Mobile Application Development (20IT505/JO1A)

By`

K. Bhaskara Rao

Asst. Prof.

Dept. of Information Technology

BEC, Bapatla

2024-25

Mobile Application Development

UNIT-I

- **Hello, Android:-**Android: An Open platform for Mobile development, Android SDK Features, Introducing the Development Framework.
- **Getting Started:-**Developing for Android, Developing for Mobile and Embedded devices.

UNIT-II

- **Creating Applications and Activities:-**Components of an Android Application, Introducing the Application Manifest File, The Android Application Lifecycle, A Closer Look at Android Activities, Creating Activities, The Activity Lifecycle, Activity States Android Application class, Android Activities.
- **Building User Interfaces:-** Fundamental Android UI Design, Android User Interface Fundamentals, Introducing Layouts, Introducing Fragments.

Mobile Application Development

UNIT-III

- **Intents and Broadcast Receivers:-**Introducing Intents, Creating Intent Filters and Broadcast Receivers.
- **Saving State and User Preferences:-**Creating and Saving Shared Preferences, Retrieving Shared Preferences Persisting the Application Instance State.
- **Creating and Using Databases:-** Working with SQLite Databases.

UNIT-IV

- **Content Providers:-** Creating Content Providers, Accessing Content Providers, using Native Android Content Providers.
- **Working in the Background:-** Creating and Controlling Services, Binding Services to Activities.
- **Expanding the User Experience:-** Introducing the Action Bar ,Creating and Using Menus and Action Bar Action Items.

Mobile application Development

- ✓ Programming languages for appl. development
- ✓ Procedural languages and Object Oriented
- ✓ Java
- ✓ Android
- ✓ Platform Vendors
- ✓ Mobile Platforms
- ✓ Mobile Applications vs Desktop Applications
- ✓ Mobile device capabilities
- ✓ Mobile Application restrictions

Procedural vs Object Oriented Programming

✓ Procedural Oriented Programming: Focuses on the procedures (functions) (rather than data) and Algorithm needed to perform the task.

- ✓ Program is divided into no. of functions with each function carries a well defined task.
- ✓ Dividing a problem into sub problems and solving simplifies the problem solving.
- ✓ Most functions share data
- ✓ Data moves openly around the functions.

Drawbacks:

❖ No proper security for the data.

❖ Design difficulty as the components function and data structures do not model the real world.

Ex: for UI, thinking for data structures instead of menus and menu items.

❖ Not possible to create new data types. (no extensibility)

Examples : C, BASIC, FORTRAN

Procedural vs Object Oriented Programming

- ✓ Object Oriented Programming: Combines process (function) and data into a unit called an object. Hence, it focuses on objects rather than procedure.
- ✓ program is composed of a collection objects that communicate with each other.

Features of Object Oriented Programming:

- Its emphasis is on data rather than procedure.
- It is based on the principles of inheritance, polymorphism, encapsulation and data abstraction.
- It implements programs using the objects.
- Data and the functions are wrapped into a single unit called class so that data is hidden and is safe from accidental alternation.
- Objects communicate with each other through functions.
- New data and functions can be easily added whenever necessary.

Examples: C++, JAVA, C#, VISUAL BASIC

JAVA

Advantages:

- Simple
- Platform independent
- Object Oriented
- Facilitating modular design
- Distributed
- Secure
- Strong multi threading support
- Strictly typed language promoting robust code development

Dis advantages:

- runs slower compared to C, C++ (because of interpreter)
- Strictly Typed

JAVA

- ✓ Using Java we can create 2 types of Applications

  1. Stand Alone Application that runs normally under JVM
  2. Applets (Java executable) that runs in a browser

DIFFERENCES:

1. Applets cannot access the local file system unless digitally signing the .jar file containing it.

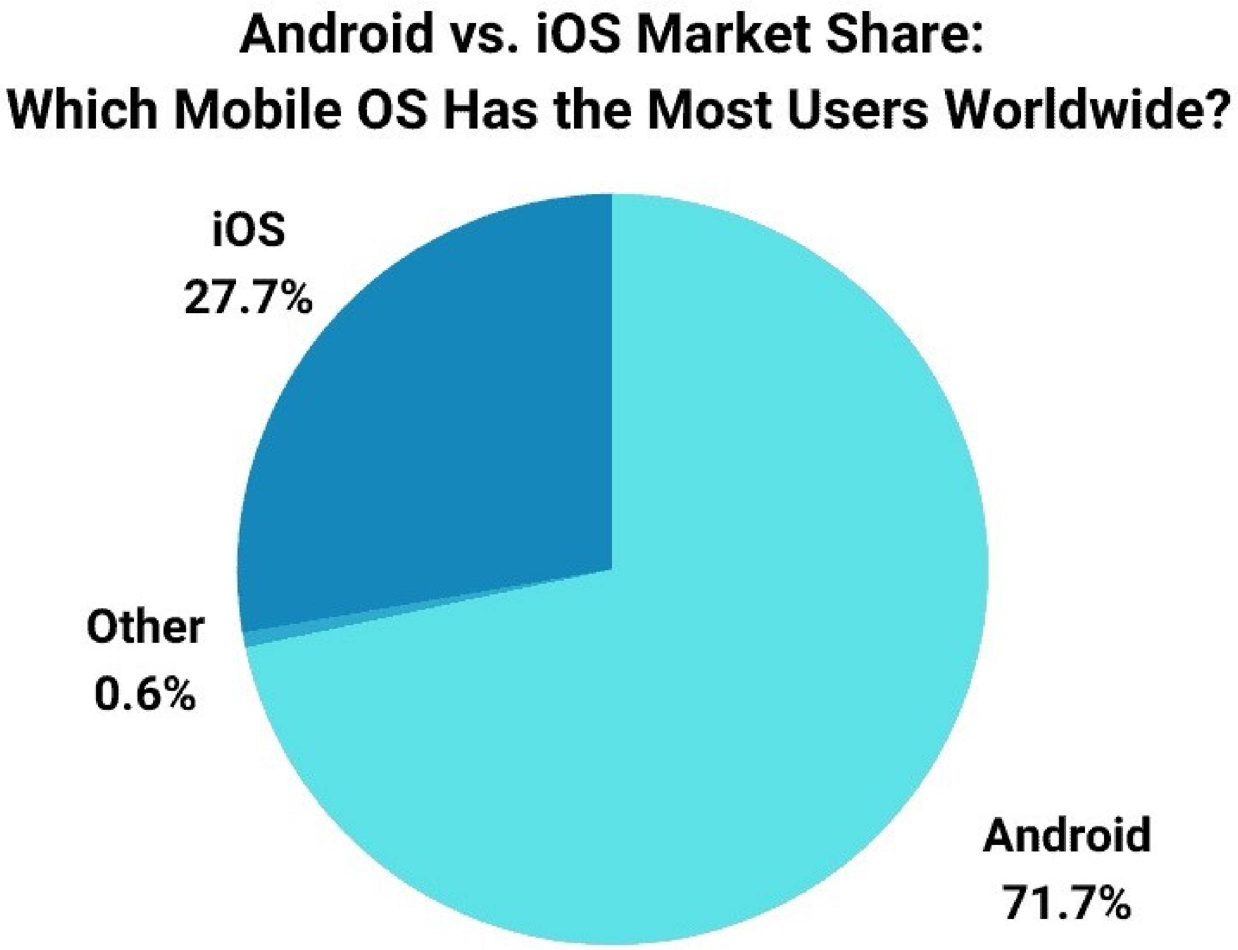
Android

- *Android is* an open source software stack that includes the **operating system, middleware, and key applications along with a set of API libraries** for writing mobile applications that can shape the look, feel, and function of mobile handsets.
- Android is an open source software stack produced and supported by the Open Handset Alliance (OHA) and designed to operate on any handset that meets the requirements.
- OHA (Open Handset Alliance) is a collection of more than 30 technology companies including hardware manufacturers, mobile carriers, and software developers. (Motorola, HTC, Qualcomm etc…)
- Android applications are written using Java language.

Native Android Applications:

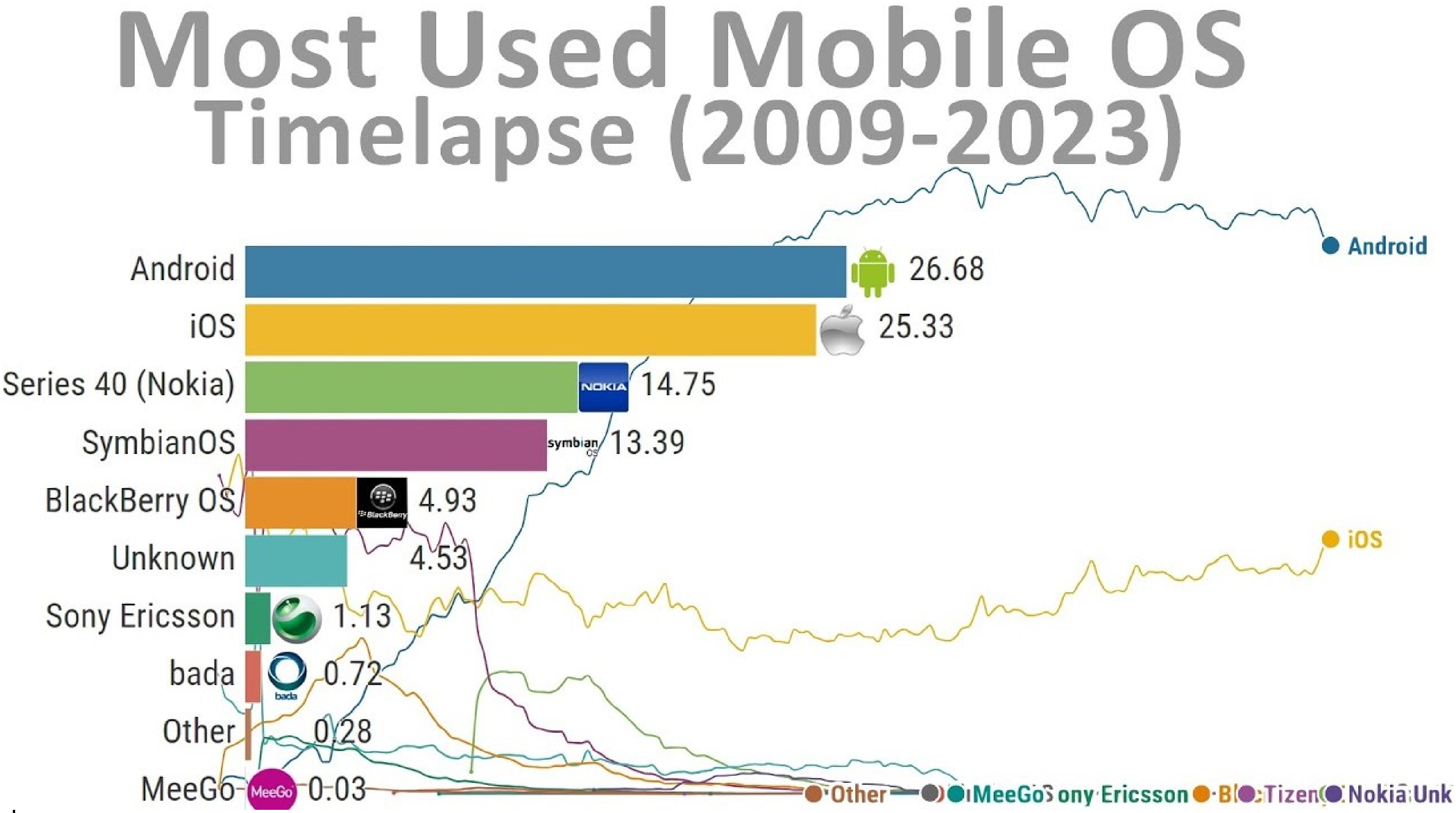
- An email client
- SMS management application
- Personal information mgt. suite (contacts list, calendar, etc…)
- Google maps, web browser, messaging app, music player, play store etc..

Android vs iOS



Android vs. iOS Market Share:
Which Mobile OS Has the Most Users Worldwide?

Mobile OS usage world wide



Mobiles with different OS

PDA (Palm OS)

Nokia E60 (Symbian)

iPhone 5 (iOS)

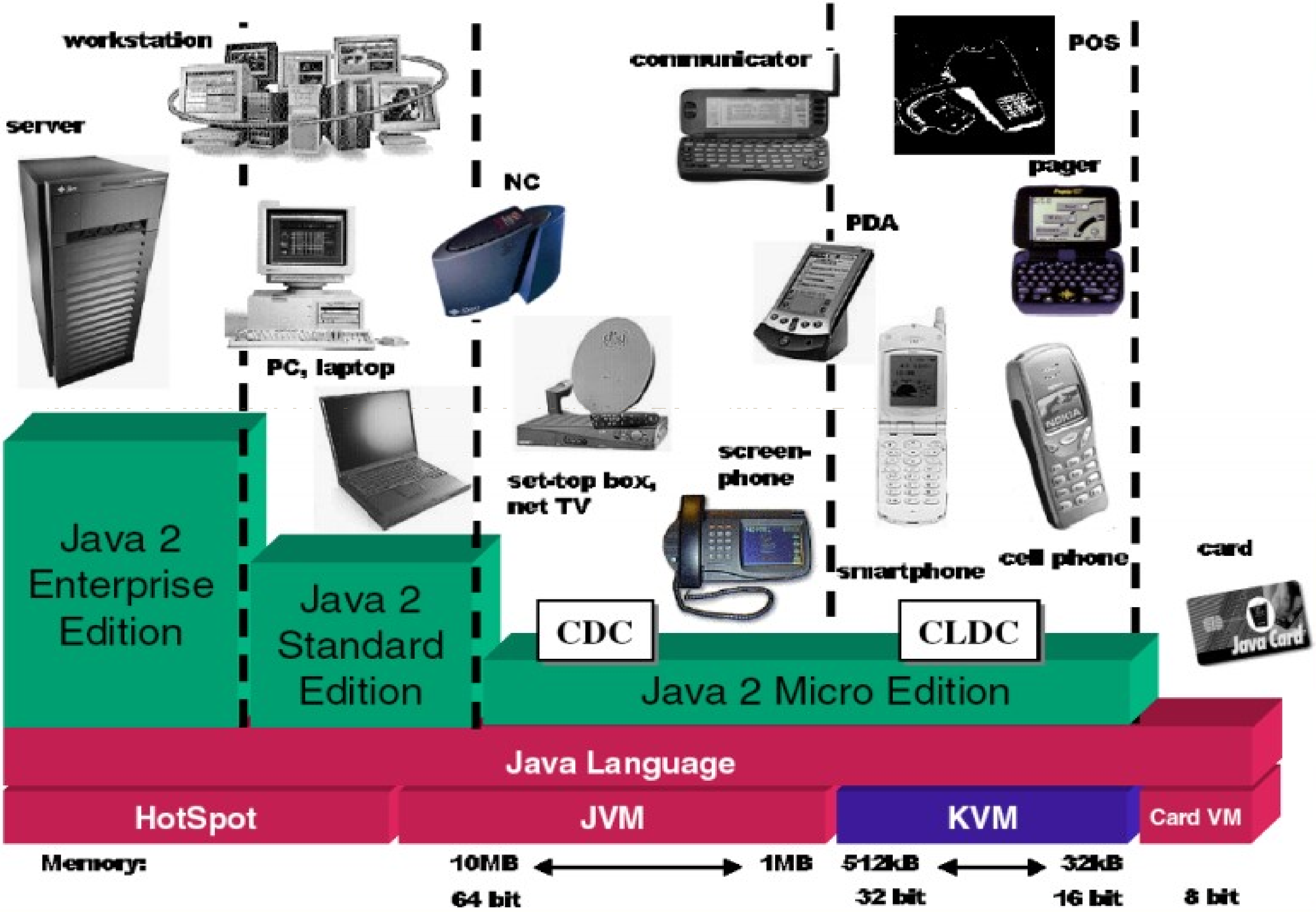
BlackBerry Q10

HTC 7 Pro

(Windows Phone)

galaxy s4 (Android)

