ADVANCED VISUAL PROGRAMMING (DROID)

VISUAL PROGRAMMING LANGUAGE

- A **VISUAL PROGRAMMING LANGUAGE** (**VPL**) IS ANY PROGRAMMING LANGUAGE THAT LETS USERS CREATE PROGRAMS BY MANIPULATING PROGRAM ELEMENTS GRAPHICALLY RATHER THAN BY SPECIFYING THEM TEXTUALLY.
- A VPL ALLOWS PROGRAMMING WITH VISUAL EXPRESSIONS, SPATIAL ARRANGEMENTS OF TEXT AND GRAPHIC SYMBOLS, USED EITHER AS ELEMENTS OF SYNTAX OR SECONDARY NOTATION. (LINK)

- VPLS MAY BE FURTHER CLASSIFIED, ACCORDING TO THE TYPE AND EXTENT OF VISUAL EXPRESSION USED, INTO
    1. ICON-BASED LANGUAGES,
    2. FORM-BASED LANGUAGES, AND
    3. DIAGRAM LANGUAGES.
- VISUAL PROGRAMMING ENVIRONMENTS PROVIDE GRAPHICAL OR ICONIC ELEMENTS WHICH CAN BE MANIPULATED BY USERS IN AN INTERACTIVE WAY ACCORDING TO SOME SPECIFIC SPATIAL GRAMMAR FOR PROGRAM CONSTRUCTION.

- THE "VISUAL LANGUAGES" (VISUAL BASIC, VISUAL C#, VISUAL J#, ETC.) OF THE MICROSOFT VISUAL STUDIO IDE ARE NOT VISUAL PROGRAMMING LANGUAGES.
- ALL OF THEM ARE TEXTUAL AND NOT GRAPHICAL.
- THE MS VISUAL STUDIO IS A VISUAL PROGRAMMING ENVIRONMENT, HOWEVER.
- SO IS ANDROID STUDIO

EXAMPLES OF VPL(S)

- MICROSOFT VPL
- RAPTOR
- BLENDER

ANDROID STUDIO

- IS THE OFFICIAL INTEGRATED DEVELOPMENT ENVIRONMENT (IDE) FOR ANDROID APP DEVELOPMENT, BASED ON INTELLIJ IDEA .
- ON TOP OF INTELLIJ'S POWERFUL CODE EDITOR AND DEVELOPER TOOLS, ANDROID STUDIO OFFERS EVEN MORE FEATURES THAT ENHANCE PRODUCTIVITY WHEN BUILDING ANDROID APPS, SUCH AS:
  - A FLEXIBLE GRADLE-BASED BUILD SYSTEM
  - A FAST AND FEATURE-RICH EMULATOR

- A UNIFIED ENVIRONMENT WHERE YOU CAN DEVELOP FOR ALL ANDROID DEVICES
- INSTANT RUN TO PUSH CHANGES TO YOUR RUNNING APP WITHOUT BUILDING A NEW APK
- CODE TEMPLATES AND GITHUB INTEGRATION TO HELP YOU BUILD COMMON APP FEATURES AND IMPORT SAMPLE CODE
- EXTENSIVE TESTING TOOLS AND FRAMEWORKS
- LINT TOOLS TO CATCH PERFORMANCE, USABILITY, VERSION COMPATIBILITY, AND OTHER PROBLEMS
- C++ AND NDK SUPPORT
- BUILT-IN SUPPORT FOR GOOGLE CLOUD PLATFORM, MAKING IT EASY TO INTEGRATE GOOGLE CLOUD MESSAGING AND APP ENGINE

VISUAL ENVIRONMENT

- WINDOWS:
  1. MICROSOFT® WINDOWS® 7/8/10 (32- OR 64-BIT)
  2. 3 GB RAM MINIMUM, 8 GB RAM RECOMMENDED; PLUS 1 GB FOR THE ANDROID EMULATOR
  3. 2 GB OF AVAILABLE DISK SPACE MINIMUM,
     4 GB RECOMMENDED (500 MB FOR IDE + 1.5 GB FOR ANDROID SDK AND EMULATOR SYSTEM IMAGE)
  4. 1280 X 800 MINIMUM SCREEN RESOLUTION
  5. FOR ACCELERATED EMULATOR: 64-BIT OPERATING SYSTEM AND INTEL® PROCESSOR WITH SUPPORT FOR INTEL® VT-X, INTEL® EM64T (INTEL® 64), AND EXECUTE DISABLE (XD) BIT FUNCTIONALITY

