

MODUL PRAKTIKUM

DEBUGGER ANDROID STUDIO



disusun oleh :

Nama : Aldi Bagus Prasetyo

Npm : 18313053

Kelas : TI 18B

FAKULTAS TEKNIK DAN ILMU KOMPUTER
TEKNOLOGI INFORMASI

2020

Materi:

- Pengantar
- Tentang men-debug
- Menjalankan debugger
- Menggunakan Breakpoint
- Merunut kode
- Menampilkan bingkai tumpukan eksekusi
- Memeriksa dan memodifikasi variabel-variabel
- Menyetel watches
- Mengevaluasi ekspresi
- Alat (bantu) lainnya untuk men-debug
- Perekaman pelacakan ke log dan manifes Android
- Praktik terkait
- Ketahui selengkapnya

Dalam bab ini Anda akan mempelajari tentang men-debug aplikasi di Android Studio.

Tentang men-debug

Men-debug adalah proses menemukan dan memperbaiki kesalahan (bug) atau perilaku tak terduga dalam kode Anda. Semua kode memiliki bug, mulai dari perilaku yang salah dalam aplikasi, perilaku yang mengonsumsi memori atau sumber daya jaringan secara berlebihan, hingga aplikasi yang membeku atau mogok.

Bug bisa terjadi karena berbagai alasan:

- Kesalahan dalam desain atau implementasi Anda.
- Batasan (bug) kerangka kerja Android.
- Tidak adanya persyaratan atau asumsi tentang cara kerja aplikasi yang seharusnya.
- Batasan (atau bug) perangkat

Gunakan kemampuan men-debug, menguji, dan membuat profil di Android Studio untuk membantu Anda mengulangi kembali, menemukan, dan mengatasi semua masalah ini. Kemampuan itu antara lain:


- Monitor Android (logcat)
- Debugger Android Studio
- Kerangka kerja pengujian seperti JUnit atau Espresso
- Dalvik Debug Monitor Server (DDMS), untuk melacak penggunaan sumber daya

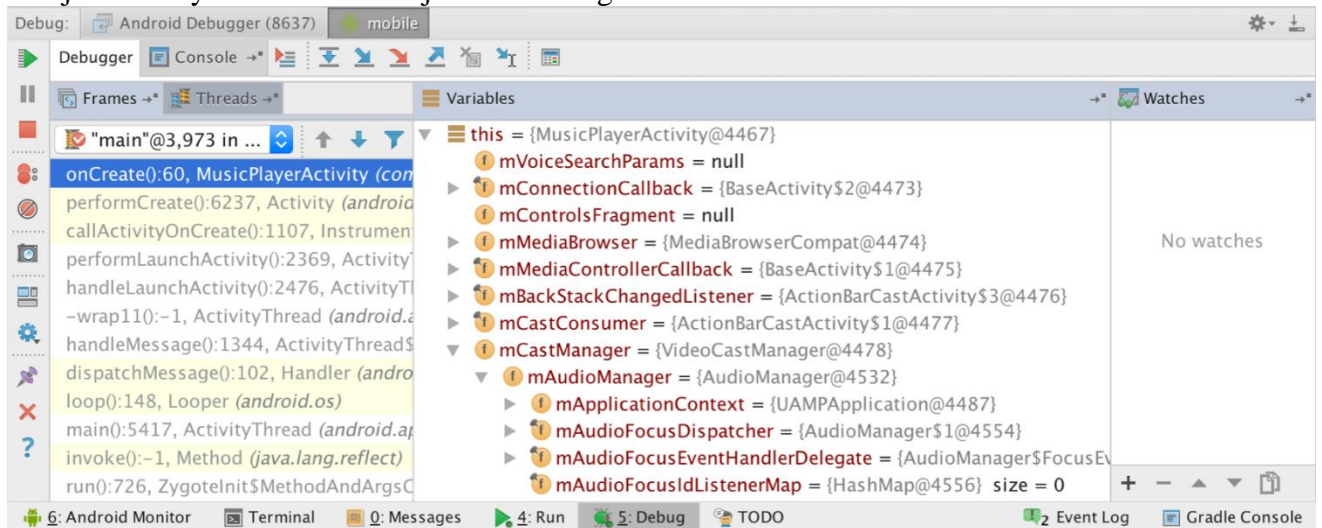
Dalam bab ini Anda akan mempelajari cara men-debug aplikasi dengan debugger Android Studio, menyetel dan menampilkan breakpoint, merunut kode, dan memeriksa variabel-variabel.