Java flavors



Android

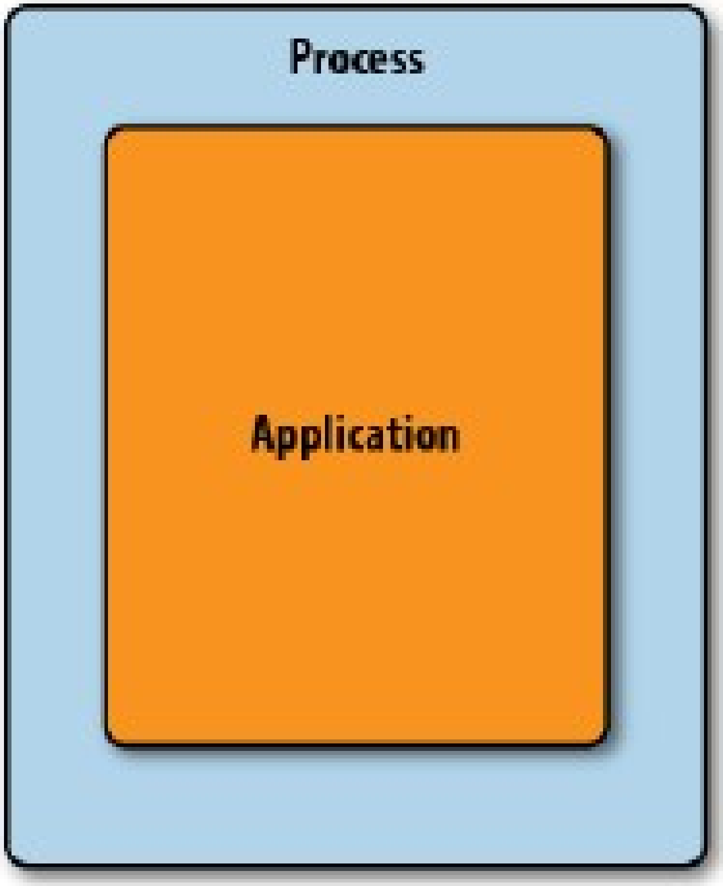
- Android applications are written using JAVA/Kotlin
- Android applications run using a custom virtual machine called 'Dalvik' (a register based VM) rather than a traditional JVM.
- Each Android application runs in a separate process within its own Dalvik instance.
- Android runtime take care of memory and process management.
- Android SDK **includes APIs, tools for development, debugging & testing of android applications, documentation, sample code.**
- Dalvik VM uses the device's underlying Linux kernel to handle low-level functionality including security, threading, and process and memory management.
- **Android APIs feature**
  - hardware access,
  - video recording,
  - location-based services,
  - support for background services,
  - map-based activities,
  - relational databases, (SQLite)
  - inter-application communication,
  - Bluetooth, NFC, and 2D and 3D graphics.

Android benefits

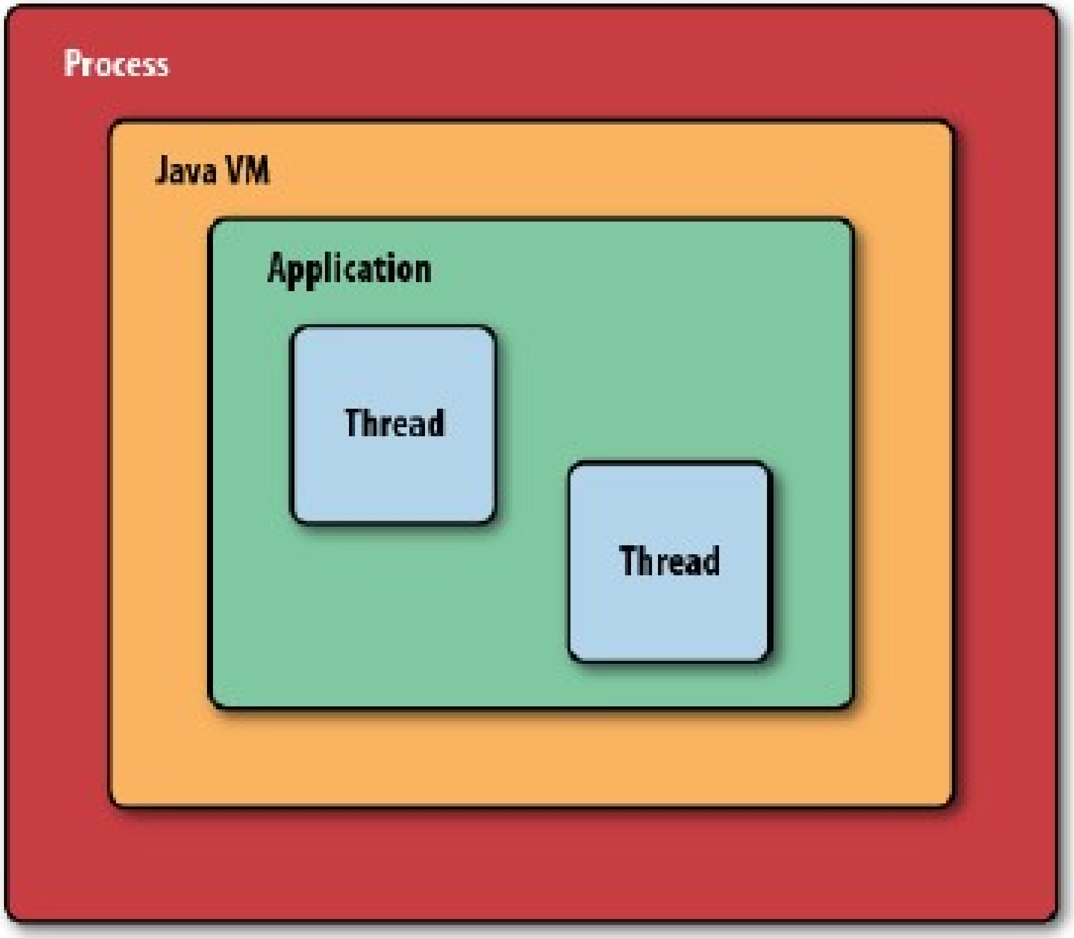
Benefits:

- Powerful SDK
- Open Source
- Excellent documentation
- Over 6 lakhs premium applications built for this platform.
- Integration of Google products and services into the OS. (Calendar, Google Maps, Google Drive, Gmail etc..)
- Ability to run multiple applications at the same time.
- Multi language support
- Multimedia, Tethering support
- Support of 2D, 3D graphics
- Better Notification system

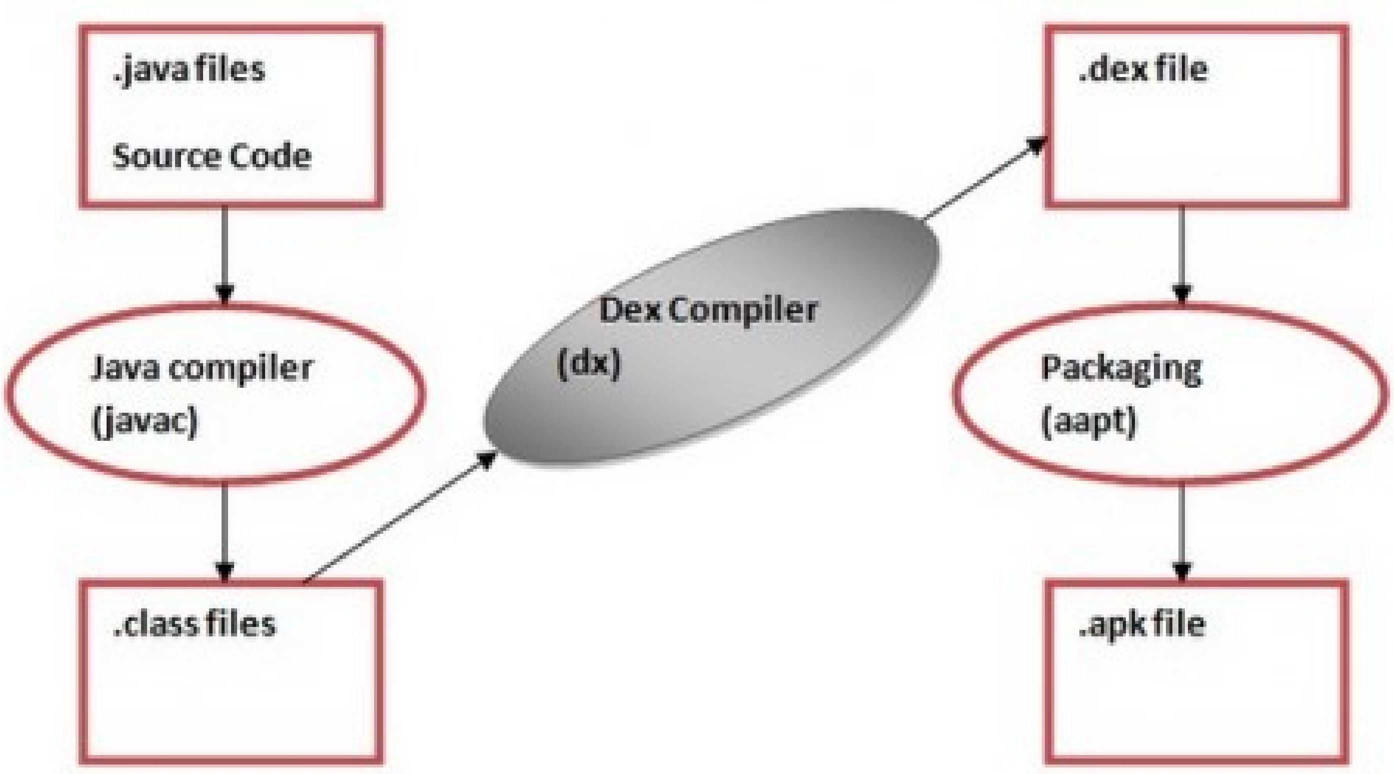
Traditional Programming Model



Android Programming Model



Compiling & Packaging of Java files



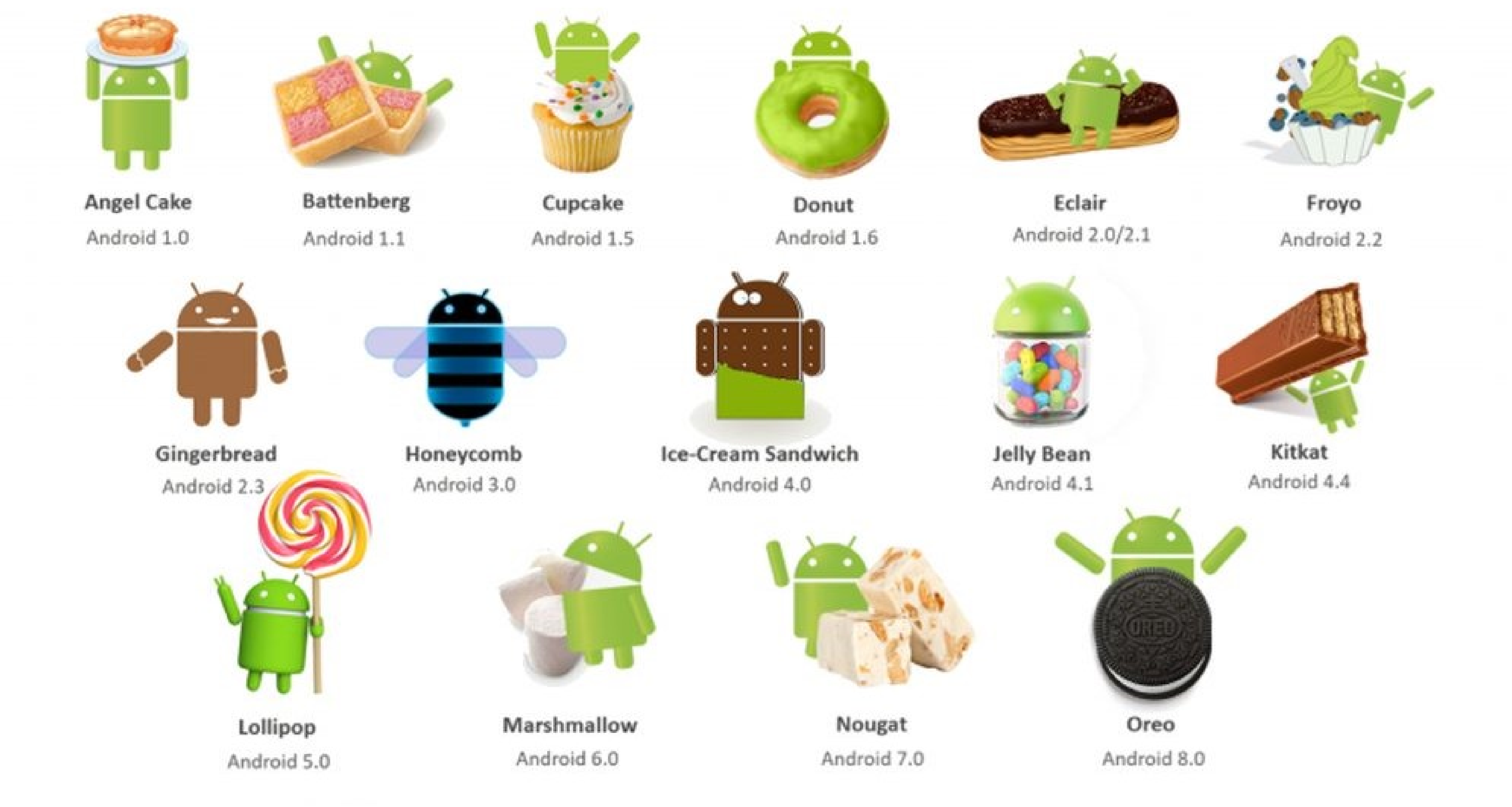
- The **javac** tool compiles the java source file into the class file.
- The **dx** tool takes all the class files of the application and generates a single .dex file. It is a platform-specific tool.
- The Android Assets Packaging Tool (**aapt**) handles the packaging process.