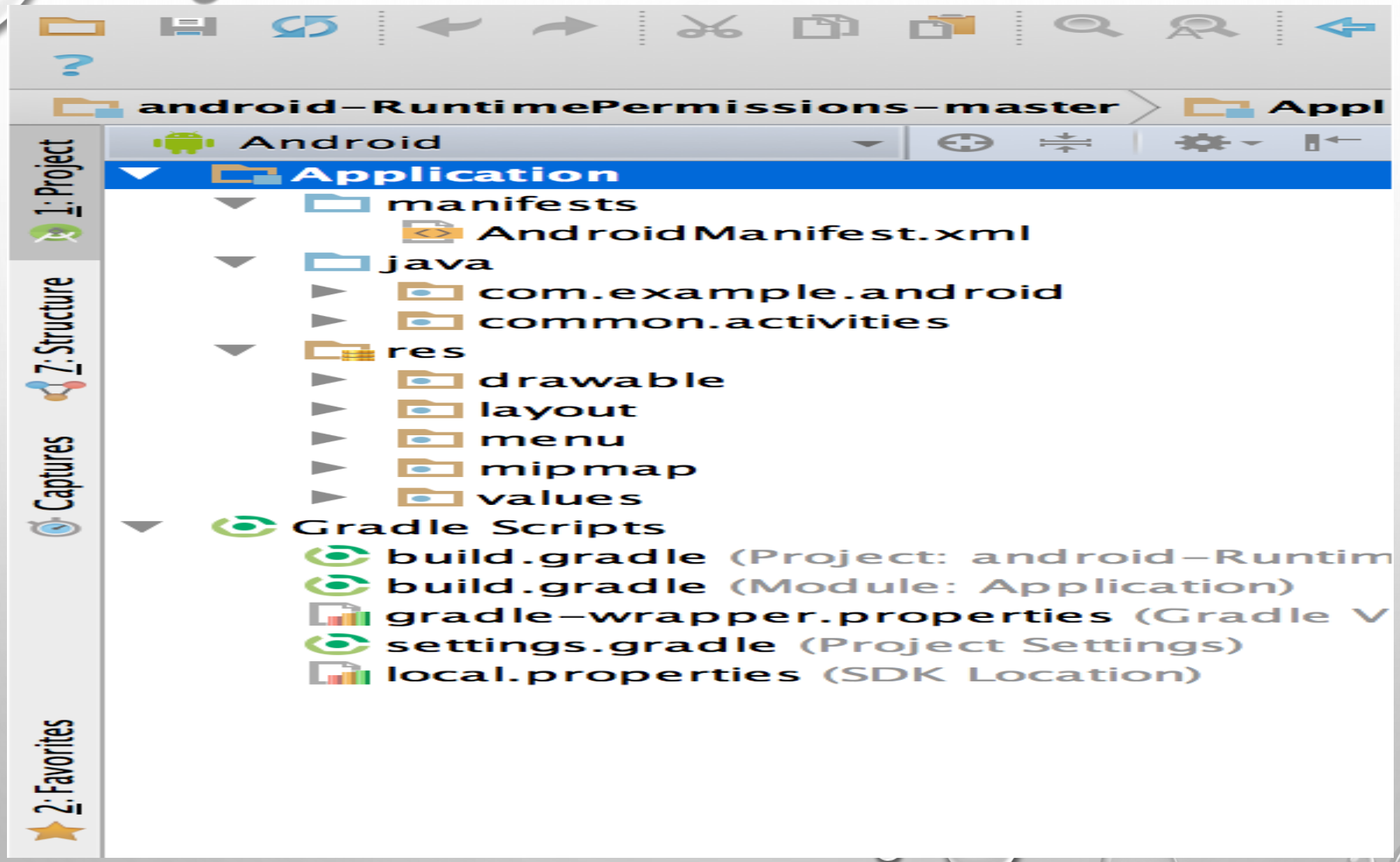
- MAC
    1. MAC® OS X® 10.10 (YOSEMITE) OR HIGHER, UP TO 10.12 (MACOS SIERRA)
    2. 3 GB RAM MINIMUM, 8 GB RAM RECOMMENDED; PLUS 1 GB FOR THE ANDROID EMULATOR
    3. 2 GB OF AVAILABLE DISK SPACE MINIMUM,
       4 GB RECOMMENDED (500 MB FOR IDE + 1.5 GB FOR ANDROID SDK AND EMULATOR SYSTEM IMAGE)
    4. 1280 X 800 MINIMUM SCREEN RESOLUTION