Menjalankan debugger

Menjalankan aplikasi dalam mode debug sama seperti menjalankan aplikasi seperti biasanya. Anda bisa menjalankan aplikasi dalam mode debug, atau melampirkan debugger ke aplikasi yang sudah berjalan.


Menjalankan aplikasi Anda dalam mode debug

Untuk mulai men-debug, klik Debug  di bilah alat. Android Studio akan membangun APK, menandainya dengan kunci debug, memasangnya pada perangkat yang Anda pilih, kemudian menjalankannya dan membuka jendela Debug.



Men-debug aplikasi yang berjalan


Jika aplikasi Anda sudah berjalan pada perangkat atau emulator, mulailah men-debug aplikasi itu dengan langkah-langkah ini:

1. Pilih **Run > Attach debugger to Android process** atau klik ikon **Attach**  di bilah alat.
2. Dalam dialog **Choose Process**, pilih proses yang ingin Anda lampirkan debugger.

Secara default, debugger menampilkan perangkat dan proses aplikasi untuk proyek saat ini, serta perangkat keras atau perangkat virtual yang terhubung pada komputer Anda. Pilih **Show all processes** untuk menampilkan semua proses di semua perangkat.

3. Klik **OK**. Jendela Debug akan muncul seperti sebelumnya.

Melanjutkan atau Menghentikan Debug

Untuk melanjutkan eksekusi aplikasi setelah men-debug, pilih **Run > Resume Program** atau klik ikon Resume .

Untuk berhenti men-debug aplikasi Anda, pilih **Run > Stop** atau klik ikon Stop  di bilah alat.

Menggunakan Breakpoint

Breakpoint adalah tempat dalam kode yang Anda inginkan untuk menghentikan sementara eksekusi normal aplikasi untuk melakukan tindakan lainnya seperti memeriksa variabel-variabel atau mengevaluasi ekspresi, atau mengeksekusi kode setiap baris untuk menentukan penyebab kesalahan waktu proses.

Menambahkan breakpoint

Untuk menambahkan breakpoint ke sebuah baris dalam kode Anda, gunakan langkah-langkah ini:


1. Carilah baris kode yang diinginkan untuk menjadi tempat Anda menghentikan sementara eksekusi.
2. Klik di gutter kiri jendela editor pada baris tersebut, di sebelah nomor baris. Titik merah muncul pada baris itu, yang menunjukkan sebuah breakpoint.

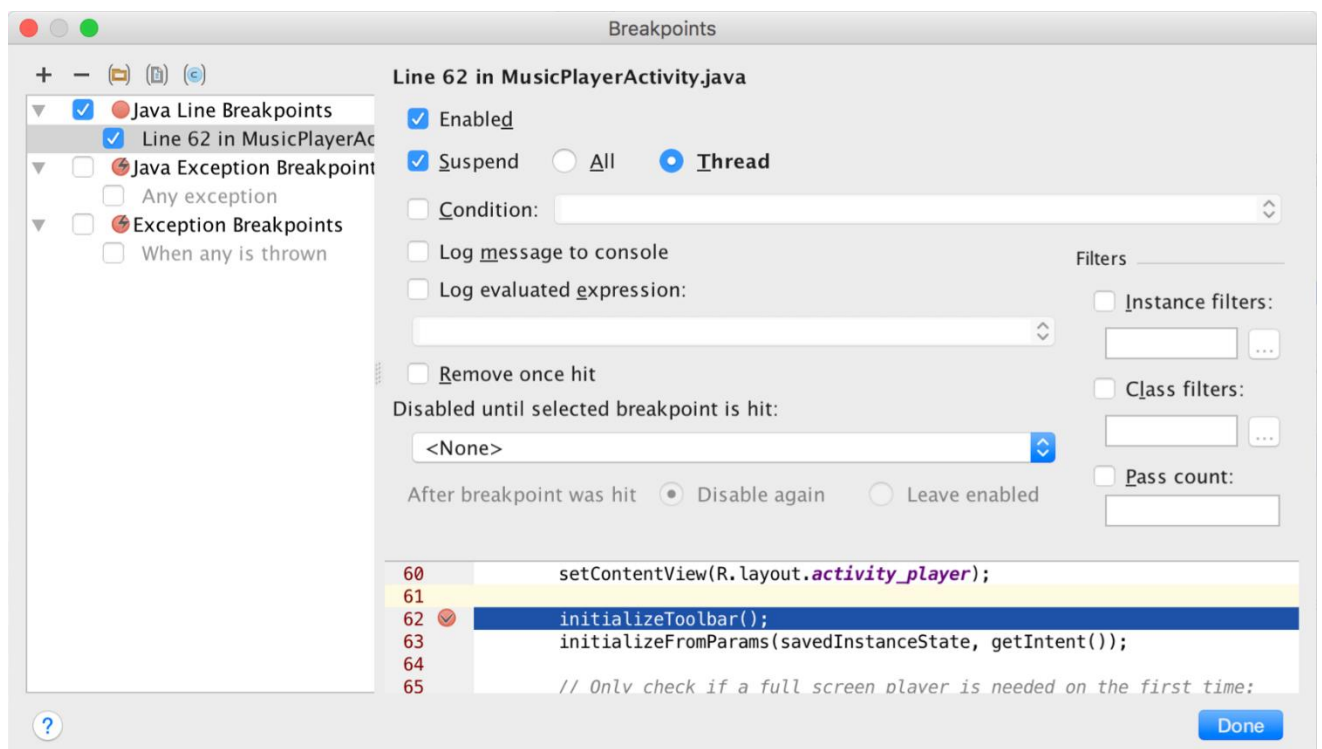
Anda juga bisa menggunakan **Run > Toggle Line Breakpoint** atau Control-F8 (Command-F8 di OS X) untuk menyetel breakpoint pada baris.

