BAB II
LANDASAN TEORI

2.1. Peribahasa

2.1.1. Pengertian Peribahasa

Peribahasa adalah ayat atau kelompok kata yang mempunyai susunan yang tetap dan mengandung pengertian tertentu, bidal, pepatah. Beberapa peribahasa merupakan perumpamaan yaitu perbandingan makna yang sangat jelas karena ia didahului oleh perkataan seolah-olah, ibarat, bak, seperti, laksana, macam, bagai dan umpama [7].

2.1.2. Pengertian Peribahasa Sunda

Menurut Satjadibrata (1945) dalam Rosidi (2005:5) paribasa (paripaos) merupakan kata-kata yang disusun menjadi ungkapan ucapan yang memiliki arti pengalaman hidup atau menjadi petuah. Selanjutnya Gandasudirdja (1977: 80) menjelaskan bahwa paribasa merupakan ungkapan yang sudah tetap susunannya dan mengandung arti pengalaman hidup atau menjadi petuah yang susunannya sudah ditetapkan oleh nenek moyang, jika diubah susunannya tentu saja artinya pun akan berubah. Ditinjau dari ilmu bahasa, Prawirasumantri (1973: 39) menjelaskan bahwa paribasa dalam ilmu bahasa 116 Patanjala Vol. 7 No. 1 Maret 2015: 113 - 130 merupakan perbandingan yang sudah menjadi perlambang tindakan dalam membentuk satu ungkapan (susunan kata yang sudah jelas polanya, sudah jelas bunyinya, dan sudah tentu bagaimana cara mengukapkannya). Begitu juga menurut Sudrayat (2003: 99) paribasa merupakan ungkapan dalam bentuk kalimat (klausa) yang kata-katanya sudah tentu, dan maksudnya sudah jelas, biasanya mengandung arti perbandingan atau siloka tindakan hidup manusia. Menurut Tamsyah (1994: 9-10) ada beberapa ciri utama yang bisa membatasi antara paribasa dan kalimat lain, di antaranya: [6]

a. Paribasa sifatnya membandingkan, mengumpamakan;
b. Paribasa merupakan ungkapan yang tidak memiliki arti yang sebenarnya
c. Paribasa merupakan bentuk kalimat (klausa) yang sangat dekat pada hati yang mengungkapkannya, dan;

d. Paribasa tidak bisa diubah, dikurangi, dilebihkan, atau diperhalus kata-katanya, karena sudah berupa pakeman.

2.2. Pengertian Android

Android merupakan suatu sistem operasi dengan basis linux yang khusus dirancang untuk perangkat layar sentuh seperti *smartphone, tablet* dan sejenisnya. Awalnya Android, inc mengembangkan sistem operasi Android dengan adanya dukungan finansial dari Google yang kemudian dibeli oleh Google itu sendiri pada tahun 2005. Pada tahun 2007 *Open Handset Allieance,* perusahaan-perusahaan software, hardware, dan telekomunikasi didirak bersamaan dengan rilisnya sistem operasi Android untuk memajukan standar terbuka perangkat seluler. Ponsel android pertama kali mulai dijual pada bulan oktober 2008.

2.2.1. Sejarah Android

Android, Inc. Didirikan di Palo Alto, California, Pada bulan oktober 2003 oleh Andy Rubin (Pendiri Danger), Rich Milner (pendiri Wildfire Cimmunication, Inc.), Nick Sears (mantan Vp T-Mobile) dan Chris White (kepala desain dan pengembangan antarmuka WebTV) untuk mengembangkan perangkat smartphone yang lebih sadar akan referensi dan lokasi penggunanya. Tujuan dikembangkan sistem operasi Android adalah untuk pengembangan sebuah sistem operasi yang canggih untuk kamera digital, namun disadari bahwa pasar untuk perangkat tersebut tidak cukup besar dan pengembangan Android lalu dialihkan ke pasar smartphone untuk menyaingi Symbian dan Windows Mobile.

Pada tanggal 17 agustus 2005,Google mengakuisisi Android Inc dan menjadikannya sebagai anak perusahaan yang dimiliki oleh Google. Setelah diakuisisi oleh Google Miner, White dan Rubin selaku Pendiri Android Inc tetap bekerja di perusahaan. Setelah itu banyak anggapan yang menyatakan bahwa Google telah berencana untuk memasuki pasar telepon seluler dengan tindakannya ini.

Hingga bulan desember 2006 spekulasi tentang niat Google untuk memasuki pasar komunikasi seluler terus berkembang.

Open Handset Alliance (OHA) didirikan Pada tanggal 5 November 2007. OHA adalah konsorsium dari perusahaan-perusahaan teknologi seperti Google,

produsen perangkat seluler seperti HTC, Sony dan Samsung, operator nirkabel seperti Sprint Nextel dan T-Mobile, serta produsen chipset seperti Qualcomm dan Texas Instruments. OHA sendiri bertujuan untuk mengembangkan standar terbuka bagi perangkat seluler Pada 22 Oktober 2008 yang juga telepon seluler komersial pertama yang menggunakan sistem operasi Android adalah HTC Dream.

Google merilis seri Nexus yaitu perangkat telepon pintar dan tablet dengan sistem operasi Android pada tahun 2010 yang diproduksi oleh mitra produsen telepon seluler seperti HTC, LG, dan Samsung. HTC bekerja sama dengan Google dalam merilis smartphone Nexus pertama, yaitu Nexus One.

Android telah melakukan sejumlah pembaruan untuk meningkatkan kinerja sistem operasi Android, menambahkan fitur baru, dan memperbaiki bug yang terdapat pada versi sebelumnya Sejak tahun 2008. Setiap versi utama yang dirilis dinamakan secara alfabetis berdasarkan nama-nama makanan pencuci mulut(desert) atau makanan ringan bergula; misalnya, versi 1.5 bernama Cupcake, yang kemudian diikuti oleh versi 1.6 Donut. Versi terbaru adalah 5.0 Lollipop, yang dirilis pada 15 Oktober 2014.

2.2.2. Arsitektur Android

Android adalah tumpukan perangkat lunak berbasis Linux sumber terbuka yang dibuat untuk berbagai perangkat dan faktor bentuk. Diagram berikut menunjukkan komponen besar dari platform Android.

Gambar 2.1 Arsitektur Android

1. **Linux Kernel**

Fondasi platform Android adalah kernel Linux. Sebagai contoh, Android Runtime (ART) bergantung pada kernel Linux untuk fungsionalitas dasar seperti *threading* dan manajemen memori tingkat rendah. Menggunakan kernel Linux memungkinkan Android untuk memanfaatkan fitur keamanan inti dan memungkinkan produsen perangkat untuk mengembangkan driver perangkat keras untuk kernel yang cukup dikenal.

1. Hardware Abstraction Layer (Hal)

Hardware Abstraction Layer (HAL) menyediakan antarmuka standar yang mengekspos kemampuan perangkat keras di perangkat ke kerangka kerja Java API yang lebih tinggi. HAL terdiri atas beberapa modul pustaka, masing-masing mengimplementasikan antarmuka untuk komponen perangkat keras tertentu, seperti modul kamera atau bluetooth. Bila API kerangka kerja melakukan panggilan untuk mengakses perangkat keras, sistem Android memuat modul pustaka untuk komponen perangkat keras tersebut.

2. Android Runtime

Untuk perangkat yang menjalankan Android versi 5.0 (API level 21) atau yang lebih tinggi, setiap aplikasi menjalankan proses masing-masing dengan tahap Android Runtime (ART). ART ditulis guna menjalankan beberapa mesin virtual pada perangkat bermemori rendah dengan mengeksekusi file DEX, format bytecode yang didesain khusus untuk Android yang dioptimalkan untuk footprint memori minimal. Buat rantai aplikasi, misalnya Jack, mengumpulkan sumber Java ke *bytecode* DEX, yang dapat berjalan pada platform Android. Beberapa fitur utama ART mencakup:

- Kompilasi mendahului waktu (AOT) dan tepat waktu (JIT)
- Pengumpulan sampah (GC) yang dioptimalkan
- Dukungan debug yang lebih baik, mencakup profiler sampling terpisah, pengecualian diagnostik mendetail dan laporan kerusakan dan kemampuan untuk mengatur titik pantau guna memantau bidang tertentu.

  Sebelum ke Android versi 5.0 (API level 21), Dalvik adalah waktu proses Android. Jika aplikasi Anda berjalan baik pada ART, semestinya berfungsi baik juga pada Dalvik, tetapi mungkin tidak sebaliknya. Android juga menyertakan serangkaian pustaka waktu proses inti yang menyediakan sebagian besar fungsionalitas bahasa pemrograman Java, termasuk beberapa fitur bahasa Java 8, yang digunakan kerangka kerja Java API.

3. Pustaka C/C++ Asli

Banyak komponen dan layanan sistem Android inti seperti ART dan HAL dibuat dari kode asli yang memerlukan pustaka asli yang tertulis dalam C dan C++.

Platform Android memungkinkan kerangka kerja Java API mengekspos fungsionalitas beberapa pustaka asli pada aplikasi. Misalnya, Anda bisa mengakses OpenGL ES melalui kerangka kerja Java OpenGL API Android guna menambahkan dukungan untuk menggambar dan memanipulasi grafik 2D dan 3D pada aplikasi Anda. Jika Anda mengembangkan aplikasi yang memerlukan kode C atau C++, Anda bisa menggunakan Android NDK untuk mengakses beberapa pustaka platform asli langsung dari kode asli.