- LINUX
  1. GNOME OR KDE DESKTOP*TESTED ON UBUNTU® 12.04, PRECISE PANGOLIN (64-BIT DISTRIBUTION CAPABLE OF RUNNING 32-BIT APPLICATIONS)*
  2. 64-BIT DISTRIBUTION CAPABLE OF RUNNING 32-BIT APPLICATIONS
  3. GNU C LIBRARY (GLIBC) 2.19 OR LATER
  4. 3 GB RAM MINIMUM, 8 GB RAM RECOMMENDED; PLUS 1 GB FOR THE ANDROID EMULATOR
  5. 2 GB OF AVAILABLE DISK SPACE MINIMUM,
     4 GB RECOMMENDED (500 MB FOR IDE + 1.5 GB FOR ANDROID SDK AND EMULATOR SYSTEM IMAGE)
  6. 1280 X 800 MINIMUM SCREEN RESOLUTION
  7. FOR ACCELERATED EMULATOR: INTEL® PROCESSOR WITH SUPPORT FOR INTEL® VT-X, INTEL® EM64T (INTEL® 64), AND EXECUTE DISABLE (XD) BIT FUNCTIONALITY, OR AMD PROCESSOR WITH SUPPORT FOR AMD VIRTUALIZATION™ (AMD-V™)

PROJECT STRUCTURE

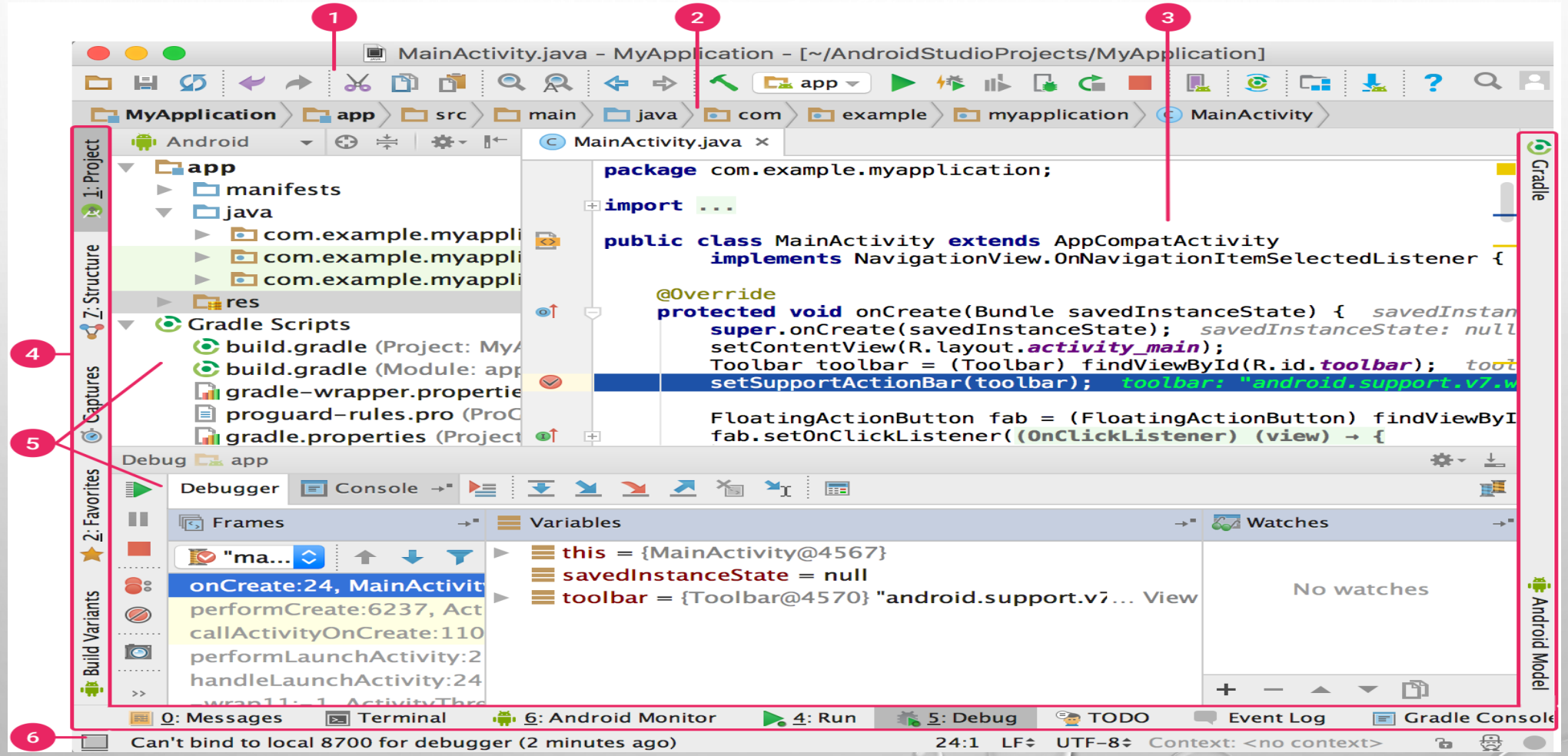
- EACH PROJECT IN ANDROID STUDIO CONTAINS ONE OR MORE MODULES WITH SOURCE CODE FILES AND RESOURCE FILES.
- TYPES OF MODULES INCLUDE:
  - ANDROID APP MODULES
  - LIBRARY MODULES
  - GOOGLE APP ENGINE MODULES