Android OS Versions

- Android 1.0
- Android 1.1
- Android 1.5 (Cupcake)
- Android 1.6 (Donut)
- Android 2.0 (Éclair)
- Android 2.1 (Éclair)
- Android 2.2 (Froyo)
- Android 2.3 (Gingerbread)
- Android 3.0 (Honeycomb)
- Android 4.0 (Ice-cream sandwich)
- Android 4.1 (Jelly Bean)
- Android 4.2 (Jelly Bean)
- Android 4.3 (Jelly Bean)
- Android 4.4 (KitKat)
- Android 5.0 (Lollipop)
- Android 5.1 (Lollipop)
- Android 6.0 (MarshMallow)
- Android 7.0 (Nougat)
- Android 7.1 (Nougat)
- Android 8.0 (Oreo)
- Android 8.1 (Oreo)

Android 9.0 (Pie)	- Aug. 2018
Android 10 (Quince Tart)	- Sep 2019
Android 11 (Red Velvet Cake)	- Sep. 2020
Android 12 (Snow Case)	- Oct. 2021
Android 13 (Tiramisu)	- Aug. 2022
Andriod 14 (Upside Down Cake)	- Q3 2023
Android 15 (Vanilla Ice Cream)	- Q3 2024

Android OS Versions



Android OS Versions

Android 8.0 Oreo:

Android 9.0 Pie (2018)

Android 8.1 Oreo:

Android 10 (2019)

Android OS Versions

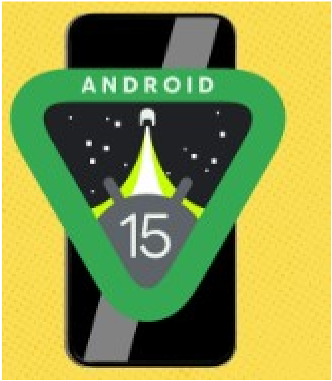
Android 11 (2020)

Android 12(2021)

Android 13(2022)

Android 14(2023)

Android 15 (2024)



Differences between Mobile App and Desktop App

For Mobiles environment:

- Same programming framework, different tools and hardware.
- Network coverage is not reliable
- Usage Patterns are different
  - ✓ Usage of Appl. in office, while moving, in sun light, noisy atmosphere.
  - ✓ different input and output methods.
  - ✓ mostly operated by one hand.
- Personalization of Appl.
- Small screen size (small UI portion is visible at one time)
- Different screen resolutions across devices.
- Limited CPU and GPU performance.
- Limited Memory.
- Mobile apps can be quit and restarted at any time (ex: when call and SMS arrived).

Mobile Device Capabilities

- OS
- Wireless Networking (Bluetooth, WiFi)
- Java Capability (Most devices support Java Micro Ed.)
- Browsers
- Location Capability (GPS)
- Keyboard
- Local Storage
- Extendible memory
- Camera
- Accelerometer , Light Sensor, Proximity Sensor, Gyroscope, E-compass etc…
- Multi Touch Screen
- Multimedia capability
- SMS, MMS, Voice

Android OS History

- Android started as a separate company in 2003. It was run by Andy Rubin
- Google bought Android in 2005
- In 2008, Google partnered with T-Mobile to launch the first-ever Android smartphone, the G1. (Android OS 1.1)
- Motorola Droid phone in 2009 used Android OS 2.0.
- Google launched its smartphone, the Nexus One, in January 2010.
- In 2010, Samsung introduced Galaxy S smart phones (Gingerbread Android OS)
- Android overtook BlackBerry in U.S. market share in the spring of 2011.
- In 2011, A new Version of Android : Ice Cream Sandwich was released.
- In 2012, Jelly Bean version was released.
- 2013 -- Kitkat
- 2014 – Lollipop
- 2015 – Marshmallow
- 2016 – Nougat
- 2017 – Oreo
- 2018 – Android P
- -- Android Q

App Stores

iOS:

- Apple's App Store

Android OS:

- Several stores
  - Android Market, Amazon Appstore for Android, GetJar
- On the Web
- Google Play Store

Windows Phone OS:

- Windows Phone Store

Android devices in market

- Smartphones
- Tablets
- E-reader devices
- Netbooks (lower weight, size, cost laptops- 1kg, 5” screen, no dvd)
- MP4 players
- Internet TVs
- Gaming devices
- GPS receivers
- Home Audio
- Phones
- Photo frames
- Printers
- Ultra mobile PCs
- Vehicles

Android

- ADT plugin integrates the developer tools.
- ADT plugin makes creating, testing and debugging your applications faster and easier.
- ADT integrates the following into Eclipse:
  - Android project wizard
  - Resource editors, form based layout
  - Building of projects
  - Conversion of .dex to .apk
  - Installation of packages on to dalvik vm
  - Emulator
  - DDMS (Dalvik Debug Monitoring Service)
  - Access to device’s emulator’s file system.
  - Runtime debugging
  - Android console and log outputs

Android

- The Dalvik VM executes Dalvik executable files, a format optimized to ensure minimal memory footprint.
- The .dex executables are created by transforming Java language compiled classes using the tools supplied within the SDK.
- Android libraries:

  android.util

  android.os

  android.graphics

  android.text

  android.database

  android.content

  android.view

  android.widget

  com.google.android.maps

  android.app

  android.provider

  android.telephony

  android.webkit

Mobile Platforms

- BlackBerry by RIM (Research In Motion) → BlackBerry Q10, Z10, Q5 By BlackBerry Ltd. for its devices (8100 series, 8200 series, BlackBerry Curve etc..
- Palm OS (Garnet OS) →by Palm Inc. → for PDA
- Windows Mobile → HTC 7 Pro, HTC 7 Surround, LG Optimus, Samsung Focus, LG Quantum, Nokia Lumia 820, 825, 720, 620, HTC 8S, 8X, Samsung ATIV Odyssey.
- Symbian → Nokia E60, Nseries, Sony Ericsson P900, P800 Nokia 7650, Nokia N-Gage, Nokia 9500
- iOS → Apple phones (iPhone3G, 4, 4S, 5, iPod Touch, iPad Tablet, iPod Touch).
- Android → Samsung galaxy, HTC dream, Motorola droid, Sony Ericsson Xperia X3, HTC dragon,
- Proprietary Systems

Constraints and Challenges for Mobile Apps

Major Challenges:

- Designing and developing multiple versions of same application to run on a wide variety of platforms.
- Small screen size
- Different screen sizes, resolutions and orientations
- Limited input devices
- Different interaction methods (keypad, stylus, touch screen)
- Text input is difficult
- Limited battery life and limited processing power
- Limited storage
- Unpredictable network connection, limited coverage, lower network bandwidth and long latency times
- Varying usage environments like ambient lighting, noise, temperature
- For developers it is challenging to learn different technologies, tools and APIs.
- Rapid changes in technology, OS versions.

Mobile Appl. Dev. Tools

Symbian → for C/C++ → Nokia Carbide dev tools for
Symbian OS C++, SymbDev,

for Java → Nokia Carbide dev tools for Java.

Windows Mobile → for C/C++ → Visual Studio 2005 +

and .NET → Windows Mobile SDK

Palm OS → for C/C++ → Garnet OS Dev. Suite, PRC Tools

BlackBerry → Java → BlackBerry Java Dev. Env.

Android → Eclipse ADT

All → for C/C++ → Eclipse CDT

for Java → Sun Java Wireless Toolkit

Android features

Features in Android :

- SQLite, a lightweight relational database, is used for data storage purposes.
- Connectivity
- Transp, LTE, and Wi-Fi network support, enabling your app to send and retrieve data across mobile and Wi-Fi networks
- Comprehensive APIs for **location-based services** such as GPS and network based location detection
- Full support for integrating **maps** within the user interface
- Full multimedia hardware arent access to telephony and Internet resources through GSM, EDGE, 3G, 4Gcontrol, including **playback and recording with**

  the camera and microphone
- Media libraries **for playing and recording** a variety of **audio/video or still image** formats
- APIs for using sensor hardware, including accelerometers, compasses, barometers, and fingerprint sensors

Android features

Features in Android :

- Libraries for **using Wi-Fi, Bluetooth, and NFC hardware**
- Shared data stores and APIs for **contacts, calendar, and multimedia**
- Background services and an advanced notification system
- An integrated web browser
- Video calling
- Messaging
- Voice based features - Google **search through voice**
- Accessibility (Text to speech tool)
- Tethering
- Screen Shot
- External Storage
- Mobile-optimized, hardware-accelerated **graphics**, including a path-based 2D graphics library and support for 3D graphics using OpenGL ES 2.0
- Localization through a dynamic resource framework

Android SDK components

- **The Android API Libraries**
- The core of the SDK is the Android API libraries that provide developer access to the Android stack. These are the same libraries that Google uses to create native Android applications.
- **Development tools**
- The SDK includes the Android Studio IDE and

  several other development tools that let you compile and debug your

  applications to turn Android source code into executable applications.
- **The Android Virtual Device Manager and Emulator**
- The Android Emulator is a fully interactive mobile device emulator featuring several alternative skins. The **Emulator** runs within an Android Virtual Device (AVD) that simulates a device hardware configuration.
- **Full documentation**
- The SDK includes extensive code-level reference information detailing exactly what's included in each package and class and how to use them.

Android SDK components

- **Sample code—**The Android SDK includes a selection of sample applications that demonstrate some of the possibilities available with Android, as well as simple programs that highlight how to use individual API features.
- **Online support—Android has vibrant developer communities on most** online social networks, Slack, and many developer forums. Stack Overflow www.stackoverflow.com /questions/tagged/android) is a hugely popular destination for Android questions and a great place to find answers to beginner questions

Popular development environments

- Java ME
- Windows Mobile (.NET Compact Framework (C++, C#, VB.NET))
- Windows Phone 7 (Silverlight and XNA)
- Qualcomm’s BREW (C or C++)
- Symbian (C++)
- BlackBerry (Java)
- Android (Java)
- iPhone (Objective-C)

Android Architecture

android architecture or **Android software stack** is categorized into five parts:

- linux kernel
- native libraries (middleware),
- Android Runtime
- Application Framework
- Applications

1) Linux kernel

It is the heart of android architecture that exists at the root of android architecture. Linux kernel is responsible for device drivers, power management, memory management, device management and resource access.

2) Native Libraries

- On the top of linux kernel, their are Native libraries such as WebKit, OpenGL, FreeType, SQLite, Media, C runtime library (libc) etc.
- The WebKit library is responsible for browser support, SQLite is for database, FreeType for font support, Media for playing and recording audio and video formats.

Android Architecture

3) Android Runtime

- In android runtime, there are core libraries and DVM (Dalvik Virtual Machine) which is responsible to run android application. DVM is like JVM but it is optimized for mobile devices. It consumes less memory and provides fast performance.

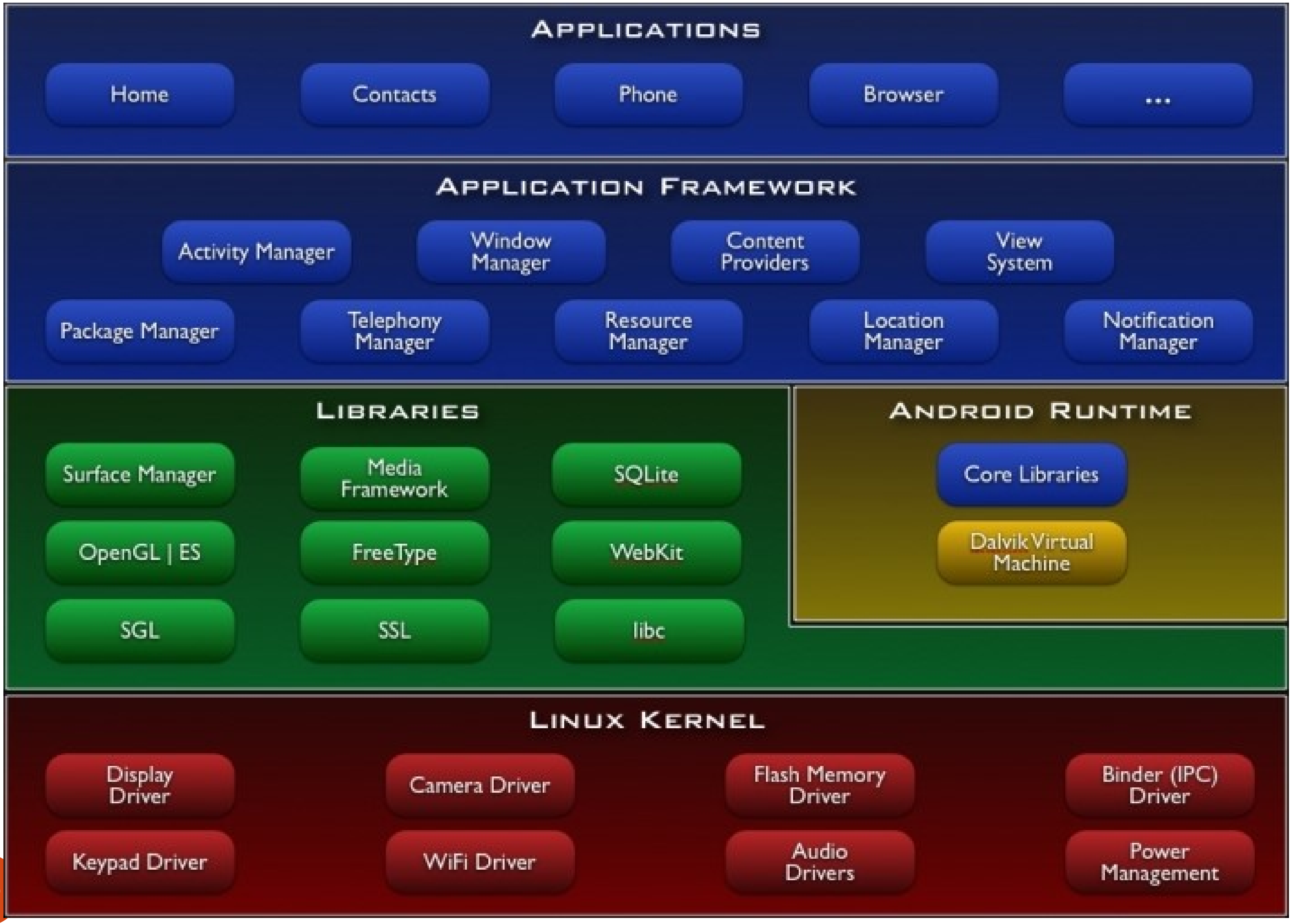
4) Android Framework

- On the top of Native libraries and android runtime, there is android framework. Android framework includes **Android API's** such as UI (User Interface), telephony, resources, locations, Content Providers (data) and package managers. It provides a lot of classes and interfaces for android application development.