4. Kerangka Kerja Java API

Keseluruhan rangkaian fitur pada Android OS tersedia untuk Anda melalui API yang ditulis dalam bahasa Java. API ini membentuk elemen dasar yang Anda perlukan untuk membuat aplikasi Android dengan menyederhanakan penggunaan kembali inti, komponen dan layanan sistem modular, yang menyertakan berikut ini:

- Tampilan Sistem yang kaya dan luas bisa Anda gunakan untuk membuat UI aplikasi, termasuk daftar, kisi, kotak teks, tombol, dan bahkan browser web yang dapat disematkan.
- Pengelola Sumber Daya, memberikan akses ke sumber daya bukan kode seperti string yang dilokalkan, grafik, dan file layout.
- Pengelola Notifikasi yang mengaktifkan semua aplikasi guna menampilkan lansiran khusus pada bilah status.
- Pengelola Aktivitas yang mengelola daur hidup aplikasi dan memberikan back-stack navigasi yang umum.
- Penyedia Materi yang memungkinkan aplikasi mengakses data dari aplikasi lainnya, seperti aplikasi Kontak, atau untuk berbagi data milik sendiri.

  Developer memiliki akses penuh ke API kerangka kerja yang sama dengan yang digunakan oleh aplikasi sistem Android.

5. Aplikasi Sistem

Android dilengkapi dengan serangkaian aplikasi inti untuk email, perpesanan SMS, kalender, menjelajahi internet, kontak, dll. Aplikasi yang disertakan bersama platform tidak memiliki status khusus pada aplikasi yang ingin dipasang pengguna.

Jadi, aplikasi pihak ketiga dapat menjadi browser web utama, pengolah pesan SMS atau bahkan keyboard utama (beberapa pengecualian berlaku, seperti aplikasi Settings sistem). Aplikasi sistem berfungsi sebagai aplikasi untuk pengguna dan memberikan kemampuan kunci yang dapat diakses oleh developer dari aplikasi mereka sendiri. Misalnya, jika aplikasi Anda ingin mengirimkan pesan SMS, Anda tidak perlu membangun fungsionalitas tersebut sendiri—sebagai gantinya Anda bisa menjalankan aplikasi SMS mana saja yang telah dipasang guna mengirimkan pesan kepada penerima yang Anda tetapkan.

2.2.3. Siklus Hidup Android

Saat pengguna menavigasi, keluar, dan kembali ke aplikasi, Aktivitas akan muncul dalam transisi di aplikasi melalui berbagai status dalam siklus hidupnya. Kelas *Activity* menyediakan sejumlah panggilan balik yang memungkinkan aktivitas mengetahui bahwa suatu keadaan telah berubah bahwa sistem membuat, menghentikan, atau melanjutkan suatu aktivitas, atau menghancurkan proses di mana aktivitas berada.

Dalam metode callback siklus hidup, dapat mendeklarasikan bagaimana aktivitas berperilaku ketika pengguna meninggalkan dan masuk kembali ke aktivitas. Misalnya, jika membuat pemutar video streaming, dapat menjeda video dan memutuskan koneksi jaringan ketika pengguna beralih ke aplikasi lain. Ketika pengguna kembali, dapat menyambung kembali ke jaringan dan memungkinkan pengguna untuk melanjutkan video dari tempat yang sama. Dengan kata lain, setiap panggilan balik memungkinkan melakukan pekerjaan tertentu yang sesuai dengan perubahan kondisi yang diberikan. Melakukan pekerjaan yang tepat di waktu yang tepat dan menangani transisi dengan benar membuat aplikasi lebih kuat dan berkinerja. Misalnya, penerapan panggilan balik siklus hidup yang baik dapat membantu memastikan bahwa aplikasi menghindari: callback yang bagus dapat membantu memastikan aplikasi dihindari:

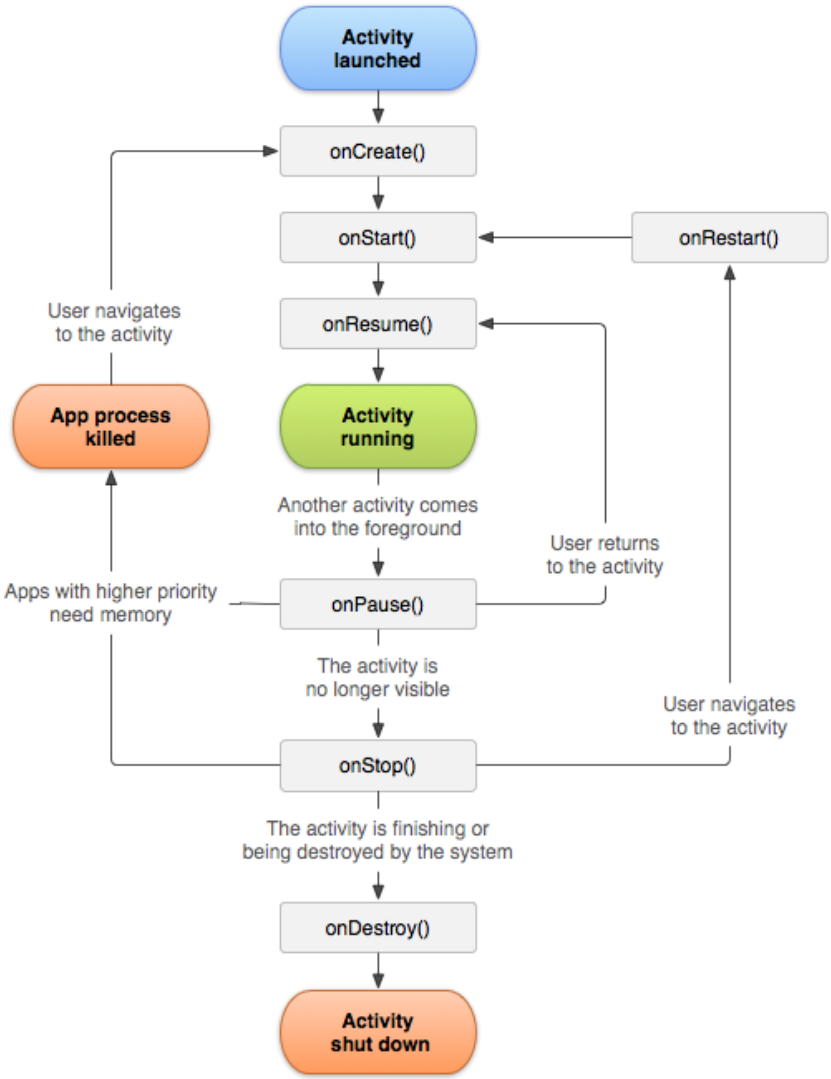
- *Crashing* jika pengguna menerima panggilan telepon atau beralih ke aplikasi lain saat menggunakan aplikasi.
- *Consuming* sumber daya sistem yang berharga saat pengguna tidak aktif menggunakannya

- *Losing* kemajuan pengguna jika mereka meninggalkan aplikasi dan kembali lagi nanti.
- *Crashing* atau kehilangan kemajuan pengguna saat layar berputar antara orientasi landscape dan portrait.

Dokumen ini menjelaskan siklus hidup aktivitas secara terperinci. Dokumen dimulai dengan menjelaskan paradigma siklus hidup. Selanjutnya, ini menjelaskan masing-masing panggilan balik: apa yang terjadi secara internal saat mereka mengeksekusi, dan apa yang harus Anda terapkan selama mereka. Ini kemudian secara singkat memperkenalkan hubungan antara keadaan aktivitas dan kerentanan suatu proses untuk dibunuh oleh sistem. Terakhir, ini membahas beberapa topik yang terkait dengan transisi antara status aktivitas.

2.2.4. Konsep Siklus Hidup

Untuk menavigasi transisi antara tahap siklus hidup aktivitas, kelas Activity menyediakan set inti enam panggilan balik yaitu onCreate (), onStart (), onResume (), onPause (), onStop (), dan onDestroy (). Sistem memanggil masing-masing panggilan balik ini saat aktivitas memasuki keadaan baru.



Gambar 2.2 Ilustrasi sederhana dari siklus hidup aktivitas

Ketika pengguna mulai meninggalkan aktivitas, sistem memanggil metode untuk membongkar aktivitas. Dalam beberapa kasus, pembongkaran ini hanya sebagian; aktivitas masih berada dalam memori (seperti ketika pengguna beralih ke aplikasi lain), dan masih dapat kembali ke latar depan. Jika pengguna kembali ke aktivitas itu, aktivitas dilanjutkan dari tempat pengguna tadi pergi. Kemungkinan sistem untuk membunuh proses tertentu bersama dengan aktivitas di dalamnya tergantung pada keadaan aktivitas pada saat itu. Status aktivitas dan ejeksi dari memori memberikan lebih banyak informasi tentang hubungan antara status dan kerentanan terhadap ejeksi.

Bergantung pada kerumitan aktivitas, mungkin tidak perlu menerapkan semua metode siklus hidup. Namun, penting untuk memahami masing-masing dan

menerapkannya yang memastikan aplikasi dapat berperilaku seperti yang diharapkan pengguna.

Bagian selanjutnya dari dokumen ini memberikan detail tentang panggilan balik yang Anda gunakan untuk menangani transisi antar negara.