Jika aplikasi sudah berjalan, Anda tidak perlu memperbaruinya untuk menambahkan breakpoint.

Bila eksekusi kode sudah mencapai breakpoint, Android Studio menghentikan sementara eksekusi aplikasi Anda. Selanjutnya Anda bisa menggunakan alat (bantu) dalam debugger Android untuk menampilkan keadaan aplikasi dan men-debug aplikasi saat berjalan.

Menampilkan dan mengonfigurasi breakpoint

Untuk menampilkan semua breakpoint yang telah disetel dan mengonfigurasi setelan breakpoint, klik ikon View Breakpoints  di tepi kiri jendela debugger. Jendela Breakpoints akan muncul.



Dalam jendela ini semua breakpoint yang telah Anda setel muncul di panel kiri, dan Anda bisa mengaktifkan atau menonaktifkan setiap breakpoint dengan kotak centang. Jika breakpoint dinonaktifkan, Android Studio tidak akan menghentikan sementara aplikasi Anda bila eksekusi mencapai breakpoint tersebut.

Pilih breakpoint dari daftar untuk mengonfigurasi setelannya. Anda bisa mengonfigurasi breakpoint agar dinonaktifkan di awal dan meminta sistem mengaktifkannya setelah breakpoint yang berbeda ditemukan. Anda juga bisa mengonfigurasi apakah breakpoint harus dinonaktifkan setelah tercapai.

Untuk menyetel breakpoint bagi suatu pengecualian, pilih Exception Breakpoints dalam daftar breakpoint.

Menonaktifkan (membisukan) semua breakpoint

Penonaktifan breakpoint memungkinkan Anda mem"bisukan" sementara breakpoint tanpa membuangnya dari kode. Jika breakpoint dibuang semuanya, Anda juga akan kehilangan semua ketentuan atau fitur lainnya yang telah dibuat untuk breakpoint tersebut, sehingga menonaktifkannya bisa menjadi pilihan yang lebih baik.