5) Applications

- On the top of android framework, there are applications. All applications such as home, contact, settings, games, browsers are using android framework that uses android runtime and libraries. Android runtime and native libraries are using linux kernal.

Android Architecture



APPLICATIONS

Home | Contacts | Phone | Browser | ...

APPLICATION FRAMEWORK

Activity Manager | Window Manager | Content Providers | View System

Package Manager | Telephony Manager | Resource Manager | Location Manager | Notification Manager

LIBRARIES

Surface Manager | Media Framework | SQLite

OpenGL | ES | FreeType | WebKit

SGL | SSL | libc

ANDROID RUNTIME

Core Libraries

Dalvik Virtual Machine

LINUX KERNEL

Display Driver | Camera Driver | Flash Memory Driver | Binder (IPC) Driver

Keypad Driver | WiFi Driver | Audio Drivers | Power Management

Android Software Stack

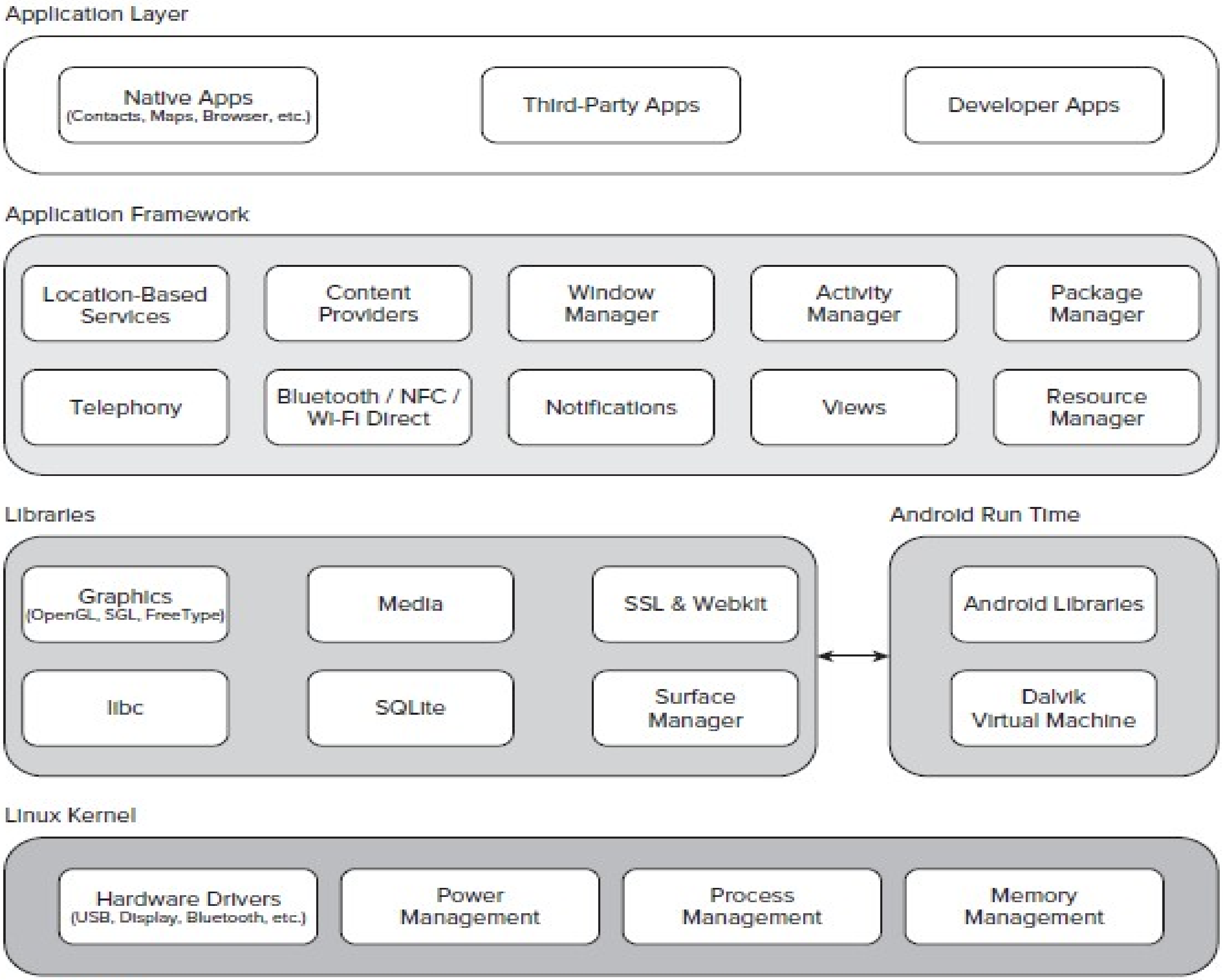


FIGURE 1-1

Android architecture

Linux kernel functionalities:

- Linux Kernel is heart of the android architecture. It manages all the available drivers such as display drivers, camera drivers, Bluetooth drivers, audio drivers, memory drivers, etc. which are required during the runtime. The Linux Kernel will provide an abstraction layer between the device hardware and the other components of android architecture. It is responsible for management of memory, power, devices etc. The features of Linux kernel are:
- **Security:** The Linux kernel handles the security between the application and the system.
- **Memory Management:** It efficiently handles the memory management thereby providing the freedom to develop our apps.
- **Process Management:** It manages the process well, allocates resources to processes whenever they need them.
- **Network Stack:** It effectively handles the network communication.
- **Driver Model:** It ensures that the application works properly on the device and hardware manufacturers responsible for building their drivers into the Linux build.

✓

Android architecture

Platform Libraries:

- The Platform Libraries includes various C/C++ core libraries and Java based libraries such as Media, Graphics, Surface Manager, OpenGL etc. to provide a support for android development.
- **Media** library provides support to play and record an audio and video formats.
- **Surface manager** responsible for managing access to the display subsystem.
- **SGL** and **OpenGL** both cross-language, cross-platform application program interface (API) are used for 2D and 3D computer graphics.
- **SQLite** provides database support and **FreeType** provides font support.
- **Web-Kit** This open source web browser engine provides all the functionality to display web content and to simplify page loading.
- **SSL (Secure Sockets Layer)** is security technology to establish an encrypted link between a web server and a web browser.

Android architecture

- **Android Runtime: Dalvik Virtual Machine :**

✓ It is a Register based Virtual Machine.

✓ It is optimized for low memory requirements.

✓ It has been designed to allow multiple VM instances to run at once.

✓ Is a Custom VM designed for mobiles.

- Performs process mgt., memory mgt., threading and security.
- Dalvik VM executes Dalvik executables (.dex extension)

Core Libraries: Core libraries are provided that enable developers to write Android apps using the Java language.

Application Framework Layer :

- Our applications directly interact with these blocks of the Android architecture.
- Provides classes used to create android applications.
- These programs manage the basic functions of phone like resource management, voice call management etc.

Android

- c/c++ libraries:
  - openGL
  - FreeType
  - SGL
  - libc
  - SQLite
  - SSL

Advanced Android libraries:

- android.location
- android.media
- android.opengl
- android.hardware
- android.bluetooth
- android.net.wifi

Android architecture

Application Framework:

Activity Manager: Manages the activity life cycle of applications. To understand the Activity component in Android

Content Providers: Manage the data sharing between applications.

Telephony Manager: Manages all voice calls. We use telephony manager if we want to access voice calls in our application.

Location Manager: Location management, using GPS or cell tower

Resource Manager: Manage the various types of resources we use in our Application.

Window Manager: Manages Windows. Decides which windows are visible, how they are laid on the screen, opening, closing windows, screen rotations etc..

Application Layer : Standard applications that come with OS are available in this layer.

Ex : SMS Client App
Web browser
Dialer, music, gallary,
Calendar, Calculator etc…

Android

- Activity is the base class for the visual, interactive components of your application; it is roughly equivalent to a Form in traditional desktop development.
  - Public class HelloWorld extends Activity{

    ```
        public void onCreate(Bundle icicle) {
                super.onCreate(icicle);
                setContentView(R.layout.main);
        }
    }
    ```

- Visual components are called views. They are similar to controls in windows apps.
- The resources for an Android project are stored in the res folder of your project hierarchy, which includes drawable, layout, and values subfolders

Android

activity_main.xml file:

```
<RelativeLayout >
 <TextView
     android:id=”@+id/myTextView”
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="@string/hello_world" />
</RelativeLayout>
```

Accessing control in code: (MyActivity.java):

1. Add id attribute to them in the xml file (activity_main.xml)
2. Use the findViewById( ) method to return a reference to the control.

```
TextView myTextView = (TextView)findViewById(R.id.myTextView);
```

Structure of an Android Application

src folder : Contains the java source files for your project

Contains MainActivity.java, the source file for your appl.

Android 4.2.2 library : contains android.jar, which contains all the class libraries needed for your appl.

gen folder: contains R.java → a compiler-generated file which references all the resources found in your project.

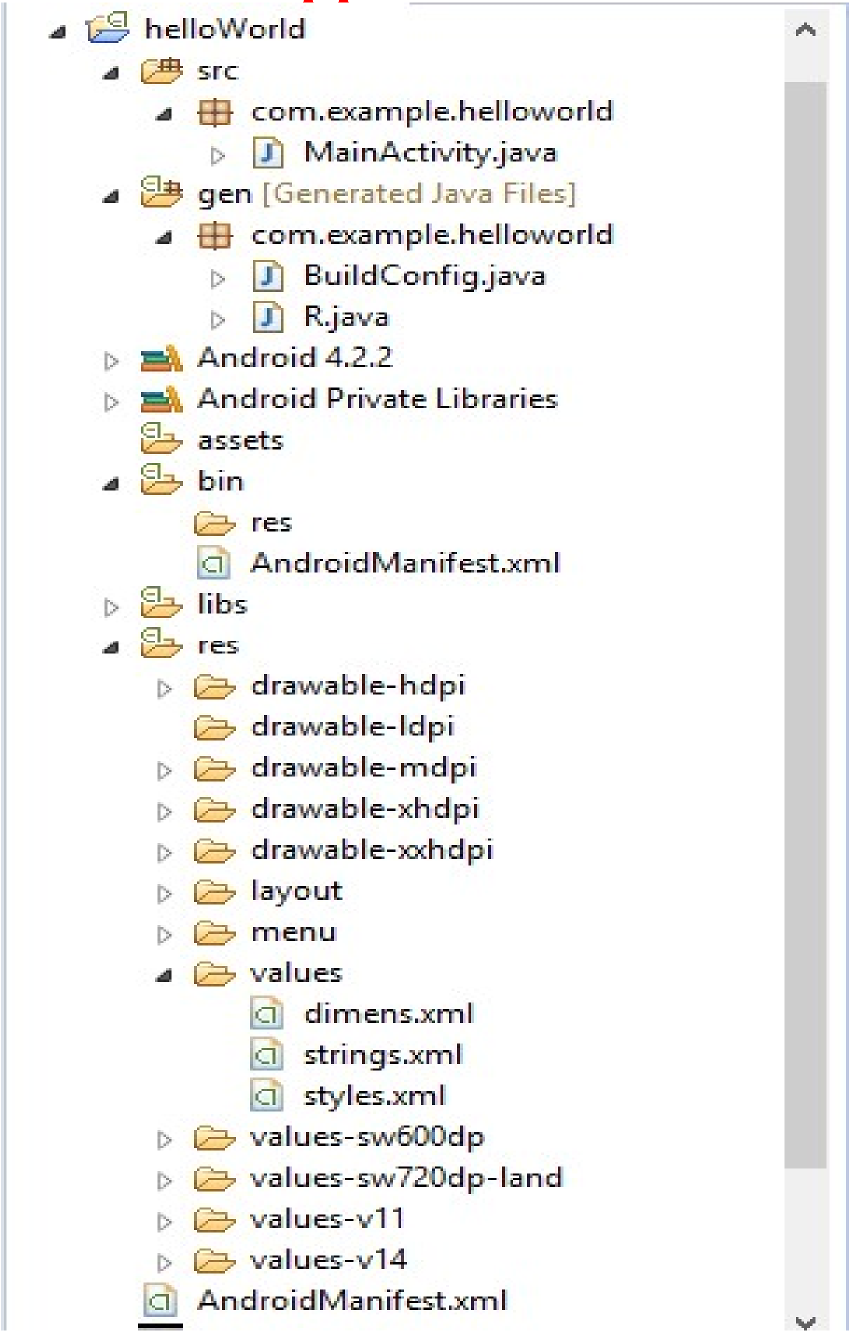
assets : contains all the assets (html, text files, databases)

Used by your project.

res: folder which contains all the resources used in your prj. Contains drawable, values, layout folders.

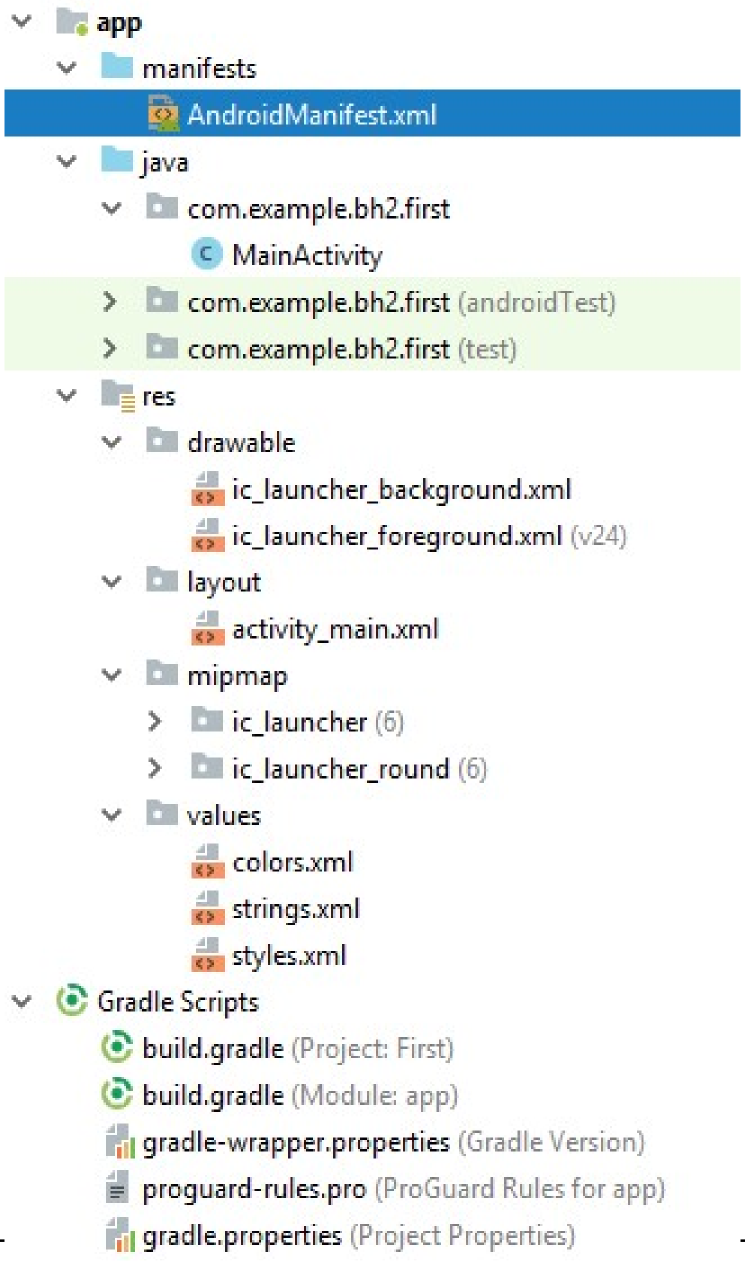
AndroidManifest.xml : specifies permissions for your appl. And other features such as intent-filters and receivers.