2.2.5. Siklus Hidup Callback

Bagian ini memberikan informasi konseptual dan implementasi tentang metode panggilan balik yang digunakan selama siklus hidup aktivitas. Beberapa tindakan, seperti memanggil setContentView(), termasuk dalam metode siklus hidup aktivitas itu sendiri. Namun, kode yang menerapkan tindakan komponen dependen harus ditempatkan dalam komponen itu sendiri. Untuk mencapai ini, Anda harus membuat komponen yang sadar siklus hidup. Lihat Menangani Siklus Hidup dengan Komponen Siklus-Sadar untuk mempelajari cara membuat komponen dependen Anda sadar siklus.

onCreate()

Anda harus menerapkan panggilan balik ini, yang menyala saat sistem pertama kali membuat aktivitas. Pada pembuatan aktivitas, aktivitas memasuki kondisi Dibuat. Dalam metode onCreate(), Anda melakukan logika *startup* aplikasi dasar yang harus terjadi hanya sekali selama seumur hidup aktivitas. Misalnya, implementasi onCreate() mungkin mengikat data ke daftar, mengaitkan aktivitas dengan *ViewModel*, dan membuat *instance* beberapa variabel lingkup kelas. Metode ini menerima parameter *SavedInstanceState*, yang merupakan objek bundel yang berisi keadaan aktivitas yang disimpan sebelumnya. Jika aktivitas belum pernah ada sebelumnya, nilai objek Bundle adalah nol.

Contoh berikut dari metode onCreate() menunjukkan pengaturan mendasar untuk aktivitas, seperti mendeklarasikan antarmuka pengguna (didefinisikan dalam file tata letak XML), mendefinisikan variabel anggota, dan mengkonfigurasi beberapa UI. Dalam contoh ini, file tata letak XML ditentukan dengan meneruskan ID sumber daya file R.layout.main_activity ke setContentView().

```
TextView textView;

// some transient state for the activity instance
String gameState;

@Override
public void onCreate(Bundle savedInstanceState) {
    // call the super class onCreate to complete the creation of activity like
    // the view hierarchy
    super.onCreate(savedInstanceState);

    // recovering the instance state
    if (savedInstanceState != null) {
        gameState = savedInstanceState.getString(GAME_STATE_KEY);
    }

    // set the user interface layout for this activity
    // the layout file is defined in the project res/layout/main_activity.xml file
    setContentView(R.layout.main_activity);

    // initialize member TextView so we can manipulate it later
    textView = (TextView) findViewById(R.id.text_view);
}

// This callback is called only when there is a saved instance that is previously saved by using
// onSaveInstanceState(). We restore some state in onCreate(), while we can optionally restore
// other state here, possibly usable after onStart() has completed.
// The savedInstanceState Bundle is same as the one used in onCreate().
@Override
public void onRestoreInstanceState(Bundle savedInstanceState) {
    textView.setText(savedInstanceState.getString(TEXT_VIEW_KEY));
}

// invoked when the activity may be temporarily destroyed, save the instance state here
@Override
public void onSaveInstanceState(Bundle outState) {
    outState.putString(GAME_STATE_KEY, gameState);
    outState.putString(TEXT_VIEW_KEY, textView.getText());

    // call superclass to save any view hierarchy
    super.onSaveInstanceState(outState);
}
```

Gambar 2.3 Metode onCreate()

Sebagai alternatif untuk mendefinisikan file XML dan meneruskannya ke setContentView (), Anda bisa membuat objek View baru dalam kode aktivitas Anda dan membangun hierarki tampilan dengan memasukkan Tampilan baru ke dalam ViewGroup. Anda kemudian menggunakan tata letak itu dengan melewatkan root ViewGroup ke setContentView(). Untuk informasi lebih lanjut tentang membuat antarmuka pengguna, lihat dokumentasi Antarmuka Pengguna. Aktivitas Anda tidak berada di negara Dibuat. Setelah metode onCreate() menyelesaikan eksekusi, aktivitas memasuki keadaan awal, dan sistem memanggil metode onStart() dan onResume() secara berurutan. Bagian selanjutnya menjelaskan panggilan balik onStart().

onStart()

Ketika aktivitas memasuki kondisi Mulai, sistem memanggil panggilan balik ini. Panggilan onStart() membuat aktivitas terlihat oleh pengguna, saat aplikasi

mempersiapkan aktivitas untuk memasuki latar depan dan menjadi interaktif. Misalnya, metode ini adalah tempat aplikasi menginisialisasi kode yang mengelola UI. Metode onStart() selesai dengan sangat cepat dan, seperti pada kondisi Dibuat, aktivitas tidak tetap berada dalam status Mulai. Setelah panggilan balik ini selesai, aktivitas memasuki status Dilanjutkan, dan sistem memanggil metode onResume().

onResume()

Ketika aktivitas memasuki keadaan dilanjutkan, ia datang ke latar depan, dan kemudian sistem memanggil panggilan balik onResume(). Ini adalah keadaan di mana aplikasi berinteraksi dengan pengguna. Aplikasi tetap dalam kondisi ini sampai terjadi sesuatu untuk mengambil fokus dari aplikasi. Peristiwa semacam itu mungkin, misalnya, menerima panggilan telepon, pengguna bernavigasi ke aktivitas lain, atau layar perangkat mati. Saat aktivitas beralih ke status yang dilanjutkan, komponen sadar siklus apa pun yang terkait dengan siklus hidup aktivitas akan menerima acara ON_RESUME. Di sinilah komponen siklus hidup dapat mengaktifkan fungsionalitas apa pun yang perlu dijalankan saat komponen terlihat dan di latar depan, seperti memulai pratinjau kamera. Ketika suatu peristiwa interupsi terjadi, aktivitas memasuki keadaan dijeda, dan sistem memanggil panggilan balik onPause(). Jika aktivitas kembali ke status Lanjutkan dari kondisi Jeda, sistem sekali lagi memanggil metode onResume(). Karena alasan ini, Anda harus menerapkan onResume() untuk menginisialisasi komponen yang Anda lepaskan selama onPause(), dan melakukan inisialisasi lainnya yang harus terjadi setiap kali aktivitas memasuki keadaan dilanjutkan.:

```
public class CameraComponent implements LifecycleObserver {

    ...

    @OnLifecycleEvent(Lifecycle.Event.ON_RESUME)
    public void initializeCamera() {
        if (camera == null) {
            getCamera();
        }
    }

    ...
}
```

Gambar 2.4 Metode onResume()

Kode di atas menempatkan kode inisialisasi kamera dalam komponen yang menyadari siklus hidup. Anda bisa menempatkan kode ini langsung ke *callback* daur hidup aktivitas seperti onStart() dan onStop() tetapi ini tidak dianjurkan. Menambahkan logika ini ke dalam komponen yang mandiri dan sadar siklus memungkinkan Anda untuk menggunakan kembali komponen tersebut di berbagai aktivitas tanpa harus menduplikasi kode. Lihat Menangani Siklus Hidup dengan Komponen Siklus-Sadar untuk mempelajari cara membuat komponen yang sadar siklus.

onPause()

Sistem memanggil metode ini sebagai indikasi pertama bahwa pengguna meninggalkan aktivitas Anda (meskipun itu tidak selalu berarti aktivitas sedang dihancurkan); itu menunjukkan bahwa aktivitas tidak lagi di latar depan (meskipun mungkin masih terlihat jika pengguna berada dalam mode multi-jendela). Gunakan metode onPause() untuk menjeda atau menyesuaikan operasi yang tidak boleh dilanjutkan (atau harus dilanjutkan dalam jumlah sedang saat Aktivitas dalam keadaan Jeda, dan Anda berharap untuk melanjutkan segera. Ada beberapa alasan mengapa suatu kegiatan dapat memasuki kondisi ini. Sebagai contoh:

- Beberapa acara mengganggu eksekusi aplikasi, seperti yang dijelaskan di bagian onResume (). Ini adalah kasus yang paling umum.
- Di Android 7.0 (API level 24) atau lebih tinggi, beberapa aplikasi berjalan dalam mode multi-jendela. Karena hanya satu aplikasi (windows) yang memiliki fokus kapan saja, sistem menjeda semua aplikasi lain.
- Aktivitas semi-transparan baru (seperti dialog) terbuka. Selama aktivitas masih terlihat sebagian tetapi tidak dalam fokus, itu tetap dijeda.

Anda juga dapat menggunakan metode onPause() untuk melepaskan sumber daya sistem, menangani sensor (seperti GPS), atau sumber daya apa pun yang dapat memengaruhi masa pakai baterai saat aktivitas Anda dijeda dan pengguna tidak membutuhkannya. Namun, seperti yang disebutkan di atas di bagian onResume(), aktivitas yang Dijeda mungkin masih sepenuhnya terlihat jika dalam mode multi-jendela. Karena itu, Anda harus mempertimbangkan menggunakan onStop()

daripada onPause() untuk sepenuhnya melepaskan atau menyesuaikan sumber daya dan operasi terkait UI untuk lebih mendukung mode multi-jendela.

```
public class JavaCameraComponent implements LifecycleObserver {

    ...

    @OnLifecycleEvent(Lifecycle.Event.ON_PAUSE)
    public void releaseCamera() {
        if (camera != null) {
            camera.release();
            camera = null;
        }
    }

    ...
}
```

Gambar 2.5 Metode onPause()

Penyelesaian metode onPause() tidak berarti bahwa aktivitas meninggalkan status Jeda. Sebaliknya, aktivitas tetap dalam kondisi ini sampai aktivitas dilanjutkan atau menjadi sama sekali tidak terlihat oleh pengguna. Jika aktivitas dilanjutkan, sistem sekali lagi memanggil panggilan balik onResume(). Jika aktivitas kembali dari status Jeda ke status Dilanjutkan, sistem menyimpan instance instance tetap dalam memori, mengingat instance tersebut ketika sistem memanggil onResume(). Dalam skenario ini, Anda tidak perlu menginisialisasi ulang komponen yang dibuat selama salah satu metode panggilan balik yang mengarah ke status Lanjutkan. Jika aktivitas menjadi benar-benar tidak terlihat, sistem memanggil onStop (). Bagian selanjutnya membahas panggilan balik onStop().

onStop()

Ketika aktivitas Anda tidak lagi terlihat oleh pengguna, itu telah memasuki status Berhenti, dan sistem memanggil panggilan balik onStop(). Ini dapat terjadi, misalnya, ketika aktivitas yang baru diluncurkan mencakup seluruh layar. Sistem juga dapat memanggil onStop() ketika aktivitas telah selesai berjalan, dan akan dihentikan.