- BY DEFAULT, ANDROID STUDIO DISPLAYS YOUR PROJECT FILES IN THE ANDROID PROJECT VIEW, AS SHOWN IN THE NEXT SLIDE.
- THIS VIEW IS ORGANIZED BY MODULES TO PROVIDE QUICK ACCESS TO YOUR PROJECT'S KEY SOURCE FILES.


android-RuntimePermissions-master
Appl
Android
1: Project
7: Structure
Captures
2: Favorites
Application
manifests
AndroidManifest.xml
java
com.example.android
common.activities
res
drawable
layout
menu
mipmap
values
Gradle Scripts
build.gradle (Project: android-Runtim
build.gradle (Module: Application)
gradle-wrapper.properties (Gradle V
settings.gradle (Project Settings)
local.properties (SDK Location)

- ALL THE BUILD FILES ARE VISIBLE AT THE TOP LEVEL UNDER **GRADLE SCRIPTS** AND EACH APP MODULE CONTAINS THE FOLLOWING FOLDERS:
    1. **MANIFESTS**: CONTAINS THE ANDROIDMANIFEST.XML FILE.
    2. **JAVA**: CONTAINS THE JAVA SOURCE CODE FILES, INCLUDING JUNIT TEST CODE.
    3. **RES**: CONTAINS ALL NON-CODE RESOURCES, SUCH AS XML LAYOUTS, UI STRINGS, AND BITMAP IMAGES.
- THE ANDROID PROJECT STRUCTURE ON DISK DIFFERS FROM THIS FLATTENED REPRESENTATION.
- TO SEE THE ACTUAL FILE STRUCTURE OF THE PROJECT, SELECT **PROJECT** FROM THE **PROJECT** DROPDOWN
- YOU CAN ALSO CUSTOMIZE THE VIEW OF THE PROJECT FILES TO FOCUS ON SPECIFIC ASPECTS OF YOUR APP DEVELOPMENT.