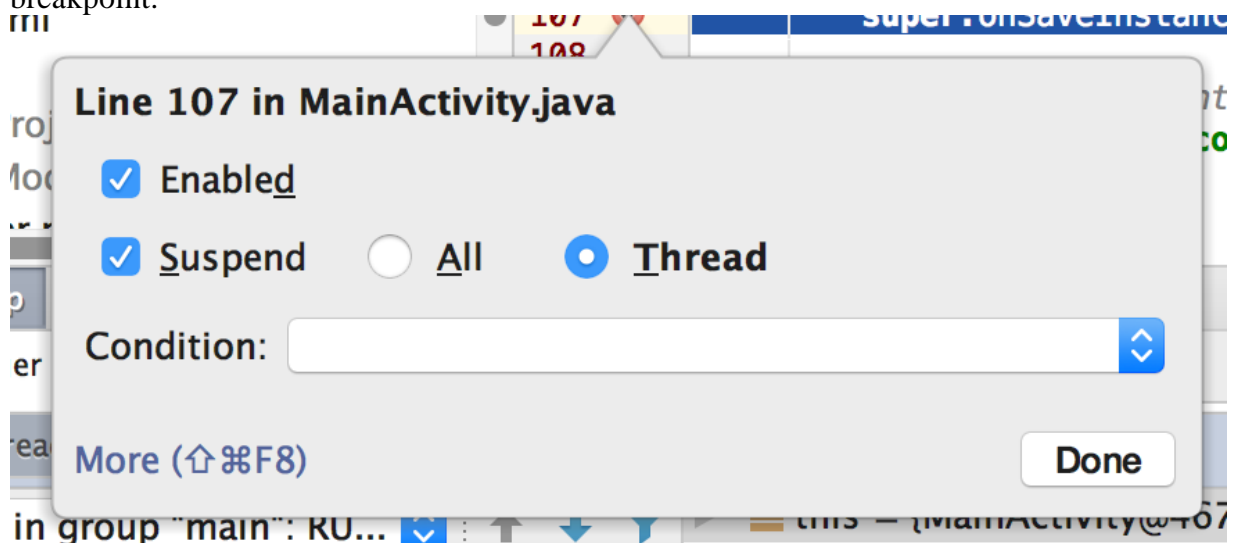
Untuk membisukan semua breakpoint, klik ikon **Mute Breakpoints** . Klik lagi ikon tersebut untuk mengaktifkan (membunyikan) semua breakpoint.

Menggunakan breakpoint bersyarat

Breakpoint bersyarat adalah breakpoint yang hanya menghentikan eksekusi aplikasi Anda jika ketentuan pengujian terpenuhi. Untuk mendefinisikan pengujian breakpoint bersyarat, gunakan langkah-langkah ini:

1. Klik-kanan pada ikon breakpoint, dan masukkan pengujian dalam bidang Condition.

Anda juga bisa menggunakan jendela Breakpoints untuk memasukkan ketentuan breakpoint.



Pengujian yang dimasukkan dalam bidang ini bisa berupa ekspresi Java asalkan mengembalikan nilai boolean. Anda bisa menggunakan nama variabel dari aplikasi sebagai bagian dari ekspresi.

2. Jalankan aplikasi dalam mode debug. Eksekusi aplikasi Anda akan berhenti di breakpoint bersyarat, jika syarat bernilai true.


Merunut kode

Setelah eksekusi aplikasi berhenti karena telah mencapai breakpoint, Anda bisa mengeksekusi kode dari titik itu sebanyak satu baris dengan fungsi Step Over, Step Into, dan Step Out.

Untuk menggunakan salah satu fungsi langkah:

1. Mulailah men-debug aplikasi Anda. Hentikan sementara eksekusi aplikasi dengan breakpoint.

Eksekusi aplikasi akan berhenti, dan debugger menampilkan keadaan aplikasi saat ini. Baris saat ini akan disorot dalam kode Anda.

2. Klik ikon **Step Over** , pilih **Run > Step Over**, atau tekan F8.


Step Over mengeksekusi baris kode berikutnya di kelas dan metode saat ini, yang mengeksekusi semua panggilan metode pada baris itu dan sisanya dalam file yang sama.

3. Klik ikon **Step Into** , pilih **Run > Step Into**, atau tekan F7.

Step Into akan melompat ke eksekusi panggilan metode pada baris saat ini (dibandingkan dengan hanya mengeksekusi metode tersebut dan sisanya pada baris yang sama). Tampilan **Frame** (yang akan Anda pelajari di bagian berikutnya) akan diperbarui untuk menampilkan bingkai tumpukan baru (metode baru). Jika panggilan metode dimuat dalam kelas lain, file untuk kelas itu akan dibuka dan baris saat ini dalam file itu akan disorot. Anda bisa terus merunut baris-baris dalam panggilan metode baru ini, atau merunut lebih dalam ke metode lainnya.