Dalam metode onStop(), aplikasi harus melepaskan atau menyesuaikan sumber daya yang tidak diperlukan saat aplikasi tidak terlihat oleh pengguna. Misalnya, aplikasi Anda dapat menjeda animasi atau beralih dari pembaruan lokasi berbutir halus ke berbutir kasar. Menggunakan onStop() dan bukan onPause()

memastikan bahwa pekerjaan terkait UI berlanjut, bahkan ketika pengguna melihat aktivitas Anda dalam mode multi-jendela.

Anda juga harus menggunakan onStop() untuk melakukan operasi shutdown yang relatif intensif CPU. Misalnya, jika Anda tidak dapat menemukan waktu yang lebih tepat untuk menyimpan informasi ke database, Anda mungkin melakukannya selama onStop(). Contoh berikut menunjukkan implementasi onStop() yang menyimpan konten catatan konsep ke penyimpanan persisten:

```
@Override
protected void onStop() {
    // call the superclass method first
    super.onStop();

    // save the note's current draft, because the activity is stopping
    // and we want to be sure the current note progress isn't lost.
    ContentValues values = new ContentValues();
    values.put(NotePad.Notes.COLUMN_NAME_NOTE, getCurrentNoteText());
    values.put(NotePad.Notes.COLUMN_NAME_TITLE, getCurrentNoteTitle());

    // do this update in background on an AsyncQueryHandler or equivalent
    asyncQueryHandler.startUpdate (
            mToken,  // int token to correlate calls
            null,    // cookie, not used here
            uri,    // The URI for the note to update.
            values,  // The map of column names and new values to apply to them.
            null,    // No SELECT criteria are used.
            null     // No WHERE columns are used.
    );
}
```

Gambar 2.6 Metode onStop()

Ketika aktivitas Anda memasuki status Berhenti, objek Aktivitas disimpan di dalam memori: Objek menyimpan semua informasi status dan anggota, tetapi tidak dilampirkan ke manajer jendela. Ketika aktivitas dilanjutkan, aktivitas mengingat informasi ini. Anda tidak perlu menginisialisasi ulang komponen yang dibuat selama salah satu metode panggilan balik yang mengarah ke status Lanjutkan. Sistem juga melacak keadaan saat ini untuk setiap objek Lihat dalam tata letak, jadi jika pengguna memasukkan teks ke dalam widget EditText, konten tersebut dipertahankan sehingga Anda tidak perlu menyimpan dan mengembalikannya.

onDestroy()

onDestroy() dipanggil sebelum aktivitas dihancurkan. Sistem memanggil panggilan balik ini karena:

- aktivitas sedang selesai (karena pengguna benar-benar mengabaikan aktivitas atau karena selesai () dipanggil pada aktivitas).

- sistem sementara menghancurkan aktivitas karena perubahan konfigurasi (seperti rotasi perangkat atau mode multi-jendela).

Jika aktivitas selesai, onDestroy() adalah panggilan balik siklus hidup terakhir yang diterima aktivitas. Jika onDestroy() dipanggil sebagai hasil dari perubahan konfigurasi, sistem segera membuat instance aktivitas baru dan kemudian memanggil onCreate() pada instance baru dalam konfigurasi baru. Callback onDestroy() harus melepaskan semua sumber daya yang belum dirilis oleh callback sebelumnya seperti onStop().

2.2.6. Android Studio

Android Studio adalah IDE resmi untuk pengembangan aplikasi Android, berdasarkan IntelliJ IDEA. Selain kemampuan yang Anda harapkan dari IntelliJ, Android Studio menawarkan:

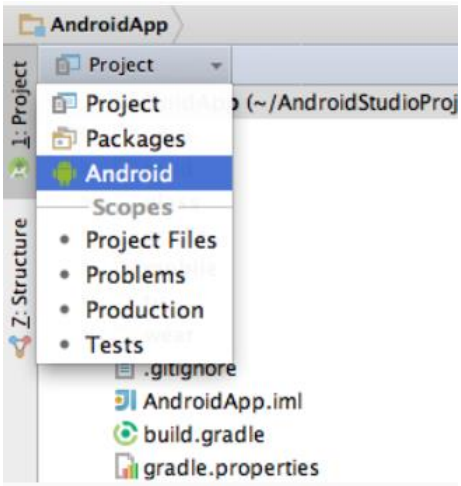
- Sistem pembangunan berbasis Gradle fleksibel.
- Buat varian dan beberapa pembuatan *file* apk.
- *Template* kode untuk membantu Anda membangun fitur aplikasi umum.
- Editor tata letak kaya dengan dukungan untuk mengedit tema *drag and drop*.
- Alat serat untuk menangkap kinerja, kegunaan, kompatibilitas versi, dan masalah lainnya.
- Kemampuan ProGuard dan penandatanganan aplikasi.
- Dukungan bawaan untuk Google Cloud Platform, membuatnya mudah untuk mengintegrasikan Google Cloud Messaging dan App Engine.

1. **Struktur Proyek**

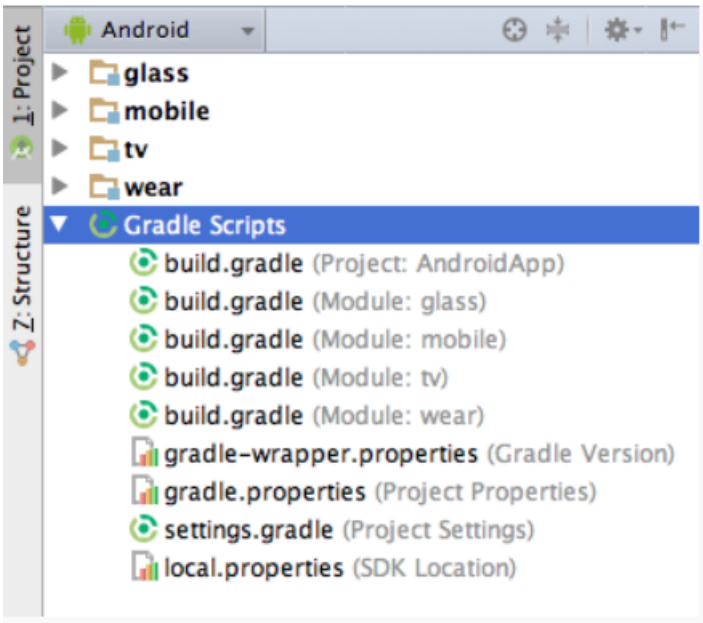
Secara default, Android Studio menampilkan file profil Anda dalam tampilan proyek Android. Tampilan ini menunjukkan versi rata dari struktur proyek Anda yang menyediakan akses cepat ke file sumber utama proyek Android dan membantu Anda bekerja dengan sistem build berbasis Gradle yang baru. Tampilan proyek Android:

- Grup file build untuk semua modul di tingkat atas hirarki proyek.
- Memperlihatkan direktori sumber paling penting di tingkat atas hirarki modul.
- Grup semua file manifes untuk setiap modul.
- Memperlihatkan file sumber daya dari semua set sumber Gradle.

- Grup file sumber daya untuk berbagai lokal, orientasi, dan jenis layar dalam satu grup per jenis sumber daya



Gambar 2.7 File proyek di tampilan Android



Gambar 2.8 File pembuatan proyek Android

Setiap proyek di Android Studio berisi satu atau beberapa modul dengan file kode sumber dan file sumber daya. Jenis-jenis modul mencakup:

- Modul aplikasi Android
- Modul Pustaka
- Modul Google *App Engine*

Secara default, Android Studio akan menampilkan file proyek Anda dalam tampilan proyek Android, seperti yang ditampilkan dalam gambar 1. Tampilan

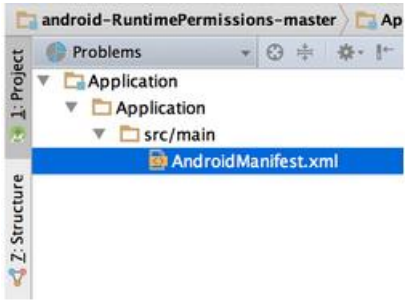
disusun berdasarkan modul untuk memberikan akses cepat ke file sumber utama proyek Anda.

Semua file versi terlihat di bagian atas di bawah **Gradle Scripts** dan masing-masing modul aplikasi berisi folder berikut:

- manifests: Berisi file *AndroidManifest.xml.*
- java: Berisi file kode sumber Java, termasuk kode pengujian JUnit.
- res: Berisi semua sumber daya bukan kode, seperti tata letak XML, string UI, dan gambar bitmap.