THE USER INTERFACE



1. THE **TOOLBAR** LETS YOU CARRY OUT A WIDE RANGE OF ACTIONS, INCLUDING RUNNING YOUR APP AND LAUNCHING ANDROID TOOLS.
2. THE **NAVIGATION BAR** HELPS YOU NAVIGATE THROUGH YOUR PROJECT AND OPEN FILES FOR EDITING. IT PROVIDES A MORE COMPACT VIEW OF THE STRUCTURE VISIBLE IN THE **PROJECT** WINDOW.
3. THE **EDITOR WINDOW** IS WHERE YOU CREATE AND MODIFY CODE. DEPENDING ON THE CURRENT FILE TYPE, THE EDITOR CAN CHANGE. FOR EXAMPLE, WHEN VIEWING A LAYOUT FILE, THE EDITOR DISPLAYS THE LAYOUT EDITOR.
4. THE **TOOL WINDOW BAR** RUNS AROUND THE OUTSIDE OF THE IDE WINDOW AND CONTAINS THE BUTTONS THAT ALLOW YOU TO EXPAND OR COLLAPSE INDIVIDUAL TOOL WINDOWS.
5. THE **TOOL WINDOWS** GIVE YOU ACCESS TO SPECIFIC TASKS LIKE PROJECT MANAGEMENT, SEARCH, VERSION CONTROL, AND MORE. YOU CAN EXPAND THEM AND COLLAPSE THEM.
6. 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.
- 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.
- THIS CAN BE VERY USEFUL IF, FOR EXAMPLE, YOU ARE TRYING TO LOCATE A PARTICULAR IDE ACTION THAT YOU HAVE FORGOTTEN HOW TO TRIGGER

TOOL WINDOWS

- INSTEAD OF USING PRESET PERSPECTIVES, ANDROID STUDIO FOLLOWS YOUR CONTEXT AND AUTOMATICALLY BRINGS UP RELEVANT TOOL WINDOWS AS YOU WORK.
- BY DEFAULT, THE MOST COMMONLY USED TOOL WINDOWS ARE PINNED TO THE TOOL WINDOW BAR AT THE EDGES OF THE APPLICATION WINDOW.

- TO EXPAND OR COLLAPSE A TOOL WINDOW, CLICK THE TOOL'S NAME IN THE TOOL WINDOW BAR.
- YOU CAN ALSO DRAG, PIN, UNPIN, ATTACH, AND DETACH TOOL WINDOWS.
- TO RETURN TO THE CURRENT DEFAULT TOOL WINDOW LAYOUT, CLICK WINDOW > RESTORE DEFAULT LAYOUT OR CUSTOMIZE YOUR DEFAULT LAYOUT BY CLICKING WINDOW > STORE CURRENT LAYOUT AS DEFAULT.
- TO SHOW OR HIDE THE ENTIRE TOOL WINDOW BAR, CLICK THE WINDOW ICON IN THE BOTTOM LEFT-HAND CORNER OF THE ANDROID STUDIO WINDOW.
- TO LOCATE A SPECIFIC TOOL WINDOW, HOVER OVER THE WINDOW ICON AND SELECT THE TOOL WINDOW FROM THE MENU.

- YOU CAN ALSO USE KEYBOARD SHORTCUTS TO OPEN TOOL WINDOWS

Tool Window	Windows and Linux	Mac
Project	**Alt+1**	**Command+1**
Version Control	**Alt+9**	**Command+9**
Run	**Shift+F10**	**Control+R**
Debug	**Shift+F9**	**Control+D**
Android Monitor	**Alt+6**	**Command+6**
Return to Editor	**Esc**	**Esc**
Hide All Tool Windows	**Control+Shift+F12**	**Command+Shift+F12**

NAVIGATION

- HERE ARE SOME TIPS TO HELP YOU MOVE AROUND ANDROID STUDIO.
  - SWITCH BETWEEN YOUR RECENTLY ACCESSED FILES USING THE *RECENT FILES* ACTION. PRESS **CONTROL+E** (**COMMAND+E** ON A MAC) TO BRING UP THE RECENT FILES ACTION. BY DEFAULT, THE LAST ACCESSED FILE IS SELECTED. YOU CAN ALSO ACCESS ANY TOOL WINDOW THROUGH THE LEFT COLUMN IN THIS ACTION.
  - VIEW THE STRUCTURE OF THE CURRENT FILE USING THE *FILE STRUCTURE* ACTION. BRING UP THE FILE STRUCTURE ACTION BY PRESSING **CONTROL+F12** (**COMMAND+F12** ON A MAC). USING THIS ACTION, YOU CAN QUICKLY NAVIGATE TO ANY PART OF YOUR CURRENT FILE.