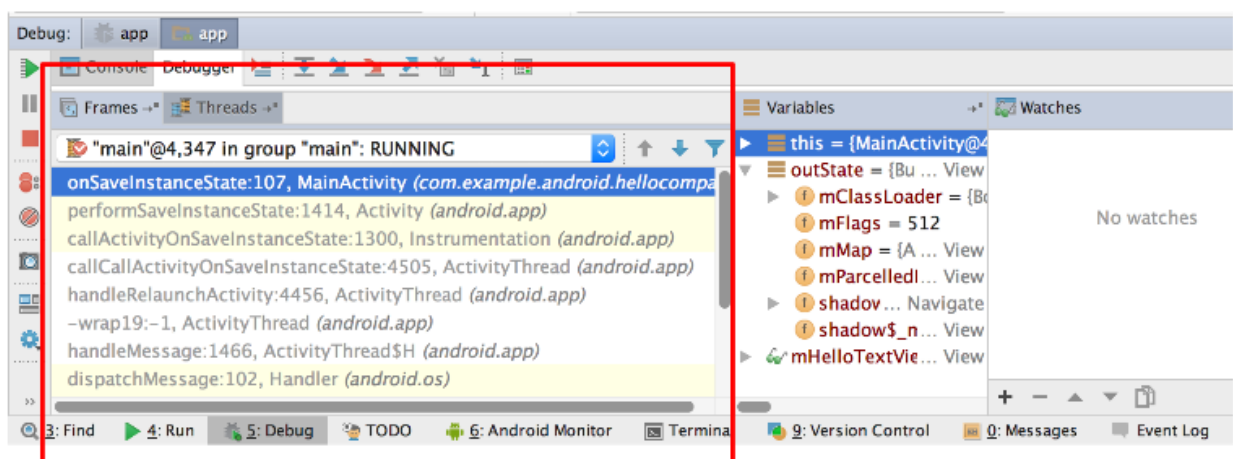
4. Klik ikon **Step Out** , pilih **Run > Step Out**, atau tekan Shift-F8.

Step Out akan menyelesaikan eksekusi metode saat ini dan mengembalikan ke titik pemanggilan metode semula.

5. Untuk melanjutkan eksekusi normal aplikasi, pilih **Run > Resume Program** atau klik ikon Resume  .

Menampilkan bingkai tumpukan eksekusi

Tampilan **Frames** pada jendela debugger memungkinkan Anda memeriksa tumpukan eksekusi dan bingkai tertentu yang menyebabkan breakpoint saat ini tercapai.




Tumpukan eksekusi menampilkan semua kelas dan metode (bingkai) yang sedang dieksekusi hingga titik ini dalam aplikasi, dalam urutan terbalik (bingkai terbaru lebih dahulu). Saat eksekusi bingkai tertentu selesai, bingkai tersebut akan dimunculkan dari tumpukan dan eksekusi kembali ke bingkai berikutnya.

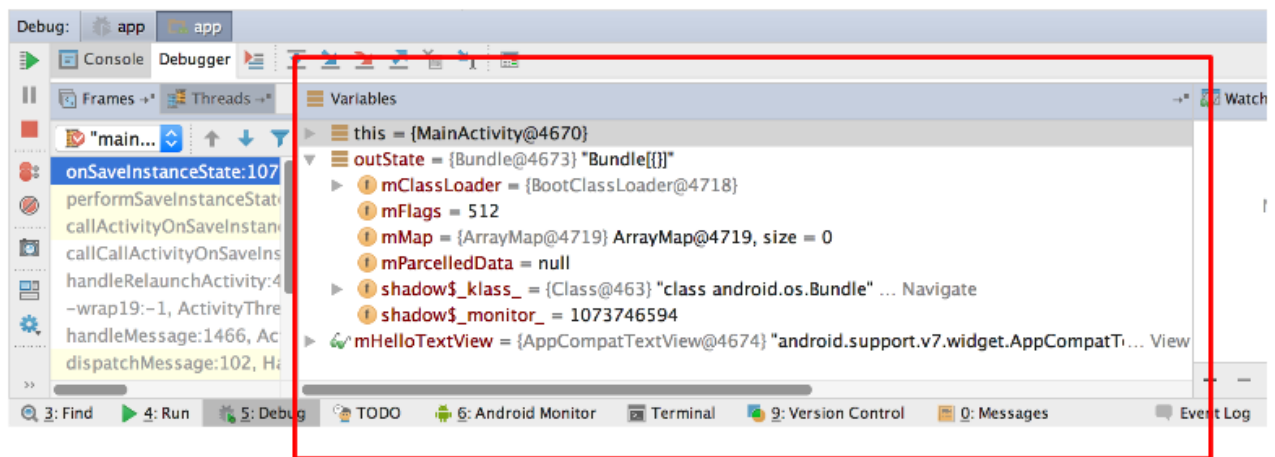
Mengeklik baris untuk bingkai dalam tampilan Frames akan membuka sumber terkait dalam editor dan menyoroti baris tempat bingkai dieksekusi pada awalnya. Tampilan Variables dan Watches juga diperbarui untuk mencerminkan keadaan lingkungan eksekusi ketika bingkai itu terakhir dimasuki.