Sruktur proyek Android pada disk berbeda dari representasi rata ini. Untuk melihat struktur file sebenarnya dari proyek ini, pilih **Project** dari menu tarik turun **Project** (dalam gambar 1, struktur ditampilkan sebagai **Android**).

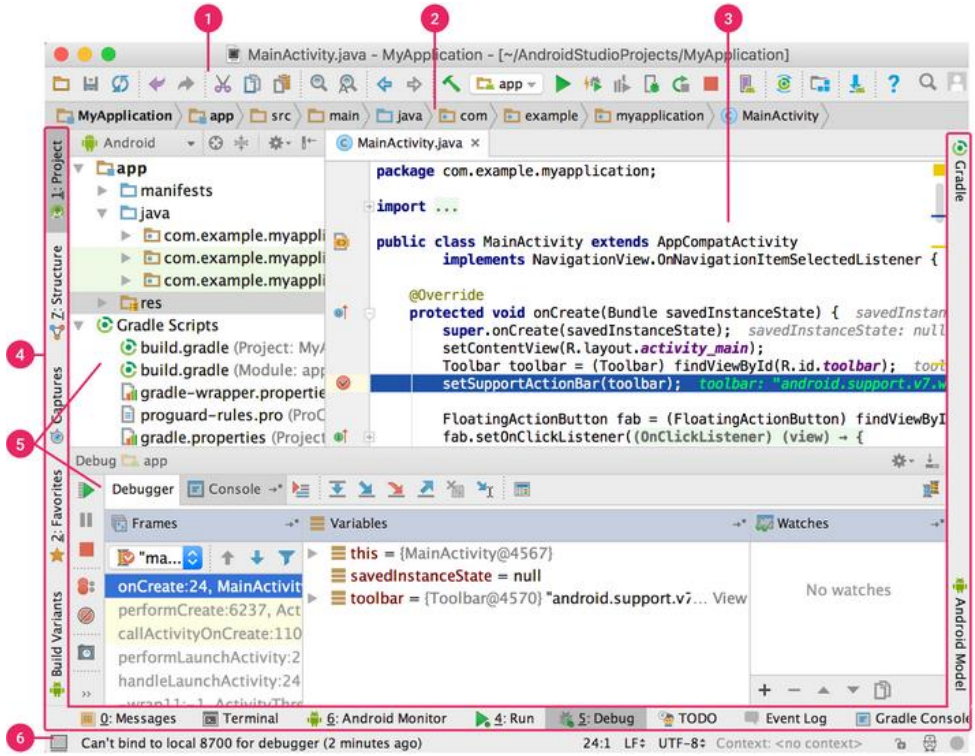
Anda juga bisa menyesuaikan tampilan file proyek untuk berfokus pada aspek tertentu dari pengembangan aplikasi Anda. Misalnya, memilih tampilan **Problems** dari tampilan proyek Anda akan menampilkan tautan ke file sumber yang berisi kesalahan pengkodean dan sintaks yang dikenal, misalnya tag penutup elemen XML tidak ada dalam file tata letak.



Gambar 2.9 File proyek di tampilan Problem.

2. Antarmuka Pengguna

Jendela utama Android Studio terdiri dari beberapa bidang logika yang diidentifikasi dalam gambar berikut.



Gambar 2.10 Jendela utama Android Studio

1) Bilah alat memungkinkan Anda untuk melakukan berbagai jenis tindakan, termasuk menjalankan aplikasi dan meluncurkan alat Android.
2) Bilah navigasi membantu Anda bernavigasi di antara proyek dan membuka file untuk diedit. Bilah ini memberikan tampilan struktur yang terlihat lebih ringkas dalam jendela Project.
3) Jendela editor adalah tempat Anda membuat dan memodifikasi kode. Bergantung pada jenis file saat ini, editor dapat berubah. Misalnya, ketika melihat file tata letak, editor menampilkan Layout Editor.
4) Bilah jendela alat muncul di luar jendela IDE dan berisi tombol yang memungkinkan Anda meluaskan atau menciutkan jendela alat individual.
5) Jendela alat memberi Anda akses ke tugas tertentu seperti pengelolaan proyek, penelusuran, kontrol versi, dan banyak lagi. Anda bisa meluaskan dan juga menciutkannya.
6) Bilah status menampilkan status proyek Anda dan IDE itu sendiri, serta setiap peringatan atau pesan.

Anda bisa menata jendela utama untuk memberi Anda ruang layar yang lebih luas dengan menyembunyikan atau memindahkan bilah alat dan jendela alat. Anda

juga bisa menggunakan pintasan keyboard untuk mengakses sebagian besar fitur IDE.

Anda dapat menelusuri seluruh kode sumber, basis data, tindakan, elemen antarmuka pengguna, dan seterusnya setiap saat dengan menekan tombol Shift dua kali, atau mengeklik kaca pembesar di sudut kanan atas dari jendela Android Studio. Ini akan sangat berguna misalnya saat Anda mencoba menemukan tindakan IDE tertentu yang Anda lupakan cara memicunya.

Jendela Alat

Dari pada menggunakan perspektif yang sudah diatur sebelumnya, Android Studio mengikuti konteks Anda dan secara otomatis memunculkan jendela alat yang relevan saat Anda bekerja. Secara default, alat yang tersering dipakai akan disematkan ke bilah jendela alat di tepi jendela aplikasi.

- Untuk meluaskan atau menciutkan jendela alat, klik nama alat di bilah jendela alat. Anda juga bisa menyeret, menyematkan, melampirkan, dan melepaskan jendela alat.
- Untuk kembali ke tata letak jendela alat default saat ini, klik *Window* > *Restore Default Layout* atau sesuaikan tata letak default Anda dengan mengeklik *Window* > *Store* Current Layout as Default.
- Untuk kembali ke tata letak jendela alat default saat ini, klik **Window > Restore Default Layout** atau sesuaikan tata letak default Anda dengan mengeklik **Window > Store Current Layout as Default**.
- Untuk menampilkan atau menyembunyikan bilah jendela alat, klik ikon jendela di sudut kiri bawah jendela Android Studio
- Untuk menemukan jendela alat tertentu, arahkan ke atas ikon jendela dan pilih jendela alat tersebut dari menu.

Anda juga bisa menggunakan pintasan keyboard untuk membuka jendela alat. Tabel 2.1 mencantumkan pintasan jendela paling umum.

Tabel 2. 1 Keyboard shortcut ke beberapa jendela alat yang penting

Jendela Alat	Windows dan Linux	Mac
Proyek	Alt+1	Command+1
Kontrol Versi	Alt+9	Command+9
Run	Shift+F10	Control+R
Debug	Shift+F9	Control+D
Android Monitor	Alt+6	Command+6
Kembali ke Editor	Esc	Esc
Menyembunyikan Semua Jendela Alat	Control+Shift+F12	Esc

Jika Anda ingin menyembunyikan semua bilah alat, jendela alat, dan tab editor, klik *View > Enter Distraction Free Mode*. Ini akan mengaktifkan *Distraction Free Mode*. Untuk keluar dari *Distraction Free Mode*, klik *View > Exit Distraction Free Mode*.

Anda bisa menggunakan *Speed Search* untuk menelusuri dan memfilter di dalam sebagian besar jendela alat dalam Android Studio. Untuk menggunakan *Speed Search*, pilih jendela alat lalu ketik kueri penelusuran Anda.

Pelengkapan Kode

Android Studio memiliki tiga jenis pelengkapan kode, yang bisa anda akses memakai pintasan keyboard.

Tabel 2. 2 Keyboard shortcut untuk pelengkapan code

Tipe	Keterangan	Windows dan Linux	Mac
Pelengkapan Dasar	Menampilkan saran dasar untuk variabel, tipe, metode, ekspresi, dan seterusnya. Jika Anda memanggil pelengkapan dasar dua kali secara berturutturut, Anda melihat lebih banyak hasil, termasuk anggota pribadi dan anggota statis yang tidak diimpor.	Control+Space	Control+Space
Pelengkapan Cerdas	Menampilkan opsi relevan berdasarkan konteks. Pelengkapan cerdas	Control+Shift+Space	Control+Shift+Space

	mengetahui tipe yang diharapkan dan alur data. Jika Anda memanggil Pelengkapan Cerdas dua kali berturut-turut, Anda akan melihat lebih banyak hasil, termasuk rantai.		
Pelengkapan Pernyataan	Membantu Anda melengkapi pernyataan saat ini, menambahkan tanda kurung, tanda kurung siku, tanda kurung kurawal, pemformatan, dsb.	Control+Shift+Enter	Shift+Command+Enter

Anda juga bisa melakukan perbaikan cepat dan menunjukkan tindakan maksud Anda dengan menekan Alt+Enter.

Navigasi

Berikut beberapa tip untuk membantu Anda menjelajahi di dalam Android Studio.

- Beralih antar file yang baru saja diakses menggunakan tindakan *Recent Files*. Tekan *Control+E* (*Command+E* pada Mac) untuk memunculkan tindakan *Recent Files*. Secara default, akses yang terakhir dipilih. Anda juga bisa mengakses jendela alat mana saja melalui kolom kiri dalam tindakan ini.
- Tampilkan struktur file saat ini menggunakan tindakan *File Structure*. Munculkan tindakan *File Structure* dengan menekan *Control+F12* (*Command+F12* pada Mac). Menggunakan tindakan ini, Anda bisa menavigasi dengan cepat ke bagian mana pun dari file Anda saat ini.
- Telusuri dan masuk ke kelas tertentu di proyek menggunakan *tindakan Navigate to Class*. Munculkan tindakan dengan menekan *Control+N* (*Command+O* pada Mac). Navigasikan ke Kelas yang mendukung ekspresi canggih, termasuk *CamelHumps*, jalur, baris menavigasi ke, nama tengah pencocokan, dan banyak lagi. Jika Anda memanggilnya dua kali berturut-turut, hasil dari kelas proyek akan ditampilkan.