- SEARCH FOR AND NAVIGATE TO A SPECIFIC CLASS IN YOUR PROJECT USING THE *NAVIGATE TO CLASS* ACTION. BRING UP THE ACTION BY PRESSING **CONTROL+N**(**COMMAND+O** ON A MAC). NAVIGATE TO CLASS SUPPORTS SOPHISTICATED EXPRESSIONS, INCLUDING CAMEL HUMPS, PATHS, LINE NAVIGATE TO, MIDDLE NAME MATCHING, AND MANY MORE. IF YOU CALL IT TWICE IN A ROW, IT SHOWS YOU THE RESULTS OUT OF THE PROJECT CLASSES.
- NAVIGATE TO A FILE OR FOLDER USING THE *NAVIGATE TO FILE* ACTION. BRING UP THE NAVIGATE TO FILE ACTION BY PRESSING **CONTROL+SHIFT+N** (**COMMAND+SHIFT+O** ON A MAC). TO SEARCH FOR FOLDERS RATHER THAN FILES, ADD A / AT THE END OF YOUR EXPRESSION.
- NAVIGATE TO A METHOD OR FIELD BY NAME USING THE *NAVIGATE TO SYMBOL* ACTION. BRING UP THE NAVIGATE TO SYMBOL ACTION BY PRESSING **CONTROL+SHIFT+ALT+N**(**COMMAND+SHIFT+ALT+O** ON A MAC).
- FIND ALL THE PIECES OF CODE REFERENCING THE CLASS, METHOD, FIELD, PARAMETER, OR STATEMENT AT THE CURRENT CURSOR POSITION BY PRESSING **ALT+F7**.

STYLE AND FORMATTING

- AS YOU EDIT, ANDROID STUDIO AUTOMATICALLY APPLIES FORMATTING AND STYLES AS SPECIFIED IN YOUR CODE STYLE SETTINGS.
- YOU CAN CUSTOMIZE THE CODE STYLE SETTINGS BY PROGRAMMING LANGUAGE, INCLUDING SPECIFYING CONVENTIONS FOR TABS AND INDENTS, SPACES, WRAPPING AND BRACES, AND BLANK LINES.
- TO CUSTOMIZE YOUR CODE STYLE SETTINGS, CLICK **FILE > SETTINGS > EDITOR > CODE STYLE** (**ANDROID STUDIO > PREFERENCES > EDITOR > CODE STYLE** ON A MAC.)

- ALTHOUGH THE IDE AUTOMATICALLY APPLIES FORMATTING AS YOU WORK, YOU CAN ALSO EXPLICITLY CALL THE *REFORMAT CODE* ACTION BY PRESSING **CONTROL+ALT+L**(**OPT+COMMAND+L** ON A MAC), OR AUTO-INDENT ALL LINES BY PRESSING **CONTROL+ALT+I** (**ALT+OPTION+I** ON A MAC).

GRADLE BUILD SYSTEM

- ANDROID STUDIO USES GRADLE AS THE FOUNDATION OF THE BUILD SYSTEM, WITH MORE ANDROID-SPECIFIC CAPABILITIES PROVIDED BY THE ANDROID PLUGIN FOR GRADLE.
- THIS BUILD SYSTEM RUNS AS AN INTEGRATED TOOL FROM THE ANDROID STUDIO MENU, AND INDEPENDENTLY FROM THE COMMAND LINE.
- YOU CAN USE THE FEATURES OF THE BUILD SYSTEM TO DO THE FOLLOWING:
  - CUSTOMIZE, CONFIGURE, AND EXTEND THE BUILD PROCESS.
  - CREATE MULTIPLE APKS FOR YOUR APP, WITH DIFFERENT FEATURES USING THE SAME PROJECT AND MODULES.
  - REUSE CODE AND RESOURCES ACROSS SOURCESETS.