Android Application Structure



```
helloWorld
  src
    com.example.helloworld
      MainActivity.java
  gen [Generated Java Files]
    com.example.helloworld
      BuildConfig.java
      R.java
  Android 4.2.2
  Android Private Libraries
  assets
  bin
    res
    AndroidManifest.xml
  libs
  res
    drawable-hdpi
    drawable-ldpi
    drawable-mdpi
    drawable-xhdpi
    drawable-xxhdpi
    layout
    menu
    values
      dimens.xml
      strings.xml
      styles.xml
    values-sw600dp
    values-sw720dp-land
    values-v11
    values-v14
  AndroidManifest.xml
```

Android Studio Project Structure



```
app
  manifests
    AndroidManifest.xml
  java
    com.example.bh2.first
      MainActivity
    com.example.bh2.first (androidTest)
    com.example.bh2.first (test)
  res
    drawable
      ic_launcher_background.xml
      ic_launcher_foreground.xml (v24)
    layout
      activity_main.xml
    mipmap
      ic_launcher (6)
      ic_launcher_round (6)
    values
      colors.xml
      strings.xml
      styles.xml
Gradle Scripts
  build.gradle (Project: First)
  build.gradle (Module: app)
  gradle-wrapper.properties (Gradle Version)
  proguard-rules.pro (ProGuard Rules for app)
  gradle.properties (Project Properties)
```

Android Studio Project Structure

AndroidManifest.xml → It contains information of the package, including components of the application such as activities, services, broadcast receivers, content providers etc.

- It is responsible to protect the application to access any protected parts by providing the permissions
- It also declares the android api that the application is going to use
- It lists the instrumentation classes. The instrumentation classes provides profiling and other informations. These information are removed just before the application is published etc.

java folder → The java folder contains the Java source code files of the application.

Android Studio Project Structure

res folder → Res folder is where all the external resources for the application such as images, layout XML files, strings, animations, audio files etc. are stored.

sub folders:

drawable: This folder contains the bitmap files to be used in the program. There are different folders to store drawables. They are drawable-ldpi, drawable-mdpi, drawable-hdpi, drawable-xdpi etc.

- The folders are to provide alternative image resources to specific screen configurations. Ldpi, mdpi & hdpi stands for low density, medium density & high density screens respectively.

layout: It contains XML files that define the User Interface of the application.

menu: XML files that define menus for the application goes into this folder.

mipmap: The mipmap folders is used for placing the app **icons** only. Any other drawable assets should be placed in the relevant drawable folders.

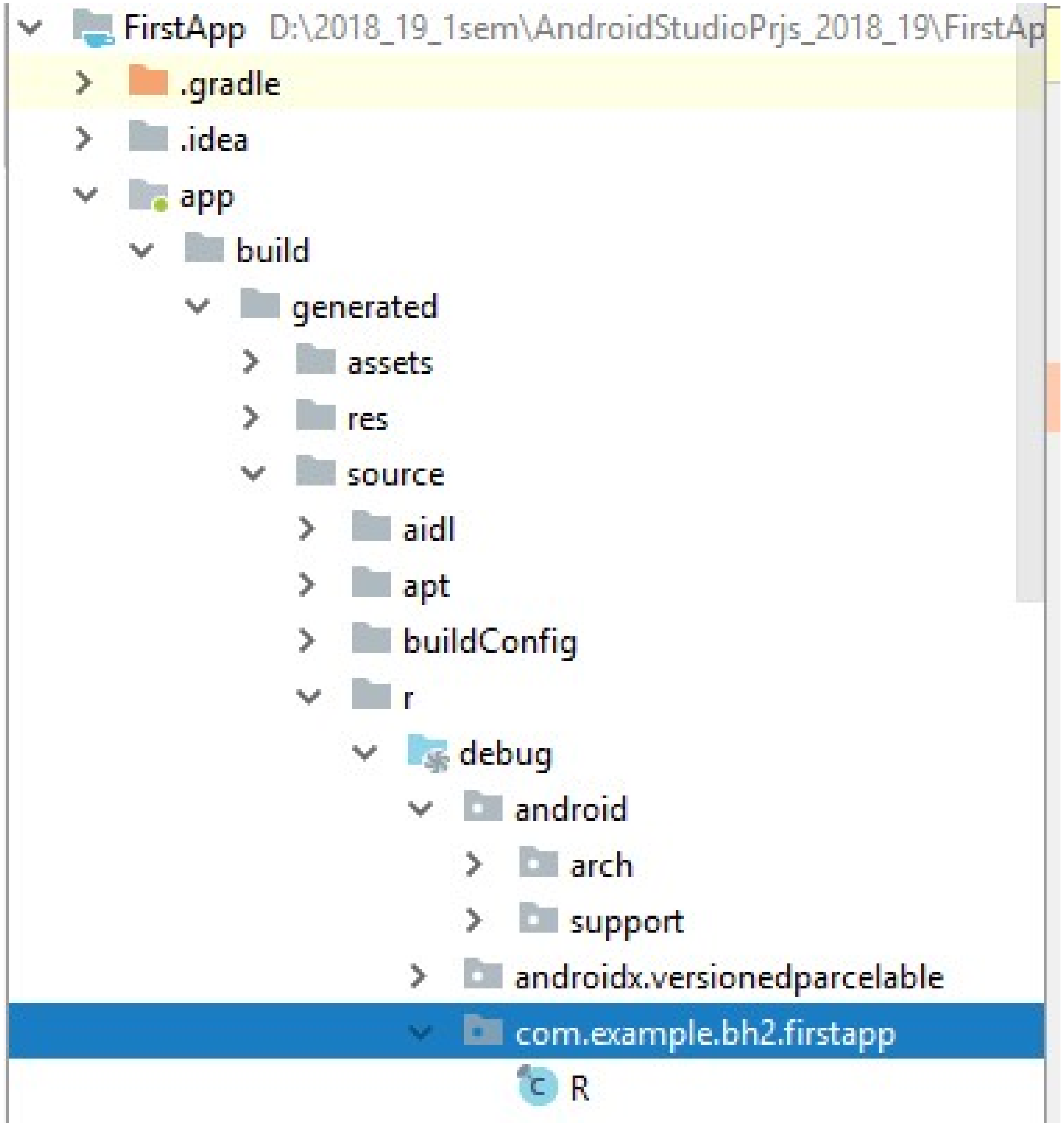
values: XML files that define simple values such as strings, arrays, integers, dimensions, colors, styles etc. are placed in this folder.

Android Studio Project Structure

R.java: A Class R is auto generated in R.java.

- Not visible from inside Android Studio is a generated Java class named R, which can be found in the **app/build/generated/source/r/debug/<pkgname>/** directory of the project. R contains nested classes that in turn contain all the resource IDs for all your resources.
- Every time you add, change or delete a resource, R is re-generated. For instance, if you add an image file named logo.png to the res/drawable directory, Android Studio will generate a field named logo under the drawable class, a nested class in R.
- The purpose of having R is so that you can refer to a resource in your code. For instance, you can refer to the logo.png image file with R.drawable.logo.

R.java



```
FirstApp  D:\2018_19_1sem\AndroidStudioPrjs_2018_19\FirstAp
  .gradle
  .idea
  app
    build
      generated
        assets
        res
        source
          aidl
          apt
          buildConfig
          r
            debug
              android
                arch
                support
              androidx.versionedparcelable
              com.example.bh2.firstapp
                R
```

Android Studio Project Structure

Gradle Scripts: Gradle scripts are used to automate tasks. For the most part, Android Studio performs application builds in the background without any intervention from the developer. This build process is handled using the Gradle system, an automated build toolkit.

Hardware imposed design considerations

Compared to laptops mobiles have relatively:

- Low processing power
- Limited RAM
- Limited permanent storage capacity
- Small screens with low resolution
- Higher costs associated with data transfer
- Slower data transfer rates with higher latency
- Less reliable data connections
- Limited battery life

Practical design considerations

- **Performance** → Be fast and efficient.

  At a time when 2 to 4GB of memory is standard for most desktop and server rigs, typical smart phones feature approximately 200MB of SDRAM. With memory such a scarce commodity, you need to take special care to use it efficiently.

- **Responsiveness** → Android takes responsiveness seriously. It Activity Manager or Window Manager detects an unresponsive application, it will display "not responding" message.
  - using worker threads and services to perform lengthy tasks.

Android monitors two conditions to determine responsiveness:

1. An application must respond to any user action, such as keypress or screen touch, within **5 seconds.**
2. A broadcast receiver must return from onReceive handler within **10 seconds.**

Practical design considerations

- **Freshness →** While designing application it is critical that you consider that how often you will update the data it uses, minimizing the time users are waiting for refreshes or updates, while limiting the effect of these background updates on the battery life.

- **Security →** Access to network and hardware causes security problems. Device security is most important one.
  - The Android security model sandboxes each application and restricts access to services and functionality by requiring applications to declare the permissions they require. During installation users are shown the application's required permissions before they commit to installing it.

- **Seamlessness →** A consistent user experience in which applications start, stop and transition instantly without perceptible delays.

Building blocks of Android application

- Activities → Your application’s presentation layer
- Services → Invisible workers of your application
- Content Providers → A shared data store
- Intents → To send broadcast messages to an activity or service.
- Broadcast Receivers → Intent broadcast consumers
- Notifications → Notifications let you signal users without stealing focus or interrupting their current Activities.

Widgets → Visual application components that are typically added to the device HOME screen.

Building Blocks of Android Application

Activity is a window that contains the UI of your application.

- It is analogue to the window or dialog in a desktop appl.
- An application can have 0 or more activities

Content Provider :

Provides a level of abstraction for any data stored on the device that is accessible by multiple applications.

Intents : Intents are system messages notifying applications of various events (SMS arrived, SD card removed, activity launched)

✓ Intent can be used to launch other activities.

Services : activities, intents and content providers are short lived. Services runs longer independent of activities .

- Broadcast Receivers: Intent listeners. Broadcast Receivers enable your application to listen for Intents that match the criteria you specify.
- Widgets: Visual application components that are typically added to the device home screen. Widgets enable you to create dynamic, interactive application components for users to embed on their home screens.

Types of Android Applications

- Foreground Applications → Having user interaction.

  Ex: games
- Background Applications → An application with limited interaction. Mostly run in the background.

  Ex: Alarm Clock
- Intermittent Applications → Email & Media Players, chat applications.
  - These applications are generally union of visible activities, invisible background services and broadcast receivers.
- Widgets and Live Wallpapers → Is a small control of the application placed on the home screen.
  - Allows you to put favourite applications on home screen in order to quickly access them.
  - Widget only applications are commonly used to display dynamic information, such as battery levels, weather forecasts, or the date and time.

Exploring Android Studio IDE
Android Development Tools

Exploring Android Studio IDE

The Android Studio IDE contains the following features:

- It has a flexible Gradle-based build system.
- It has a fast and feature-rich emulator for app testing.
- Android Studio has a consolidated environment where we can develop for all Android devices.
- Apply changes to the resource code of our running app without restarting the app.
- Android Studio provides extensive testing tools and frameworks.
- It supports C++ and NDK.(Native Dev Kit)
- It provides build-in supports for Google Cloud Platform. It makes it easy to integrate Google Cloud Messaging and App Engine.

1. **The Toolbar →** The **toolbar** lets you carry out a wide range of actions, including running your app and launching Android tools.
2. **Navigation Bar →**The navigation bar helps you navigate through your project and open files for editing.
3. **Editor Window →** The editor window is where you create and modify code.
4. **Tool windows →**Tool windows give you access to specific tasks like project management, search, version control, and more.
5. **Status Bar →** The status bar displays the status of your project and the IDE itself, as well as any warnings or messages.

- ➢ You can organize the main window to give yourself more screen space by hiding or moving toolbars and tool windows.
- ➢ You can also use keyboard shortcuts to access most IDE features.

Exploring Android Studio IDE

Search: At any time, you can search across your source code, databases, actions, elements of the user interface, and so on, by double-pressing the Shift key, or clicking the magnifying glass in the upper right-hand corner of the Android Studio window.

Tool Windows: By default, the most commonly used tool windows are pinned to the tool window bar at the edges of the application window.

- ➢ You can also drag, pin, unpin, attach, and detach tool windows.

Code completion: You can also perform quick fixes and show intention actions by pressing **Alt+Enter**.

1. Basic Complection: Ctrl + Space → Displays basic suggestions for variables, types, methods, expressions, and so on.

2. Smart Completion: Ctrl + Shift + Space →Displays relevant options based on the context.

3. Statement Completion: Ctrl + Shift + Enter → Completes the current statement for you, adding missing parentheses, brackets, braces, formatting, etc.

Exploring Android Studio IDE

Tool Windows:

- Android Studio also includes a number of other windows which, when enabled, are displayed along the bottom and sides of the main window.
- The tool window quick access menu can be accessed by hovering the mouse pointer over the button located in the far left hand corner of the status bar without clicking the mouse button.

Layout Editor:

- We can build the layout quickly by adding different attributes either by hard-code or drag and drop using the layout editor feature of the android studio. The layout editor feature can also be used, to preview the codes that can be seen easily on the visual editor screen and changes can be made accordingly by resizing it dynamically.