- Masuk ke file atau folder menggunakan tindakan "*Navigate to File**. Munculkan tindakan *Navigate to File* dengan menekan *Control+Shift+N* (*Command+Shift+O* pada Mac). Untuk menelusuri folder dan bukan file, tambahkan / di akhir ekspresi Anda.
- Masuk ke metode atau bidang menurut nama menggunakan tindakan *Navigate to Symbol*. Munculkan tindakan *Navigate to Symbol* dengan menekan *Control+Shift+Alt+N* (*Command+Shift+Alt+O* pada Mac).
- Temukan semua bagian kode yang merujuk kelas, metode, bidang, parameter, atau pernyataan di posisi kursor saat ini dengan menekan Alt+F7

Gaya dan Pemformatan

Saat Anda mengedit, Android Studio otomatis menerapkan pemformatan dan gaya seperti yang ditetapkan dalam setelan gaya kode. Anda dapat menyesuaikan setelan gaya kode dengan bahasa pemrograman, termasuk menetapkan konvensi untuk tab dan inden, spasi, pembungkusan dan tanda kurung kurawal, dan baris kosong. Untuk menyesuaikan setelan gaya kode, *klik File > Settings > Editor > Code Style (Android Studio > Preferences > Editor > Code Style* pada Mac.) Meski IDE otomatis menerapkan pemformatan saat Anda bekerja, Anda juga dapat secara eksplisit memanggil tindakan Reformat Code dengan menekan *Control+Alt+L* (*Opt+Command+L* pada Mac), atau inden otomatis semua baris dengan menekan *Control+Alt+I* (*Alt+Option+I* pada Mac).

Dasar-dasar Kontrol Versi

Android Studio mendukung berbagai versi sistem kontrol, termasuk *Git, GitHub, CVS, Mercurial, Subversion*, dan Penyimpanan *Google Cloud Source*.

Setelah mengimpor aplikasi Anda ke dalam Android Studio, gunakan opsi menu Android Studio VCS untuk mengaktifkan dukungan VCS bagi sistem kontrol versi yang diinginkan, membuat penyimpanan, mengimpor file baru ke dalam kontrol versi, dan melakukan pengoperasian kontrol versi lainnya:

- Dari menu Android Studio VCS, klik *Enable Version Control Integration*.
- Dari menu tarik-turun, pilih sistem kontrol versi yang terkait dengan akar proyek, lalu klik OK.

- Menu VCS sekarang menunjukkan sejumlah opsi kontrol versi berdasarkan sistem yang Anda pilih.

2.2.7. Sistem Versi Gradel

Android Studio menggunakan Gradle sebagai dasar sistem versi, dengan kemampuan khusus Android yang disediakan oleh Plugin Android untuk Gradle. Sistem ini bisa dijalankan sebagai alat terpadu dari menu Android Studio dan secara independen dari baris perintah. Anda bisa menggunakan fitur-fitur sistem versi untuk melakukan yang berikut:

- Menyesuaikan, mengonfigurasi, dan memperluas proses pembangunan.
- Membuat beberapa APK untuk aplikasi Android Anda, dengan aneka fitur menggunakan proyek dan modul yang sama. Menggunakan kembali kode dan sumber daya pada seluruh set sumber

Dengan menerapkan fleksibilitas Gradle, Anda dapat mencapai semua ini tanpa mengubah file sumber inti aplikasi. File versi Android Studio diberi nama build.gradle. File ini adalah teks biasa yang menggunakan Groovy mengonfigurasi versi dengan elemen yang disediakan oleh plugin Android untuk Gradle. Masing-masing proyek memiliki file versi level atas untuk seluruh proyek dan file versi level modul terpisah untuk setiap modul. Saat Anda mengimpor proyek saat ini, Android Studio otomatis menghasilkan file versi yang diperlukan.

Varian Versi

Sistem versi dapat membantu Anda membuat versi berbeda dari aplikasi yang sama dari satu proyek. Ini berguna ketika Anda sama-sama memiliki versi gratis dan versi berbayar dari aplikasi, atau jika Anda ingin mendistribusikan beberapa APK untuk perangkat berbeda di Google Play.

Pemisahan APK

Pemisahan APK memungkinkan Anda untuk membuat beberapa APK berdasarkan kepadatan layar atau ABI. Misalnya, pemisahan APK memungkinkan Anda membuat versi hdpi dan mdpi terpisah dari aplikasi sembari masih mempertimbangkannya sebagai satu varian dan memungkinkannya untuk berbagi setelan aplikasi pengujian, javac, dx, dan ProGuard.

Penyusutan Sumber Daya

Penyusutan sumber daya di Android Studio secara otomatis membuang sumber daya yang tidak terpakai dari aplikasi terkemas dan dependensi perpustakaan. Misalnya, jika aplikasi Anda menggunakan layanan Google Play untuk mengakses fungsi Google Drive, dan saat ini Anda tidak memakai Google Sign-In, maka penyusutan sumber daya dapat membuang berbagai aset yang dapat digambar untuk tombol SignIn Button.

Mengelola Dependensi

Dependensi untuk proyek Anda ditetapkan oleh nama dalam file build.gradle. Gradle menangani penemuan dependensi Anda dan menyediakannya di versi Anda. Anda bisa mendeklarasikan dependensi modul, dependensi biner jarak jauh, dan dependensi biner setempat dalam file build.gradle Anda. Android Studio mengonfigurasi proyek untuk menggunakan Penyimpanan Pusat Maven secara default. (Konfigurasi ini disertakan dalam file versi tingkat atas untuk proyek tersebut).

2.3. Java

Java merupakan Bahasa pemrograman tingkat tinggi yang dipelopori oleh James Gosling yang merupaka enginer di Sun Microsystem. Java mulai dibangun pada tahun 1991. Versi alpha dan beta dari java dirilis pada tahun 1995, 4 tahun setelah projek Java diinisiasi. Pada tahun 2010, Sun Microsystem diakusisi oleg Oracle dan menjadikan Java dikembangkan di bawah kuasa Oracle.

Per Januari 2018, versi stabil terkahir dari Java yaitu versi Java SE 9. Sebelumnya Java SE 9, terdapat beberapa versi dari Java yang telah dirilis. Versi – versi dari java yang pernah dirilis yaitu :

- JDK Aplha dan Beta (1995)
- JDK 1.0 (23 Januari 1996)
- JDK 1.1 (19 Februari 1997)
- J2SE 1.2 (8 Desember 1998)
- J2SE 1.3 (8 Mei 200)
- J2SE 1.4 (6 Februari 2002)
- J2SE 5.0 (30 September 2004

- Java SE 6 (11 Desember 2006)
- Java SE 7 (18 Juli 2011)
- Java SE 8 (18 Maret 2014)
- Java SE 9 (21 September 2017)

Java merupakan salah satu Bahasa yang popular saat ini. Saking populernya, beberapa edisi untuk Java pu diciptakan, contohnya yaitu J2EE untuk Aplikasi Enterprise dan J2ME untuk Aplikasi Mobile. Salah satu alasan lain mengapa Java merupakan bahasa yang popular yaitu dikarenakan Java dapat berjalan di bebagai platfrom system operasi. Java pun dikenal dengan *Write Once, Run Anywhere* karena kompatibilitasnya tersebut. Menuru Sun yang merupakan pelopor dari Java, Java itu:

Simpel

Java merupakan Bahasa yang simple sehingga mudah dipahami. Bermodalkan pengetahuan programming dasar, fundamental dari pemrograman java akan cepat dipahamai

Berorientasi Objek

Java menggunakan konsep pemrograman berorientasi objek sehingga pembuat aplikasi bisa dilakukan lebih modular.

Kuat

Java didesain untuk membuat aplikasi yang memiliki reliabilitas tinggi. Salah satu cara Java untuk membuat program yang memiliki reliabilitas tinggi yaitu dengan mengeliminasi situasi error dengan memeriksanya pada compile time dan runtime.

Aman

Java didesain untuk digunakan pada lingkungan yang terdistribusi. Keamanan merupakan hal yang penting dalam lingkungan terdistribusi. Fitur – fitur keamanan yang dimiliki Java membuat perangkat lunak yang dibuat tidak bisa diserang dari luar atau disisipi virus.

Arsitektur Netral

Aplikasi yang dibuat menggunakan Java merupakan aplikasi yang platfrom independent. Aplikasi hanya perlu satu buah versi yang bisa dijalankan pada platfrom sistem operasi yang berbeda.

Portable

Java dengan karakteristik arsitektur netralnya membuat Java tidak bergantung pada mesin tertentu atau dengan kata lain Java sangatlah portable.

Perfoma Tinggi

Java sangat memperhatikan perfoma. Dengan pengenalan *Just in Time compilation*, proses kompilasi Java menjadi lebih cepat.

Multithreaded

Java memungkinkan pembuatan aplikasi yang bisa melakukan beberapa pekerjaan secara bersamaan.

Dinamis

Java lebih dinamis dibandingan C atau C++ karena didesain untuk dijalankan pada langkungan yang dinamis.