- BY EMPLOYING THE FLEXIBILITY OF GRADLE, YOU CAN ACHIEVE ALL OF THIS WITHOUT MODIFYING YOUR APP'S CORE SOURCE FILES.
- ANDROID STUDIO BUILD FILES ARE NAMED BUILD.GRADLE.
- THEY ARE PLAIN TEXT FILES THAT USE GROOVY SYNTAX TO CONFIGURE THE BUILD WITH ELEMENTS PROVIDED BY THE ANDROID PLUGIN FOR GRADLE.
- EACH PROJECT HAS ONE TOP-LEVEL BUILD FILE FOR THE ENTIRE PROJECT AND SEPARATE MODULE-LEVEL BUILD FILES FOR EACH MODULE.
- WHEN YOU IMPORT AN EXISTING PROJECT, ANDROID STUDIO AUTOMATICALLY GENERATES THE NECESSARY BUILD FILES.
- TO CONFIGURE BUILD TYPE, READ THIS LINK

MANAGING DEPENDENCIES

- DEPENDENCIES FOR YOUR PROJECT ARE SPECIFIED BY NAME IN THE BUILD.GRADLE FILE.
- GRADLE TAKES CARE OF FINDING YOUR DEPENDENCIES AND MAKING THEM AVAILABLE IN YOUR BUILD.
- YOU CAN DECLARE MODULE DEPENDENCIES, REMOTE BINARY DEPENDENCIES, AND LOCAL BINARY DEPENDENCIES IN YOUR BUILD.GRADLE FILE.
- ANDROID STUDIO CONFIGURES PROJECTS TO USE THE MAVEN CENTRAL REPOSITORY BY DEFAULT.
- (THIS CONFIGURATION IS INCLUDED IN THE TOP-LEVEL BUILD FILE FOR THE PROJECT.) FOR MORE INFORMATION ABOUT CONFIGURING DEPENDENCIES, READ CONFIGURE BUILD VARIANTS

DEBUG AND PROFILE TOOLS

- ANDROID STUDIO ASSISTS YOU IN DEBUGGING AND IMPROVING THE PERFORMANCE OF YOUR CODE, INCLUDING INLINE DEBUGGING AND PERFORMANCE ANALYSIS TOOLS

INLINE DEBUGGING

- USE INLINE DEBUGGING TO ENHANCE YOUR CODE WALK-THROUGHS IN THE DEBUGGER VIEW WITH INLINE VERIFICATION OF REFERENCES, EXPRESSIONS, AND VARIABLE VALUES.
- INLINE DEBUG INFORMATION INCLUDES:
  - INLINE VARIABLE VALUES
  - REFERRING OBJECTS THAT REFERENCE A SELECTED OBJECT
  - METHOD RETURN VALUES
  - LAMBDA AND OPERATOR EXPRESSIONS
  - TOOLTIP VALUES
- TO ENABLE INLINE DEBUGGING, IN THE DEBUG WINDOW, CLICK SETTINGS AND SELECT THE CHECKBOX FOR SHOW VALUES INLINE
- MORE ON DEBUGGING SEE LINKS

DATA FILE ACCESS

- THE ANDROID SDK TOOLS, SUCH AS SYSTRACE, LOGCAT, AND TRACEVIEW, GENERATE PERFORMANCE AND DEBUGGING DATA FOR DETAILED APP ANALYSIS.
- TO VIEW THE AVAILABLE GENERATED DATA FILES, OPEN THE CAPTURES TOOL WINDOW.
- IN THE LIST OF THE GENERATED FILES, DOUBLE-CLICK A FILE TO VIEW THE DATA.
- RIGHT-CLICK ANY .HPROF FILES TO CONVERT THEM TO THE STANDARD .HPROF FILE FORMAT

CODE INSPECTIONS

- WHENEVER YOU COMPILE YOUR PROGRAM, ANDROID STUDIO AUTOMATICALLY RUNS CONFIGURED LINT AND OTHER IDE INSPECTIONS TO HELP YOU EASILY IDENTIFY AND CORRECT PROBLEMS WITH THE STRUCTURAL QUALITY OF YOUR CODE.
- THE LINT TOOL CHECKS YOUR ANDROID PROJECT SOURCE FILES FOR POTENTIAL BUGS AND OPTIMIZATION IMPROVEMENTS FOR CORRECTNESS, SECURITY, PERFORMANCE, USABILITY, ACCESSIBILITY, AND INTERNATIONALIZATION.