓ Koristi klasu **Log** za slanje poruka u sistemski dnevnik Android OS koje možemo pregledati u programu Android Studio u prozoru **Logcat**.
- ✓ Poruke pomažu da razumemo tok izvršavanja prikupljanjem izveštaja o otklanjanju grešaka sistema dok smo u interakciji sa aplikacijom.
- ✓ Poruke ukazuju koji deo vaše aplikacije nije odradio svoj zadatak.

2. Praćenje i zapisivanje

- ✓ Omogućava da vidimo koliko vremena traje izvršavanje nekih metoda
- ✓ Da biste kreirali datoteke praćenja, uključite klasu **Debug** i pozovite jednu od metoda **startMethodTracing().**
- ✓ U pozivu navodimo osnovno ime za fajl praćenja koji OS generiše
- ✓ Da biste zaustavili praćenje, pozovite **stopMethodTracing()**.
- ✓ Metode pokreću i zaustavljaju praćenje na celoj virtualnoj mašini.
- ✓ Možete da nazovete **startMethodTracing()** u **onCreate()** metodi vaše aktivnosti i pozovete **stopMethodTracing ()** u **onDestroy ()** metodi

12.5–Ostali alati za otkrivanje grešaka

3. **Android Debug Bridge (ADB)**

✓ Alat za komandnu liniju koji vam omogućava da komunicirate sa instancom emulatora ili povezanim Android uređajem. Čine ga tri osnovne komponente: klijent, server i sistemska usluga (*daemon*)

4. **Hierarchy Viewer**

✓ Prikazuje u vizelnom obliku hijerahiju svih prikaza koji čine ekran aplikacije(**Layout View**) kao i uvećan prikaz elemenata ekrana (*Pixel PerfectView*). Sastoji se iz tri prikaza:

1. **Tree View** – hijerahijski dijagram prikaza koji su deo ekrana.
2. **Propertries View** – lista osobina izabranog elementa ekrana.
3. **Wire-frame View** – blok šema strukture ekrana (povezani elementi)

5. **Alati ugrađeni u Android OS (komande Linux OS)**

✓ ***Adb shell top*** - prikazuje sve akrivne aplikacije I njene resurse

✓ ***Adb shell ps*** – prikazuje listu svih aktivnih procesa u sistemu

✓ **GDB** (GNU project Debuger) – otklanja greške u C bibliotekama

12.5–Ostali alati za otkrivanje grešaka

6. **Android Profiler**
- ✓ pruža podatke u realnom vremenu za procesor, memoriju i mrežnu aktivnost vaše aplikacije.
- ✓ možete izvršiti praćenje metoda zasnovanih na uzorku kako biste izvršili pračenje koda, pregledali raspodelu memorije i pregledali detalje o mrežnim datotekama.

7. **Profiler CPU-a**
- ✓ pomaže da vidimo aktivnost CPU u aplikaciji i aktivnosti niti u realnom vremenu kao i da pratimo izvršavanje metoda, kako bi mogli de optimiziramo i ispravimo kod aplikacije.

8. **Mrežni Profiler**
- ✓ prikazuje aktivnost mreže u realnom vremenu, prikazuje primljene i poslate podatka, kao i trenutni broj veza. Ovo vam omogućava da ispitate koliko, kako i kada aplikacija prenosi podatke

9. **Trace View** - pomaže kod optimizovanja performansi aplikacje.

10. **APK Analyzer** – omogućuje da pregledamo sadržaj APK datoteke aplikacije, datoteku manifesta, resurse i DEX datoteke.

12.5–Ostali alati za otkrivanje grešaka

11. **Zapisivanje praćenja i datoteka AndroidManifest.xml**

- ✓ Možemo koristiti logovanje i praćenje da biste otkrili probleme u kodu
- ✓ Kada imate datoteku dnevnika praćenja (generisanu dodavanjem koda za praćenje u aplikaciju), možete učitati datoteke dnevnika u **Traceview**, koji prikazuju podatke dnevnika u dva prozora:
  1. **Prozor vremenske linije** opisuje kada svaka nit ili metoda počinje ili završava svoje izvršavanje.
  2. **Profil prozor** daje rezime onoga što se desilo unutar metode.
- ✓ Da bi se aplikacija mogla debugovati čak i kada se aplikacija pokreće na uređaju u korisničkom režimu, možete **postaviti android: debuggable** u **<application>** tagu u **AndroidManifest.xml** na "**true**".
- ✓ Podrazumevano, **debuggable** vrednost je postavljena na "**false**".
- ✓ Možete kreirati i konfigurisati **build** tipove u datoteci **build.gradle** na nivou modula unutar android {} bloka.
- ✓ Kada kreirate novi modul, Android Studio kreira tipove za debug i rezerviše ih za aplikaciju.
- ✓ Iako se tip **debug build** ne pojavljuje u konfiguracionoj datoteci, Android Studio ga konfiguriše i postavlja **debuggable** na “true".

12.5-Ostali alati za otkrivanje grešaka

- Kada pripremite aplikaciju za objavljivanje, morate ukloniti sav dodatni kod u izvornim datotekama koje ste napisali u svrhu testiranja.
- Pored pripreme samog koda, potrebno je ispuniti još nekoliko zadataka kako bi vaša aplikacija bila spremna za objavljivanje.
  - ✓ Uklanjanje izveštaja o evidenciji.
  - ✓ Uklonite sve pozive koji prikazuju poruke **Tost**.
  - ✓ Onemogućite otklanjanje grešaka u datoteci **AndroidManifest.xml** tako što ćete ukloniti android: atribut **debuggable** iz **<application>** taga, ili podesiti **android: debuggable** to "false".
  - ✓ Uklonite sve pozive za otklanjanje grešaka iz datoteka izvornog koda kao što su **startMethodTracing()** i **stopMethodTracing()**.
  - ✓ Sve ove promene napravljene za otklanjanje grešaka **moraju biti uklonjene iz koda pre izdavanja** jer mogu da utiču na proizvodni kod izvršavanja i performanse aplikacije.

12.6 – Napredne opcije Android OS

- Upotreba ekrana osetljivog na dodir
- Rad sa senzorima (ekološki, pozicioni i senzori pokreta)
- Grafika i animacija
- Multimedija
- Android biometrijska provera identiteta
- Telefonija i mrežni rad
- Rad sa Internetom i Web servisima
- Lokacijski servis i Google Maps Android API
- Priprema aplikacije za izdavanje (Play Store)
- Opcije Backend as a Servicea (BaaS-a)
- Gradle sistem za upravljanje složenim projektima u Android studiju
- Programski jezik KOTLIN
- Android Jetpack – (Android Studio, Android Architecture Componentsa i Android Support Library)

12.6 – Korisni sajtovi

1. **https://developer.android.com/**
2. **https://developer.android.com/guide/**
3. **https://developers.google.com/training/android/**
4. **https://www.androidauthority.com/android-development/**
5. **https://androidweekly.net/**
6. **https://android-developers.googleblog.com/**
7. **https://android-developers.googleblog.com/**
8. **https://codelabs.developers.google.com/advanced-android-training/**
9. **https://www.class-central.com/subject/android-development**
10. **https://www.codementor.io/collections/learn-android-development-online-bwba0mlle**

Hvala na pažnji !!!

Pitanja

? ? ?