2.3.1. Sejarah Java

Sejarah Java dimulai pada tahun 1991. Sebuah tim kecil yang berisikan engineer dari Sun Microsystem mempelopori proyek Java ini. Tim ini disebut Green Team dan dipimpin oleh James Gosling. Saat awal pengembangag,, bahasa yang dikembangkan oleh Green Team disebut nama “Greentalk”. Setelah itu, mereka mengganti Namanya menjadi “Oak” ketika pohon oak yang tiba-tiba muncul di luar kantornya. Pada tahun 1995, Green Team menggnati Namanya menjadi “Java” karena Oak sudah menjadi trademark dari Oak Technologies. Pada tahun itu, Sun Microsystem lalu merilis Java untuk pertama kali.Pada 13 November 2006, Sun Microsystem merilis Java dengan gratis dan open source dengan lisensi GNU General Public License (GPL). Tetapi pada tahun 2010, Sun Microsystem dibeli oleh Oracle dan menjadikan Java dikembangkan dan dipelihara di bawah kendali Oracle.

2.3.2. Spesifikasi Java

Spesifikasi minimum untuk komputer menjalankan Java yaitu Pentium 2 266 MHz dengan RAM 128 MB. Selain itu, perlu juga menyiapkan software berikut: [8]

- Sistem operasi Windows XP/7/8/10 atau Ubuntu Linux minimal versi 12.04 atau Mac OS X minimal versi 10.8.3.
- Java SE 8
- IDE atau text editor seperti Intellij IDEA, Netbeans, Eclipse, Visual Studio Code, Sublime, Atom, dan sebagainya.

2.4. API

Application Programming Interface (API) adalah seperangkat prosedur, fungsi, protokol dan aturan untuk membangun sebuah peangkat lunak. Secara umum, API menentukan bagaimana komponen perangkat lunak atau paket yang berbeda harus berinteraksi satu sama lain meskipun dengan bahasa yang sangat berbeda. API yang bagus memudahkan pengembangan program komputer dengan menyediakan semua blok bangunan, yang kemudian disatukan oleh pemrogram. API digunakan untuk sistem berbasis web, sistem operasi, sistem basis data, perangkat keras komputer atau perpustakaan perangkat lunak. Spesifikasi API dapat mengambil banyak bentuk, namun sering kali mencakup spesifikasi untuk rutinitas, struktur data, kelas objek, variabel atau panggilan jarak jauh. POSIX, Microsoft Windows API, C ++ Standard Template Library dan Java API adalah contoh dari berbagai bentuk API. Dokumentasi untuk API biasanya disediakan untuk memudahkan penggunaan.

2.5. *Speech Recogniton (Speech-To-Text)*

Speech to text adalah bidang teknologi yang berfokus pada identifikasi ucapan manusia dalam bentuk teks transkripsi. *Speech to text* dikembangkan dengan tujuan untuk memperoleh informasi dari audio, dan pengembangan sistem komputer pintar (kecedasan buatan) yang mampu memahami bahasa manusia. Menurut Waibel, sistem *speech to text* dapat dibagi menjadi beberapa jenis berdasarkan tingkat kesulitan intrinsik dan dimensinya (Waibel, 1990): [9]

A. *Word Recognition-Isolated (WR)*

Sistem WR adalah sistem speech to text dengan tipe masukan berupa ucapan kata-kata terisolasi. Setiap kata diberi jeda saat diucapkan, sehingga penggunaannya terbatas. Ukuran kosakata WR berkisar antara 10 hingga 300 kata.

B. *Connected Speech Recognition-restricted (CSR)*

Sistem CSR adalah sistem speech to text dengan tipe masukan berupa ucapan kata bersambung tanpa jeda. Sistem jenis ini dapat menyimpan kosakata antara 30 hingga 500 kata namun hanya dapat menerima bahasa perintah terbatas, bukan semua bahasa. Saat perekaman audio masukan, lingkungan perekaman harus hening dengan tingkat kebisingan rendah serta ucapan pembicara harus jelas. Tingkat akurasi sistem CSR dipengaruhi oleh kejelasan audio masukan.

C. *Speech Understanding-restricted (SU)*

Sistem SU merupakan jenis sistem *speech to text* yang bertujuan untuk memahami ucapan masukan. Oleh karena itu sistem SU dapat menerima ucapan kata bersambung dan dapat digunakan secara lebih bebas dibandingkan sistem WR dan CSR. Bahasa yang dapat dipahami oleh sistem SU adalah bahasa sesuai data pengetahuan linguistik (misal: bahasa Inggris) namun dengan tata bahasa terbatas. Keterbatasan tata bahasa SU dipengaruhi oleh kapasitas penyimpanan kosa kata sistem SU yaitu antara 100 hingga 2000 kata. Namun pada SU, sistem tidak lagi bergantung sepenuhnya pada siapa pembicaranya namun sistem sudah lebih pintar untuk dapat memahami ucapan berbagai pembicara sebagai satu kata yang sama. Tingkat pemahaman sistem *speech to text* bergantung pada sumber pengetahuan linguistik, semakin lengkap pengetahuan linguistik sistem maka tingkat pemahaman SU akan semakin kompleks.

D. *Dictation Machine-restricted (DM*)

Dictation machine adalah sistem *speech to text* yang merekam ucapan dan menyimpan hasil transkripsi teksnya untuk penggunaan selanjutnya. DM telah banyak digunakan untuk keperluan personal maupun legal dan medis. DM

memerlukan kosakata yang lebih besar daripada sistem speech to text lainnya yaitu antara 1000 hingga 10.000 kata dan pengguna DM harus merekam suaranya dengan jelas dalam lingkungan hening. Tingkat pemahaman DM bergantung pada kompleksitas pengetahuan linguistiknya dan kejelasan audio masukan.

E. *Unrestricted Speech Understanding (USU)*

USU adalah sistem *speech to text* yang membutuhkan kapasitas kosakata tidak terbatas untuk dapat menerima masukan ucapan kata bersambung dengan menggunakan pengetahuan linguistik untuk pemahaman teks transkripsi.

F. *Unrestricted Connected Speech Recognition*

Unrestricted Connected SR adalah sistem *speech to text* yang memiliki kapasitas kosakata dan jenis masukan sama dengan USU namun sistem tidak mengasumsikan jenis informasi apa yang disampaikan oleh pembicara sehinggapemahaman sistem lebih kompleks dari segi pengetahuan linguistik. [9]

Dari seluruh jenis sistem *speech to text* tersebut, akurasi sistem bergantung dari jenis informasi yang ingin diperoleh. Jenis informasi yang ingin diperoleh menggunakan *speech to text* akan mempengaruhi waktu respon sistem dalam memproses audio masukan dan relasinya dengan pengetahuan linguistik. Akurasi dan waktu respon sistem dapat diatur, semakin tinggi akurasi sistem maka waktu respon semakin lama, hal ini juga berlaku sebaliknya.

Menurut Ethonologue, salah satu sumber tata bahasa dunia terbaik, pada tahun 2014 terdapat 7,106 bahasa dengan 6,2 milyar pembicara di seluruh dunia (Lewis, 2014). Jumlah ini menunjukkan betapa kompleks dan luasnya potensi bidang speech to text. Namun terdapat beberapa isu dalam pengembangan speech to text (Waibel, 1990), antara lain:

1) Ucapan bersambung, terisolasi

Ucapan manusia memiliki berbagai keragaman meliputi gaya bahasa dan dialek, sebagian berbicara secara berkesinambungan tanpa jeda, sebagian berbicara dengan jeda untuk setiap kata. Karena perbedaan eksistensi jeda ini, maka modul

speech to text harus mampu membedakan ucapan yang berbeda namun merupakan kata yang sama.

2) Ukuran kosakata

Pada Desember 2010, sebuah studi oleh Harvard dan Google menemukan bahwa bahasa Inggris memiliki 1.022.000 kata dan terus bertambah dengan rating 8.500 kata per tahun (Injeeli, 2013). Berdasarkan data ini, modul *speech to text* harus mampu menyimpan setidaknya satu juta kata untuk setiap bahasa agar dapat mengidentifikasi ucapan secara akurat.

3) Keterbatasan tata bahasa

Setiap bahasa memiliki tata bahasa yang berbeda, tata bahasa ini diterapkan sebagai pengetahuan linguistik.

4) Ketergantungan sistem pada karakter suara pengguna

Modul *speech to text* memiliki dua jenis sistem identifikasi, yaitu ketergantungan pada suara pengguna dan sistem independen. Sistem yang bergantung pada suara pengguna memiliki keterbatasan identifikasi karena bersifat statis. Akurasi terbaik dapat diperoleh jika karakter suara pengguna sama atau mendekati data pelatihan. Sistem independen bersifat dinamis, setiap sistem mengidentifikasi suara baru, suara dimasukan ke datapelatihan untuk meningkatkan akurasi pada identifikasi selanjutnya.

5) Ambiguitas akustik

Dalam beberapa kosakata bahasa, satu kata yang sama dapat memiliki arti yang berbeda tergantung pada konteksnya pada kalimat. Ambiguitas ini diselesaikan melalui aspek semantik pada pengetahuan linguistik.