Memeriksa dan memodifikasi variabel-variabel

Tampilan **Variables** pada jendela debugger memungkinkan Anda memeriksa variabel yang tersedia di bingkai tumpukan saat ini ketika sistem menghentikan aplikasi pada breakpoint. Variabel-variabel yang menyimpan objek atau kumpulan seperti larik bisa diluaskan untuk menampilkan komponennya.

Panel **Variable** juga memungkinkan Anda mengevaluasi ekspresi dengan cepat menggunakan variabel dan/atau metode statis yang tersedia dalam bingkai yang dipilih.

Jika tampilan Variables tidak terlihat, klik ikon **Restore Variables View** .



Untuk memodifikasi variabel-variabel di aplikasi Anda saat dijalankan:

1. Klik-kanan sutau variabel di tampilan Variable, dan pilih **Set Value**. Anda juga bisa menggunakan F2.
2. Masukkan nilai baru untuk variabel, dan tekan Return.

Nilai yang Anda masukkan harus sesuai tipenya dengan variabel tersebut, atau Android Studio akan mengembalikan kesalahan "type mismatch".

Menyetel watches

Tampilan **Watches** menyediakan fungsionalitas yang sama dengan tampilan Variables hanya saja ekspresi yang ditambahkan ke panel Watches akan bertahan di antara sesi debug. Tambahkan watches untuk variabel-variabel dan bidang yang sering Anda akses atau yang menyediakan keadaan yang berguna untuk sesi debug saat ini.

Untuk menggunakan watches:

1. Mulailah men-debug aplikasi Anda.
2. Di panel Watches, klik tombol plus (+).


Dalam kotak teks yang muncul, ketikkan nama variabel atau ekspresi yang ingin Anda amati, kemudian tekan Enter.

Buang item dari daftar Watches dengan memilih item tersebut kemudian mengeklik tombol minus (-).

Ubah urutan elemen dalam daftar Watches dengan memilih sebuah item kemudian mengeklik ikon naik atau turun.

Mengevaluasi ekspresi

Gunakan Evaluate Expression untuk menjelajahi keadaan variabel-variabel dan objek dalam aplikasi Anda, termasuk metode panggilan pada objek itu. Untuk mengevaluasi ekspresi:

1. Klik ikon Evaluate Expression , atau pilih **Run > Evaluate Expression**. Anda juga bisa mengeklik-kanan pada suatu variabel dan memilih Evaluate Expression.

Jendela Evaluate Expression akan muncul.

2. Masukkan suatu ekspresi ke dalam jendela Expression dan klik **Evaluate**.

Jendela Evaluate Expression akan memperbarui hasil eksekusi. Perhatikan, hasil yang Anda dapat dari mengevaluasi ekspresi adalah berdasarkan pada keadaan aplikasi saat ini. Bergantung pada nilai variabel-variabel dalam aplikasi saat mengevaluasi ekspresi, Anda mungkin mendapatkan hasil yang berbeda. Mengubah nilai variabel-variabel dalam ekspresi juga akan mengubah keadaan aplikasi yang berjalan saat ini.

Alat (bantu) lainnya untuk men-debug

Android Studio dan Android SDK menyertakan sejumlah alat (bantu) lainnya untuk membantu Anda menemukan dan mengoreksi masalah dalam kode. Alat (bantu) ini antara lain:

- Log sistem (logcat). Seperti yang telah dipelajari dalam pelajaran sebelumnya, Anda bisa menggunakan kelas Log untuk mengirim pesan ke log sistem Android, dan menampilkan pesan itu di Android Studio.
 - ❖ Untuk menulis pesan log di kode Anda, gunakan kelas Log. Pesan log membantu memahami alur eksekusi dengan mengumpulkan keluaran debug sistem saat Anda berinteraksi dengan aplikasi. Pesan log bisa memberi tahu Anda tentang bagian aplikasi yang gagal. Untuk informasi selengkapnya tentang pembuatan log, lihat Membaca dan Menulis Log.
- Melacak dan Membuat Log. Analisis jejak memungkinkan Anda melihat banyaknya waktu yang dihabiskan dalam metode tertentu, dan metode yang membutuhkan waktu terlalu lama.
 - ❖ Untuk membuat file pelacakan, sertakan kelas Debug dan panggil salah satu metode `startMethodTracing()`. Dalam panggilan tersebut, tetapkan nama dasar untuk file pelacakan yang dihasilkan sistem. Untuk menghentikan pelacakan, panggil `stopMethodTracing()`. Semua metode ini memulai dan menghentikan pelacakan metode di seluruh mesin virtual. Misalnya, Anda bisa memanggil `startMethodTracing()` di metode `onCreate()` aktivitas, dan memanggil `stopMethodTracing()` di metode `onDestroy()` aktivitas itu.
- Android Debug Bridge (ADB). ADB adalah alat (bantu) baris perintah yang memungkinkan Anda berkomunikasi dengan instance emulator atau perangkat berbasis Android yang terhubung.

- Dalvik Debug Monitor Server (DDMS). Alat (bantu) DDMS menyediakan layanan penerusan porta, tangkapan layar, informasi thread dan heap, logcat, proses, dan informasi keadaan radio, panggilan masuk dan spoofing SMS, spoofing data lokasi, dan lainnya.
- Monitor CPU dan memori. Android Studio menyertakan sejumlah monitor untuk membantu memvisualisasikan perilaku dan kinerja aplikasi Anda.
- Tangkapan layar dan rekaman video.

Perekaman pelacakan ke log dan manifes Android

Ada beberapa jenis debug yang tersedia untuk Anda di luar setelan breakpoint dan peruntutan kode. Anda juga bisa menggunakan pembuatan log dan pelacakan untuk menemukan masalah pada kode. Bila memiliki file log jejak (dihasilkan dengan menambahkan kode pelacakan ke aplikasi Anda atau melalui DDMS), Anda bisa memuat file log di Traceview, yang akan menampilkan data log dalam dua panel:

- Panel timeline -- menjelaskan waktu memulai dan menghentikan setiap thread dan metode
- Panel profil -- menyediakan rangkuman tentang hal yang terjadi di dalam metode

Demikian juga, Anda bisa menyetel `android:debuggable` di tag `<application>` Manifes Android ke "true", yang menyetel apakah aplikasi bisa di-debug atau tidak, bahkan saat dijalankan pada perangkat dalam mode pengguna. Secara default, nilai ini disetel ke "false".

Anda bisa membuat dan mengonfigurasi tipe pembangunan dalam file `build.gradle` tingkat modul di dalam blok `{ }` `android`. Bila membuat modul baru, Android Studio secara otomatis membuatkan tipe pembangunan rilis dan debug untuk Anda. Meskipun tipe pembangunan debug tidak muncul dalam file konfigurasi pembangunan, Android Studio akan mengonfigurasinya dengan `debuggable true`. Hal ini memungkinkan Anda men-debug aplikasi pada perangkat Android yang aman dan mengonfigurasi penandatanganan APK dengan keystore debug generik. Anda bisa menambahkan tipe pembangunan debug ke konfigurasi jika ingin menambahkan atau mengubah setelan tertentu.

Semua perubahan yang dibuat untuk men-debug harus dibuang dari kode sebelum rilis karena bisa memengaruhi eksekusi dan kinerja kode produksi.

Saat mempersiapkan aplikasi untuk rilis, Anda harus membuang semua kode ekstra di file sumber yang ditulis untuk keperluan pengujian.

Selain mempersiapkan kode itu sendiri, ada beberapa tugas lain yang perlu diselesaikan agar aplikasi Anda siap dipublikasikan. Tugas tersebut antara lain:

- Membuang pernyataan log
- Membuang panggilan untuk menampilkan Toast
- Menonaktifkan debug di manifes Android dengan:
 - ❖ Membuang atribut `android:debuggable` dari tag `<application>`
 - ❖ Atau menyetel atribut `android:debuggable` ke false

Buang semua panggilan pelacakan debug dari file kode sumber Anda seperti `startMethodTracing()` dan `stopMethodTracing()`.