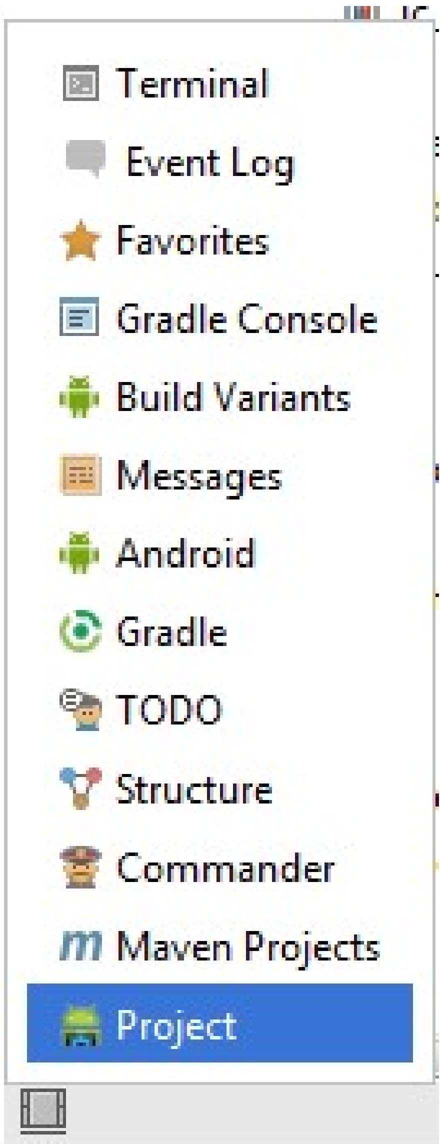
Exploring Android Studio IDE

- **Templates**
- Android also has the feature of templates built-in. If you know to build that accordingly that makes your task easier. Templates can be used to create common Android designs and components.
- **Support KOTLIN**
- KOTLIN can be considered the official language for Android. It runs fast and equivalent to Java. Kotlin can be easily learned and used by java developers as it based on automated Java only.
- **Enabling Integration with Firebase**
- You can get real-time experience with IoT based project development with dynamic upgrades in the application using the firebase feature of an android studio. Chat applications can be created by using firebase connectivity it helps you to give a happy chat experience.
- **Emulator**
- The emulator feature of the android studio provides an emulator that is exactly like the android phones to test how the application looks like in physical devices. It gives real-time experience to Android applications. It allows you to test your applications faster and on different-different configuration devices like tablets, android phones, etc. Basically it helps to run and debug apps in the Android studio.

Exploring Android Studio IDE

- **Colour Previews**
- Android studio helps to see the code XML part in a preview to know that how perfectly we are designing the application according to the need before launching the application. It provides powerful functionality and enhanced features of drag and drops or resizes the application. It contains drag and drop features but not support for every function, that’s why be careful while doing that.
- **Maven Repository**
- In Android Studio, Maven integration of its repository can be done, within SDK manager support libraries of IDE is used. It’s a kind of a repository which is a directory in which various jar files like project jars, Plugin are stored.

Exploring Android Studio IDE



Exploring Android Studio IDE

Some Mostly used Windows:

- **Project** – The project view provides an overview of the file structure that makes up the project allowing for quick navigation between files. Generally, double clicking on a file in the project view will cause that file to be loaded into the appropriate editing tool.
- **Structure** – The structure tool provides a high level view of the structure of the source file currently displayed in the editor. This information includes a list of items such as classes, methods and variables in the file. Selecting an item from the structure list will take you to that location in the source file in the editor window.
- **Favorites** – A variety of project items can be added to the favorites list. Right clicking on a file in the project view, for example, provides access to an Add to Favorites menu option. Similarly, a method in a source file can be added as a favorite by right clicking on it in the Structure tool window. Anything added to a Favorites list can be accessed through this Favorites tool window.
- **Build Variants** – The build variants tool window provides a quick way to configure different build targets for the current application project (for example different builds for debugging and release versions of the application, or multiple builds to target different device categories).

Exploring Android Studio IDE

- **TODO** – As the name suggests, this tool provides a place to review items that have yet to be completed on the project. Android Studio compiles this list by scanning the source files that make up the project to look for comments that match specified TODO patterns. These patterns can be reviewed and changed by selecting the File -> Settings… menu option and navigating to the TODO page listed under IDE Settings.
- **Messages** – The messages tool window records output from the Gradle build system (Gradle is the underlying system used by Android Studio for building the various parts of projects into a runnable applications) and can be useful for identifying the causes of build problems when compiling application projects.
- **Android** – The Android tool window provides access to the Android debugging system. Within this window tasks such as monitoring log output from a running application, taking screenshots and videos of the application, stopping a process and performing basic debugging tasks can be performed.
- **Terminal** – Provides access to a terminal window on the system on which Android Studio is running. On Windows systems this is the Command Prompt interface, whilst on Linux and Mac OS X systems this takes the form of a Terminal prompt.

Exploring Android Studio IDE

Run – The run tool window becomes available when an application is currently running and provides a view of the results of the run together with options to stop or restart a running process. If an application is failing to install and run on a device or emulator, this window will typically provide diagnostic information relating to the problem.

Event Log – The event log window displays messages relating to events and activities performed within Android Studio. The successful build of a project, for example, or the fact that an application is now running will be reported within this window tool.

Gradle Console – The Gradle console is used to display all output from the Gradle system as projects are built from within Android Studio. This will include information about the success or otherwise of the build process together with details of any errors or warnings.

Addition of New Activity as a Code Template

Maven Projects – Maven is a project management and build system designed to ease the development of complex Java based projects and overlaps in many areas with the functionality provided by Gradle. Google has chosen Gradle as the underlying build system for Android development, so unless you are already familiar with Maven or have existing Maven projects to import, your time will be better spent learning and adopting Gradle for your projects. The Maven projects tool window can be used to add, manage and import Maven based projects within Android Studio.

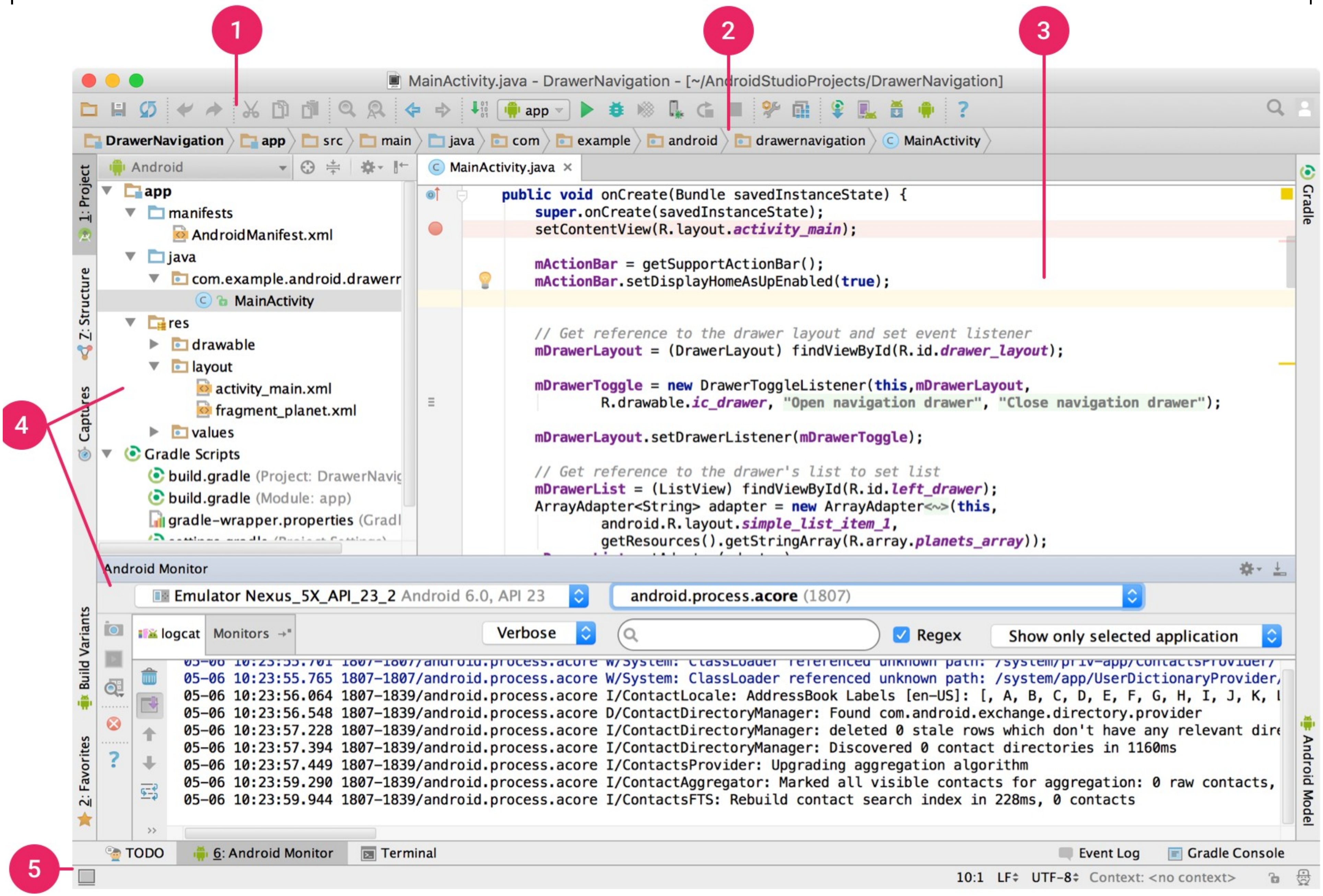
Exploring Android Studio IDE

- **Gradle** – The Gradle tool window provides a view onto the Gradle tasks that make up the project build configuration. The window lists the tasks that are involved in compiling the various elements of the project into an executable application. Right-click on a top level Gradle task and select the Open Gradle Config menu option to load the Gradle build file for the current project into the editor. Gradle will be covered in greater detail later in this book.
- **Commander** – The Commander window tool can best be described as a combination of the Project and Structure tool windows, allowing the file hierarchy of the project to be traversed and for the various elements that make up classes to be inspected and loaded into the editor or designer windows.
- **Memory Monitor** – Connects to running Android applications and monitors memory usage statistics in the form of a real-time graph.
- **Designer** – Available when the UI Designer is active, this tool window provides access to the designer's Component Tree and Properties panels.

Android Studio ID E features

- **Addition of New Activity as a Code Template**
- Yes, Android also has the feature of templates built-in. If you know to build that accordingly that makes your task easier. it has both pros and cons, you don’t find every template in Android Studio. It’s an additional feature which helps the developer to build an application efficiently and effectively which provide effective solutions.
- **Help to Build Up App for All Devices**
- Android studio builds applications for every screen size, for wear and gear devices etc. It also can stimulate the various type of features which a hardware has like GPS location tracker, multi-touch.

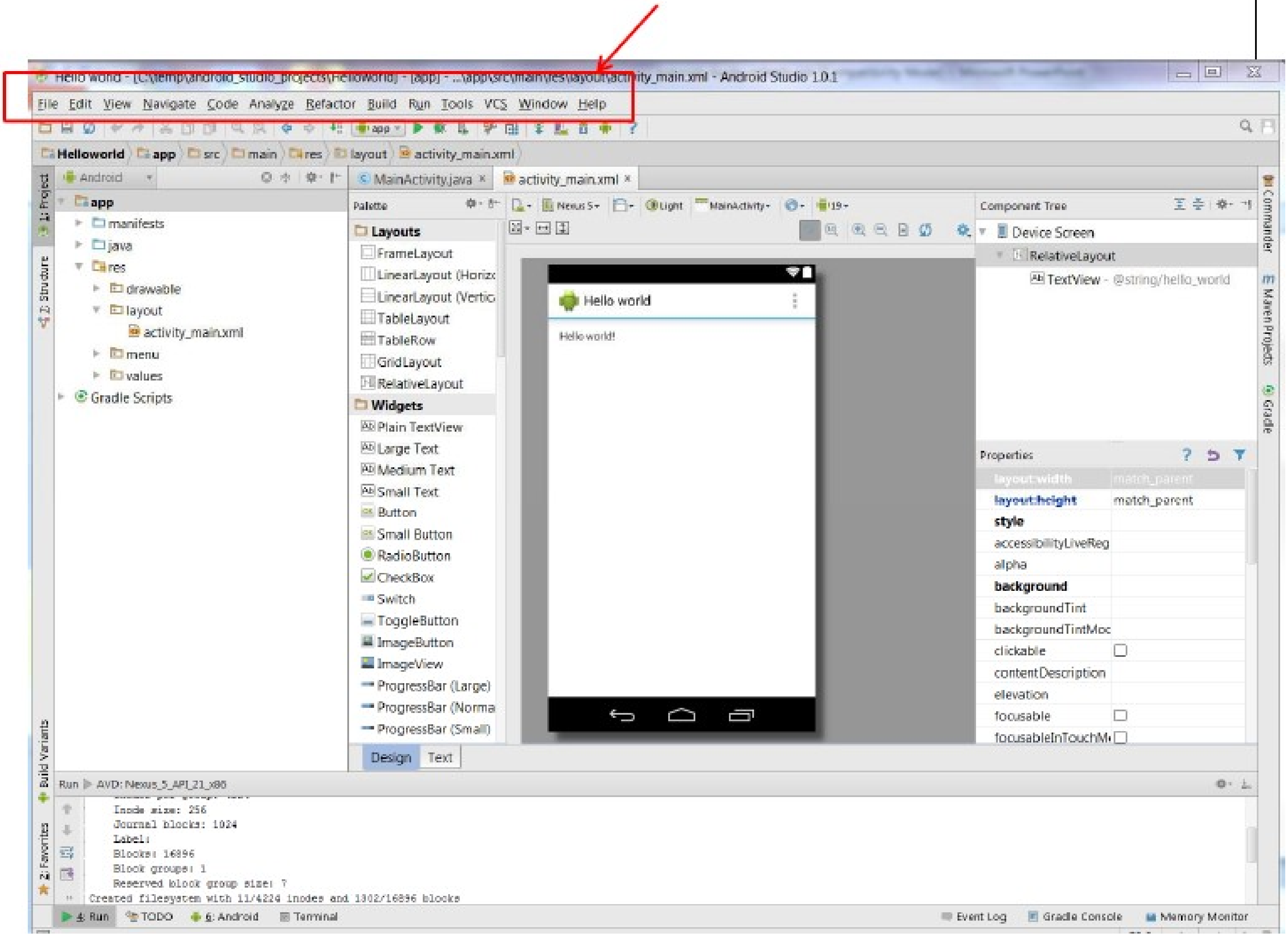
Exploring Android Studio IDE



```java
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    mActionBar = getSupportActionBar();
    mActionBar.setDisplayHomeAsUpEnabled(true);

    // Get reference to the drawer layout and set event listener
    mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);

    mDrawerToggle = new DrawerToggleListener(this,mDrawerLayout,
            R.drawable.ic_drawer, "Open navigation drawer", "Close navigation drawer");

    mDrawerLayout.setDrawerListener(mDrawerToggle);

    // Get reference to the drawer's list to set list
    mDrawerList = (ListView) findViewById(R.id.left_drawer);
    ArrayAdapter<String> adapter = new ArrayAdapter<~>(this,
            android.R.layout.simple_list_item_1,
            getResources().getStringArray(R.array.planets_array));
```

```
05-06 10:23:55.701 1807-1807/android.process.acore W/System: ClassLoader referenced unknown path: /system/priv-app/ContactsProvider/
05-06 10:23:55.765 1807-1807/android.process.acore W/System: ClassLoader referenced unknown path: /system/app/UserDictionaryProvider/
05-06 10:23:56.064 1807-1839/android.process.acore I/ContactLocale: AddressBook Labels [en-US]: [, A, B, C, D, E, F, G, H, I, J, K, L
05-06 10:23:56.548 1807-1839/android.process.acore D/ContactDirectoryManager: Found com.android.exchange.directory.provider
05-06 10:23:57.228 1807-1839/android.process.acore I/ContactDirectoryManager: deleted 0 stale rows which don't have any relevant dire
05-06 10:23:57.394 1807-1839/android.process.acore I/ContactDirectoryManager: Discovered 0 contact directories in 1160ms
05-06 10:23:57.449 1807-1839/android.process.acore I/ContactsProvider: Upgrading aggregation algorithm
05-06 10:23:59.290 1807-1839/android.process.acore I/ContactAggregator: Marked all visible contacts for aggregation: 0 raw contacts,
05-06 10:23:59.944 1807-1839/android.process.acore I/ContactsFTS: Rebuild contact search index in 228ms, 0 contacts
```