6) Kebisingan lingkungan

Untuk dapat memperoleh akurasi tinggi, sinyal ucapan masukan harus memiliki level kebisingan yang rendah. Namun tidak setiap masukan memiliki level kebisingan yang rendah, oleh karena itu bidang teknologi noise filter dikembangkan untuk mengurangi level kebisingan audio [9]

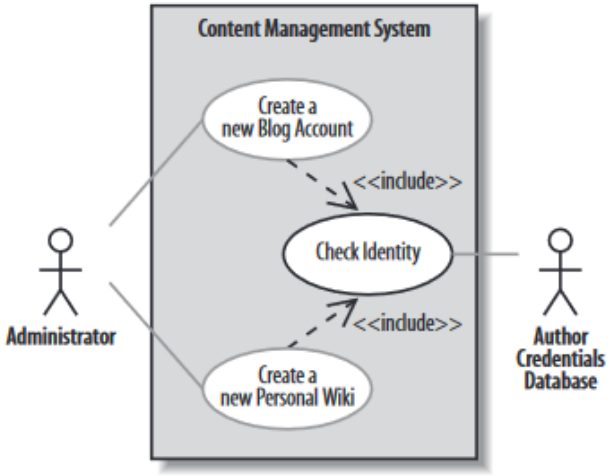
2.6. UML

UML merupakan singkatan dari Unified Modelling Language yang secara alih bahasa artinya pemodelan standar. Bahasa pemodelan bisa apa saja yang mengandung notasi dan deskripsi. Namun ada beberapa keuntungan dan kerugian dalam penggunaan UML sebagai pemodelan salah satunya yaitu bahasa yang formal dan jelas, dengan begitu kita tidak perlu khawatir gambaran akan model kita akan disalah artikan. [10]

2.6.1. Use Case

Use Case menjelaskan fungsionalitas dari sistem yang akan dimodelkan dari sudut pandang luar. Use Case nantinya akan dibutuhkan guna menjelaskan apa yang nanti akan dilakukan.

Use Case menjelaskan bagaimana nilai-nilai dari sistem akan disampaikan. Dalam pemodelan use case ada beberapa step yang harus dilakukan yaitu menemukan kebutuhan sistem, menentukan actor. Setiap actor nantinya akan mempunyai sebuah tugas dalam sistem yang akan dibuat dan mempunya deskripsi pekerjaan. [10]



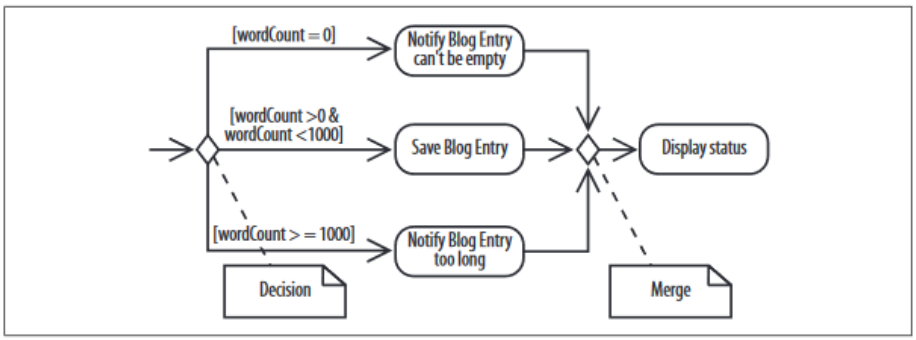
Gambar 2.11 Use Case Diagram

Use case name	Create a new Personal Wiki	
Related Requirements	Requirement A.2.	
Goal In Context	A new or existing author requests a new personal Wiki from the Administrator.	
Preconditions	The author has appropriate proof of identity.	
Successful End Condition	A new personal Wiki is created for the author.	
Failed End Condition	The application for a new personal Wiki is rejected.	
Primary Actors	Administrator.	
Secondary Actors	Author Credentials Database.	
Trigger	The Administrator asks the CMS to create a new personal Wiki.	
Main Flow	**Step**	**Action**
	1	The Administrator asks the system to create a new personal Wiki.
	2	The Administrator enters the author's details.
	3	The author's details are verified using the Author Credentials Database.
	4	The new personal Wiki is created.
	5	A summary of the new personal Wiki's details are emailed to the author.
Extensions	**Step**	**Branching Action**
	3.1	The Author Credentials Database does not verify the author's details.
	3.2	The author's new personal Wiki application is rejected.

Gambar 2.12 Use Case Scenario

2.6.2. Activity Diagram

Activity Diagram digunakan untuk menjelaskan lebih detil lagi bagaimana sistem akan mengerjakan tujuan-tujuannya. Activiy Diagram memperlihatkan aksi yang saling berhubungan satu sama lain untuk menunjukan proses yang ada dalam sebuah sistem. [10]

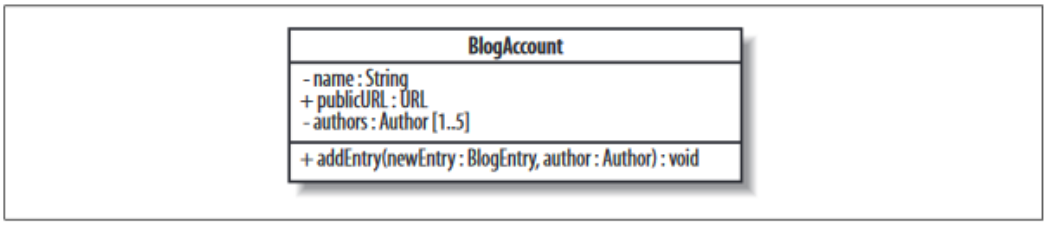


Gambar 2.13 Activity Diagram

2.6.3. Class Diagram

Class diagram merupakan blueprint dari object yang akan kita buat. Seperti contohnya mobil, mobil memiliki ribuan jenis namun class mobil hanyalah satu,

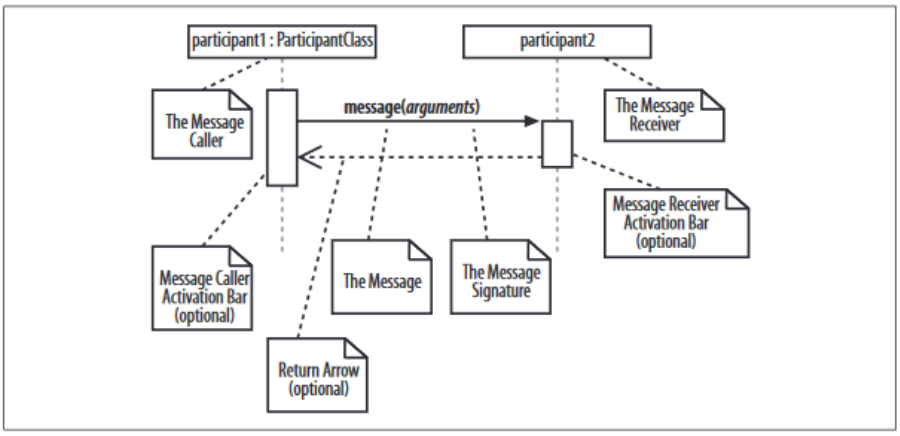
beroda empat dan mempunyai mesin. Sebuah class memiliki details tentang hal penting dalam model dan sistem. [10]



Gambar 2.14 Class Diagram

2.6.4. Sequence Diagram

Sequence diagram menjelaskan bagian-bagian sistem untuk memodelkan bagian mana yang saling terhubung dan berkomunikasi. Dalam sequence diagram perintah-perintah interaksi antara bagian satu dan bagian lainnya dalam sistem dinotasikan dengan event, signals dan messages. [10]



Gambar 2.15 Sequence Diagram

2.7. Database

Database merupakan kumpulan file-file yang saling berkaitan dan berinteraksi, relasi tersebut bila ditunjukan dengan kunci dari tiap-tiap file yang ada. Satu database menunjukkan suatu kumpulan data yang dipakai dalam suatu lingkup perusahaan, instansi. Pengolahan database merupakan suatu cara yang dilakukan terhadap file-file yang berada di suatu instansi yang mana file tersebut dapat disusun, diurut, diambil sewaktu-waktu serta dapat ditampilkan dalam bentuk suatu

laporan sehingga dapat mengolah file-file yang berisikan informasi tersebut secara rapi.

Database secara sederhana, dapat kita sebut sebagai gudang data. secara teori, database adalah kumpulan data atau informasi yang kompleks, data-data tersebut disusun menjadi beberapa kelompok dengan tipe data yang sejenis disebut table/entity), di mana setiap datanya dapat saling berhubungan satu sama lain atau dapat berdiri sendiri, sehingga mudah diakses. MySQL merupakan database yang awalnya hanya berjalan pada sistem Unix dan Linux. Seiring berjalannya waktu dan banyaknya peminat yang menggunakan database ini, MySQL merilis versi yang dapat diinstal pada hampir semua platform, termasuk Windows. Lisensi dari MySQL adalah freeware. Kita dapat mendownload dan menggunakannya tanpa harus membayarnya. Meskipun kita menjual produk menyertakan software MySQL, kita tidak melanggar hak cipta. Mungkin bagi kita yang baru dengan MySQL akan bingung dengan dua kata "SQL" dan "MySQL". Pertanyaan yang mungkin muncul adalah, apakah SQL itu sebenarnya, dan apa bedanya dengan MySQL? SQL merupakan kependekan dari kata "Structured Query Language". SQL merupakan suatu bahasa permintaan yang terstruktur yang melekat pada satu database atau SMBD tertentu, sedangkan MySQL merupakan databasenya. Dengan kata lain, MySQL merupakan SMBD-nya dan SQL adalah perintah atau bahasa yang melekat di dalam SMBD tersebut. [11]