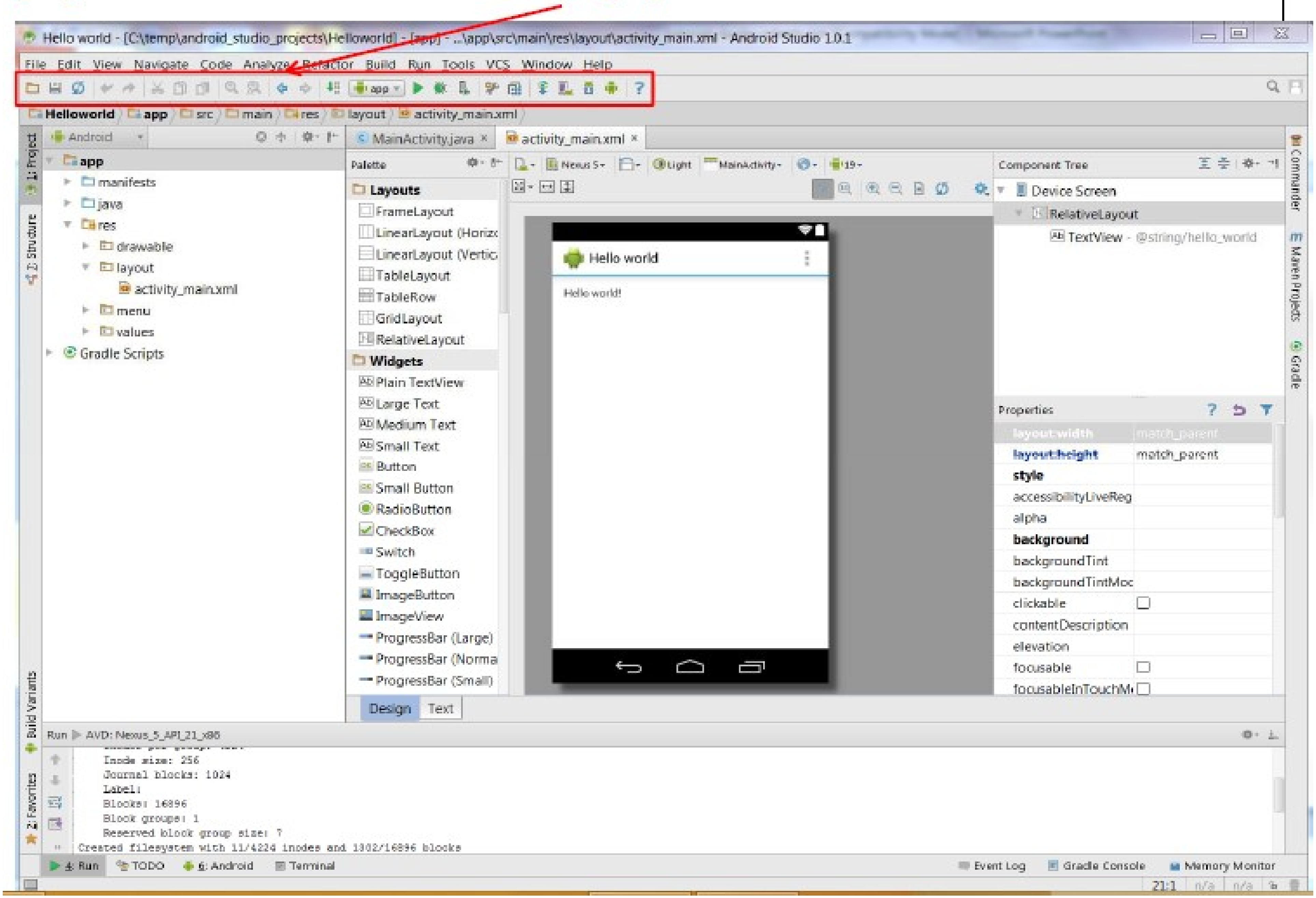
Typical Windows Menu Bar (File, edit, etc)



layout:width match_parent
layout:height match_parent
style
accessibilityLiveReg
alpha
background
backgroundTint
backgroundTintMoc
clickable □
contentDescription
elevation
focusable □
focusableInTouchM □

```
    Inode size: 256
    Journal blocks: 1024
    Label:
    Blocks: 16896
    Block groups: 1
    Reserved block group size: 7
Created filesystem with 11/4224 inodes and 1302/16896 blocks
```

Tool Bar: Shortcuts to Frequently used Android-specific Functions (E.g. One-click access to SDK manager)

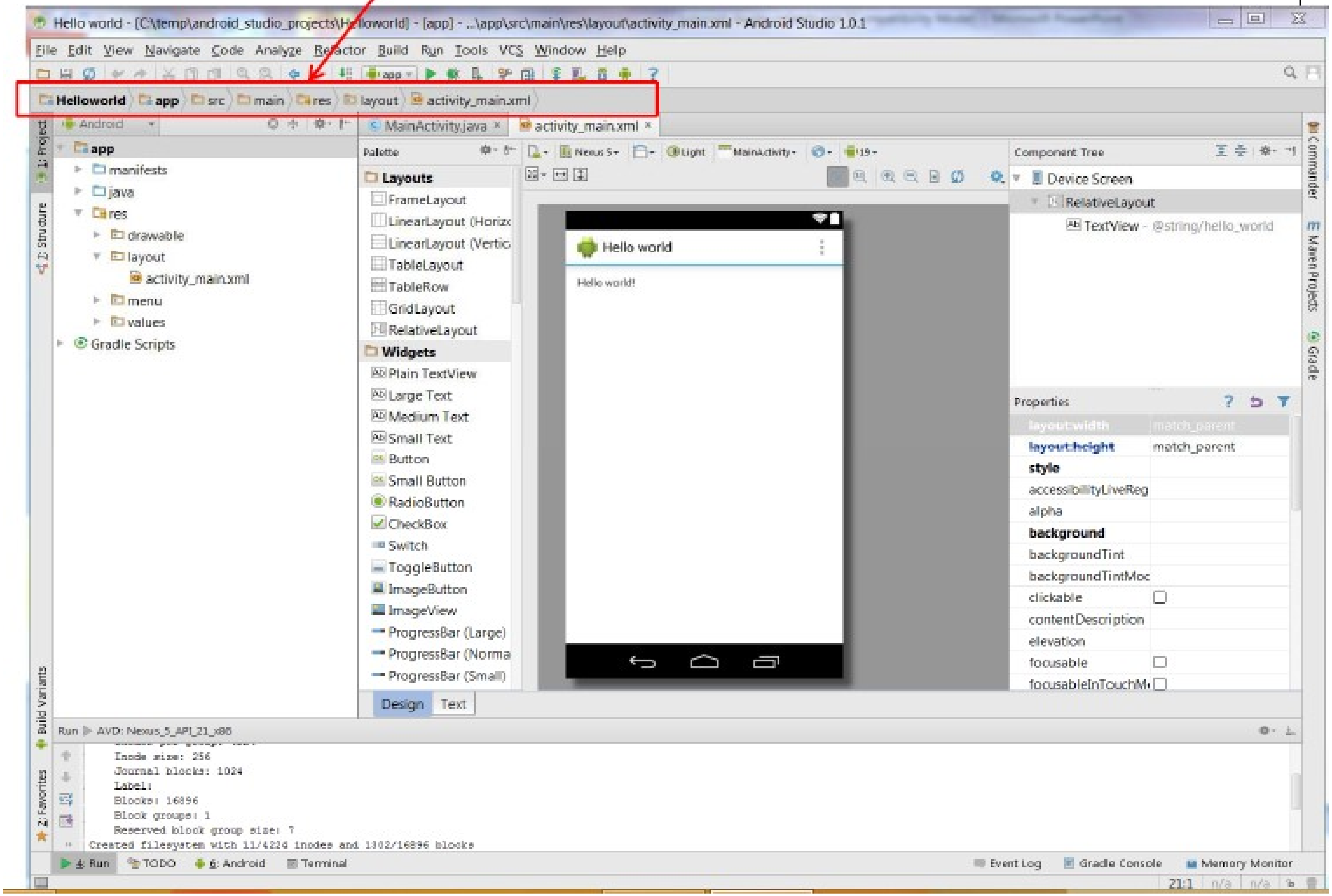


layout:width match_parent
layout:height match_parent
style
accessibilityLiveReg
alpha
background
backgroundTint
backgroundTintMo
clickable □
contentDescription
elevation
focusable □
focusableInTouchM □

Run ▶ AVD: Nexus_5_API_21_x86

```
    Inode size: 256
    Journal blocks: 1024
    Label:
    Blocks: 16896
    Block groups: 1
    Reserved block group size: 7
Created filesystem with 11/4224 inodes and 1302/16896 blocks
```

Path to Current File in IDE Window (Clickable)



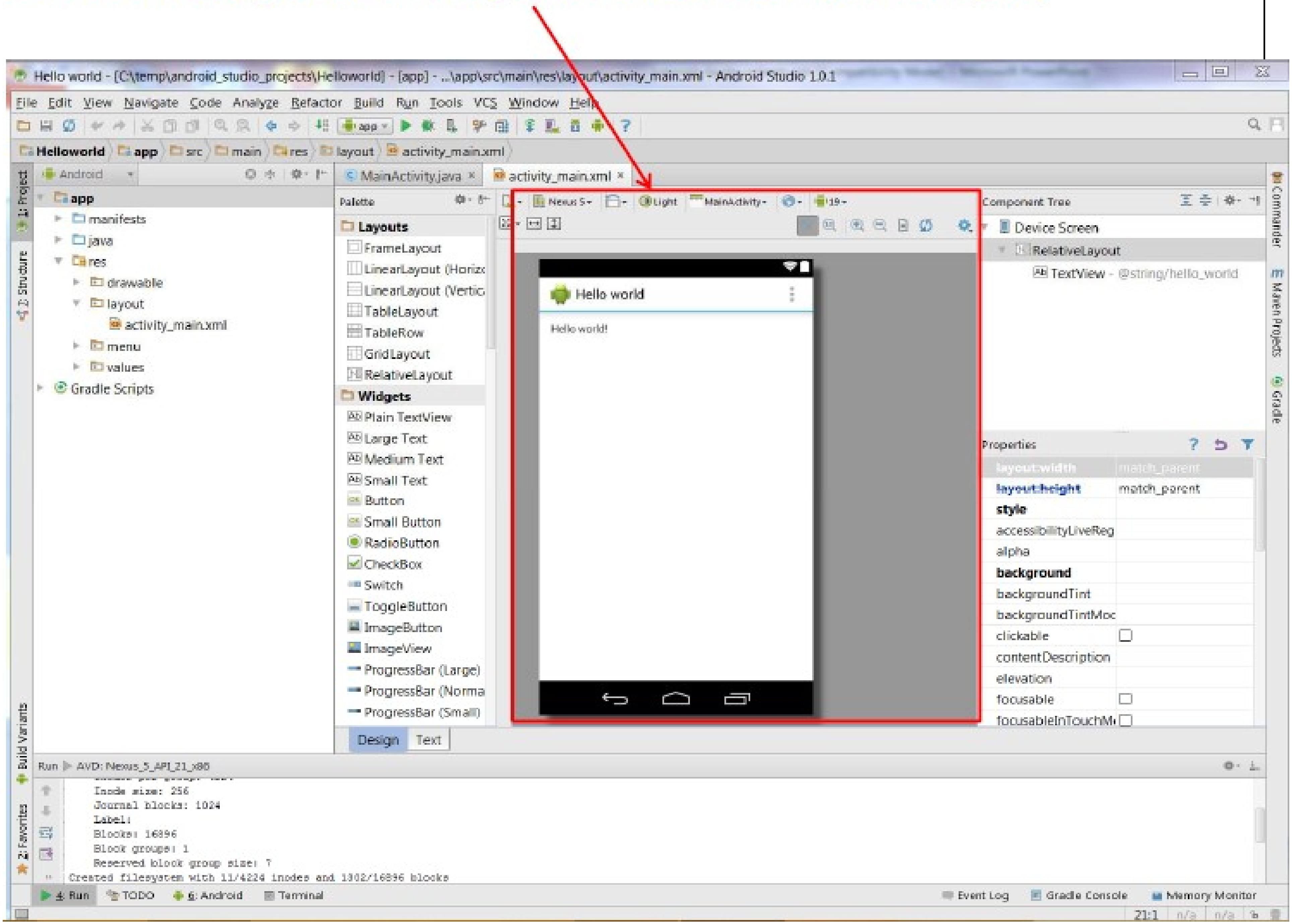
Hello world - [C:\temp\android_studio_projects\Helloworld] - [app] - ...\app\src\main\res\layout\activity_main.xml - Android Studio 1.0.1

Helloworld 〉 app 〉 src 〉 main 〉 res 〉 layout 〉 activity_main.xml

layout:width	match_parent
layout:height	match_parent
style	
accessibilityLiveReg	
alpha	
background	
backgroundTint	
backgroundTintMoc	
clickable	☐
contentDescription	
elevation	
focusable	☐
focusableInTouchM	☐

```
    Inode size: 256
    Journal blocks: 1024
    Label:
    Blocks: 16896
    Block groups: 1
    Reserved block group size: 7
Created filesystem with 11/4224 inodes and 1302/16896 blocks
```

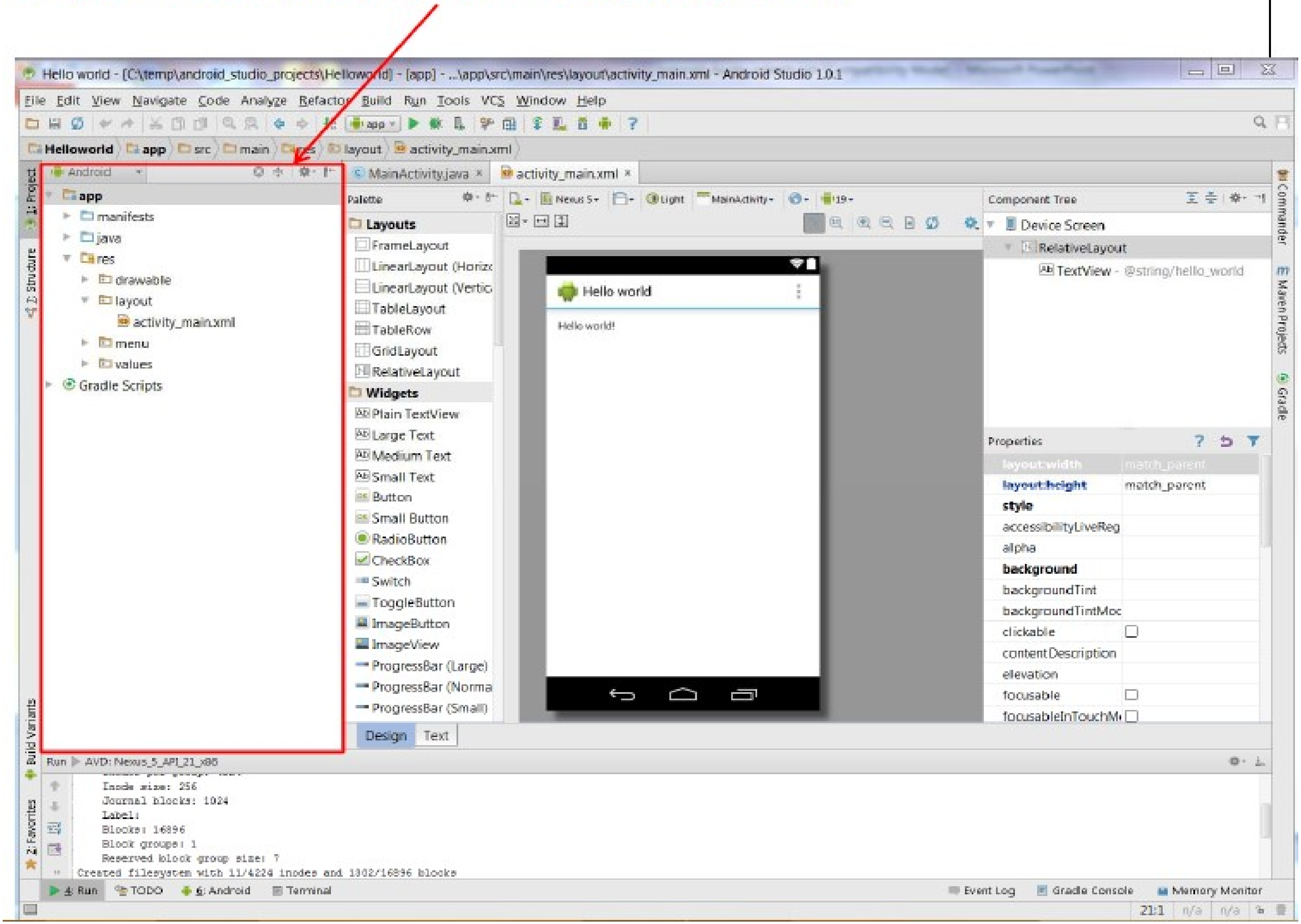
Editor Window (Allows editting of current file we are working on)



layout:width	match_parent
layout:height	match_parent
style	
accessibilityLiveReg	
alpha	
background	
backgroundTint	
backgroundTintMoc	
clickable	□
contentDescription	
elevation	
focusable	□
focusableInTouchM	□

```
    Inode size: 256
    Journal blocks: 1024
    Label:
    Blocks: 16896
    Block groups: 1
    Reserved block group size: 7
Created filesystem with 11/4224 inodes and 1302/16896 blocks
```

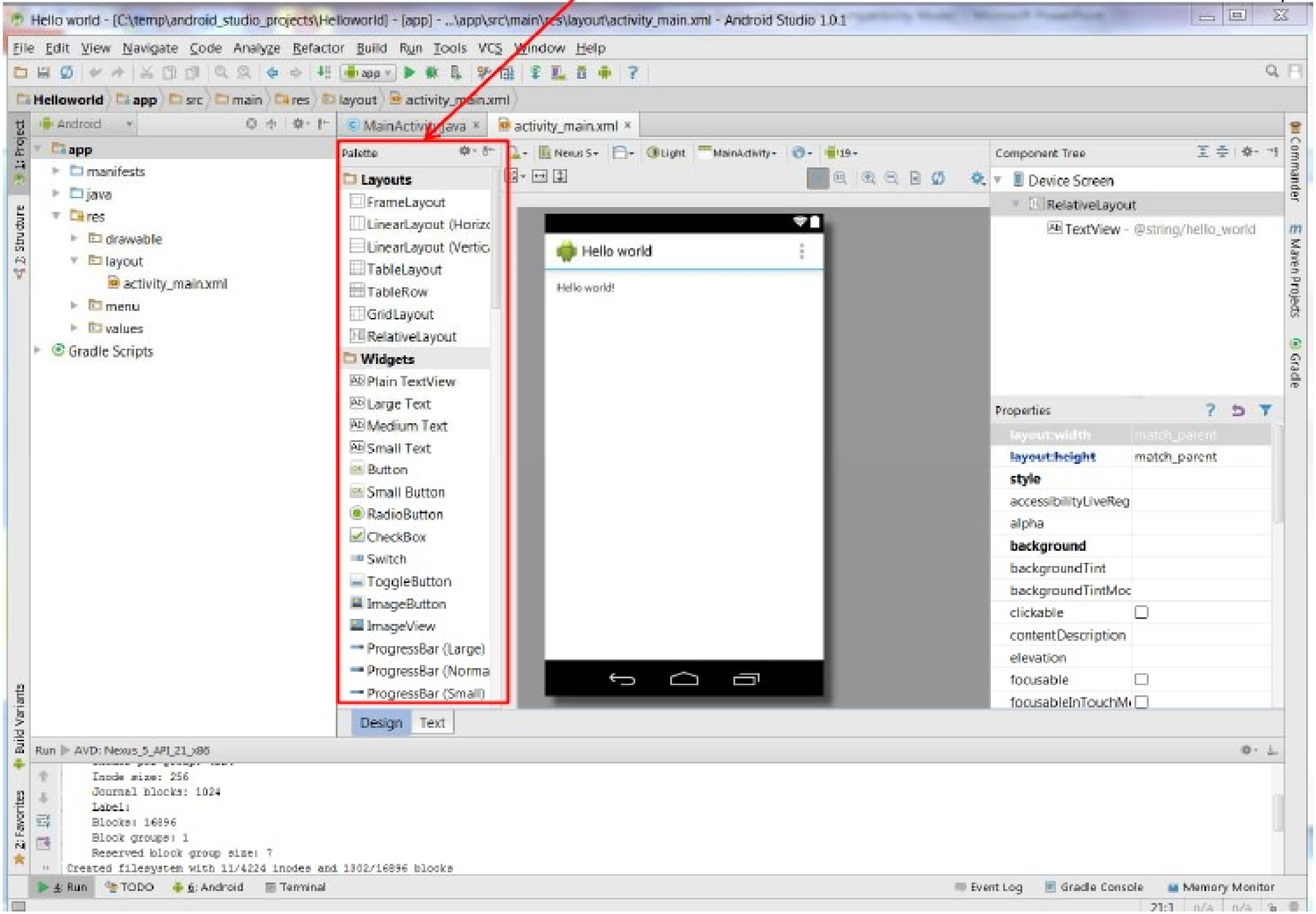
Project Window (Shows project files, packages, etc)



layout:width	match_parent
layout:height	match_parent
style	
accessibilityLiveReg	
alpha	
background	
backgroundTint	
backgroundTintMoc	
clickable	□
contentDescription	
elevation	
focusable	□
focusableInTouchM	□

```
    Inode size: 256
    Journal blocks: 1024
    Label:
    Blocks: 16896
    Block groups: 1
    Reserved block group size: 7
Created filesystem with 11/4224 inodes and 1302/16896 blocks
```

Palette of Drag-and-Drop Elements for Designing Interface (Layout, widgets, etc)



Hello world - [C:\temp\android_studio_projects\Helloworld] - [app] - ...\app\src\main\res\layout\activity_main.xml - Android Studio 1.0.1

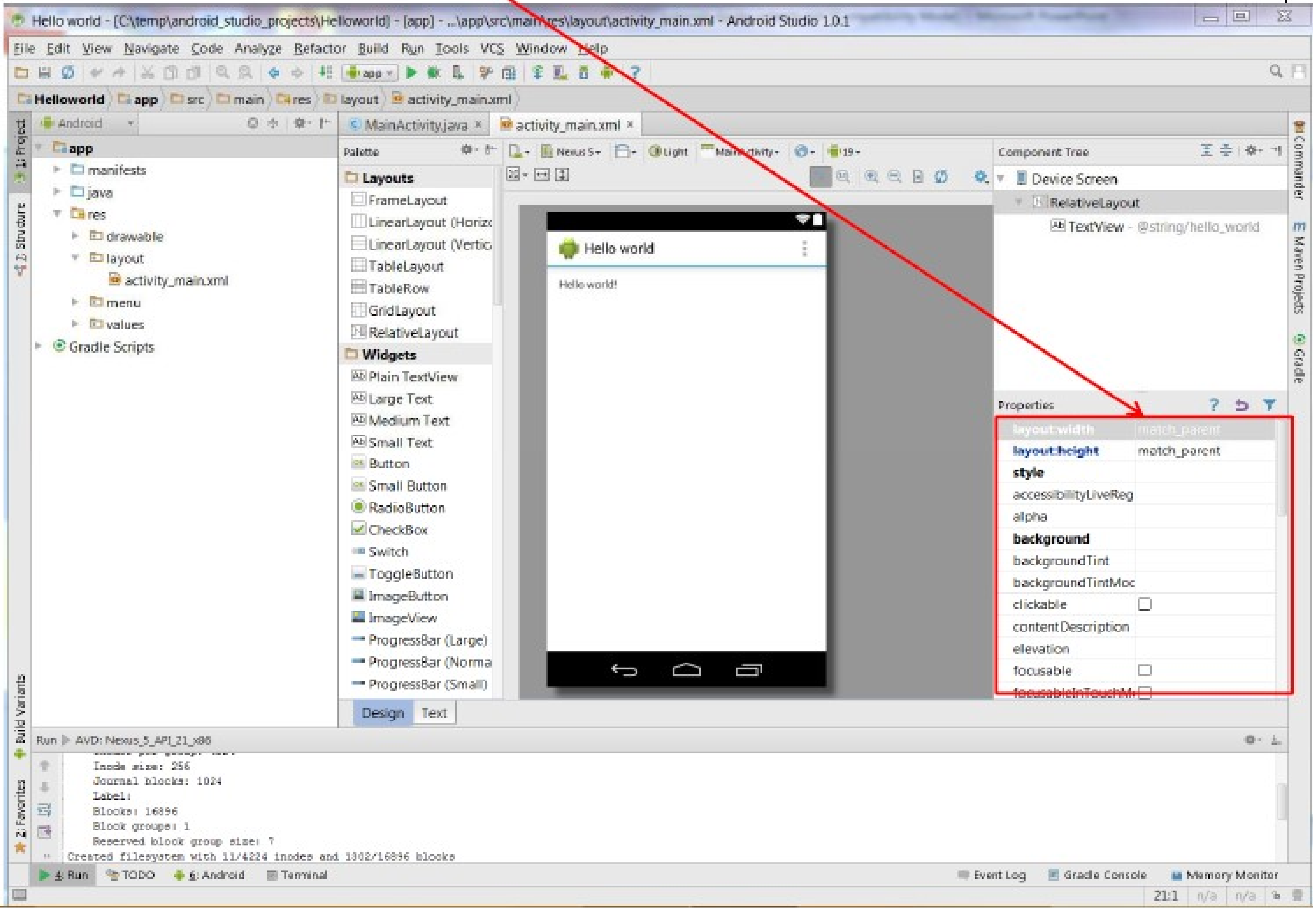
Properties

layout:width	match_parent
layout:height	match_parent
style	
accessibilityLiveReg	
alpha	
background	
backgroundTint	
backgroundTintMod	
clickable	□
contentDescription	
elevation	
focusable	□
focusableInTouchM	□

Run ▶ AVD: Nexus_5_API_21_x86

```
    Inode size: 256
    Journal blocks: 1024
    Label:
    Blocks: 16896
    Block groups: 1
    Reserved block group size: 7
Created filesystem with 11/4224 inodes and 1302/16896 blocks
```

Parameters of Drag-and-Drop Elements for Designing Interface (e.g. colors, dimensions of widgets, etc)



layout:width	match_parent
layout:height	match_parent
style	
accessibilityLiveReg	
alpha	
background	
backgroundTint	
backgroundTintMoc	
clickable	☐
contentDescription	
elevation	
focusable	☐
focusableInTouchM	☐

```
    Inode size: 256
    Journal blocks: 1024
    Label:
    Blocks: 16896
    Block groups: 1
    Reserved block group size: 7
Created filesystem with 11/4224 inodes and 1302/16896 blocks
```

ADT (Andriod Development Tools)

Android Development Tools

Android Studio incorporates all of these tools:

The Android Virtual Device Manager and Emulator—The AVD Manager is used to create and manage AVDs, virtual hardware that hosts an Emulator running a particular build of Android. Each AVD can specify a particular screen size and resolution, memory and storage capacities, and available hardware capabilities (such as touch screens and GPS). The Android Emulator is an implementation of the Android Run Time designed to run within an AVD on your host development computer.

The Android SDK Manager—Used to download SDK packages including Android platform SDKs, support libraries, and the Google Play Services SDK.

Android Profiler—Visualize the behavior and performance of your app. The Android Profiler can track memory and CPU use in real time, as well as analyze network traffic.

Lint—A static analysis tool that analyzes your application and its resources to suggest improvements and optimizations.

Android Development Tools

Gradle—An advanced build system and toolkit that manages the compilation, packaging, and deployment of your applications.

Vector Asset Studio—Generates bitmap files for each screen density to

support older versions of Android that don't support the Android vector drawable format.

APK Analyzer—Provides insight into the composition of your built APK

files.

- The following additional tools are also available:

Android Debug Bridge (ADB)—A client-server application that provides a link between your host computer and virtual and physical Android devices. It lets you copy files, install compiled application packages (.apk), and run shell commands.

Logcat—A utility used to view and filter the output of the Android logging system.

Android Asset Packaging Tool (AAPT)—Constructs the distributable Android package files (.apk).

SQLite3—A database tool that you can use to access the SQLite database files created and used by Android.

Hprof-conv—A tool that converts HPROF profiling output files into a standard format to view in your preferred profiling tool.

Android Development Tools

Dx—Converts Java .class bytecode into Android .dex bytecode**.**

Draw9patch—A handy utility to simplify the creation of **NinePatch** graphics using a WYSIWYG editor.

Monkey and Monkey Runner—

- ✓ **Monkey** runs within the Android Run Time, generating pseudorandom user and system events.
- ✓ **Monkey Runner** provides an API for writing programs to control the VM from outside your application.

ProGuard—A tool to shrink and obfuscate your code by replacing class, variable, and method names with semantically meaningless alternatives.

- This is useful to make your code more difficult to reverse engineer.