REVIEWER

Simplifying Collecting User Input

- The onscreen keyboard is called a **soft keyboard**
- Input can be in the form of tapping or gestures (using two fingers to pan, rotate, or zoom)
- Primary design challenge is to simplify user experiences
- Use legible fonts, simplify input, and optimize each device's capabilities to maximize user experience

Using Android Text Components

- Text entered via a soft or physical keyboard is the most common type of mobile input
- Text Fields are the most common type of mobile input
    - Can be free-form plain text
    - Numbers
    - A person's name, password, email, phone number
    - A date
    - Multiline text
- The component type can constrain the characters available

Adding a Text Field

- Use the Id property in the Properties pane to enter a name that begins with the prefix txt
- Use descriptive names like txtTickets instead of txtText1

Using the String Table

- Uses the file strings.xml
- A **string** is a series of alphanumeric characters that can include spaces
- A **string array** can be used to store a group of related strings
- **Localization** is the use of the String table to change text based on the user's preferred language
    - Android runs on many devices in many regions. To reach the most users, your app should handle text, audio files, numbers, currency, and graphics in ways appropriate to the locales where your app is used.

Adding a String Array

- A **string array** defines a string resource of related items in a central location within strings.xml
    - In the Concert Tickets app, the user can select one of ten performance groups: 6 Cycle Mind, Aegis, Itchyworms, Kamikazee, and etc.
- An **item** defines an individual entry within a string array
    - As you type the String array XML code, the Android Studio editor offers suggestions in a panel that can complete the statement

Setting the Hint Property for the Text Field

- A **hint** is a short description of a field visible as light-colored text (called a watermark) inside a Text Number component or any Text input
    - Often used to provide guidelines about the input expected in a component for the user
- You can change the color of the hint by changing the **textColorHint** attribute
    - May be necessary to ensure adequate contrast between the text and background

Using the Android Spinner Control

- A **Spinner control** is similar to a drop-down list for selecting a single item from a fixed list
- The spinner control displays a list of strings called **items** in a pop-up window
- A **prompt**, which can be used to display instructions at the top of the Spinner control, also is stored in strings.xml and is named prompt
- The Spinner property called **Entries** connects the String Array to the Spinner control for display in the application.

Adding the Button, TextView, and ImageView Controls

- ImageView components should have a content description for visually impaired users
- This description is provided by setting the contentDescription attribute

Coding the EditText Class for the Text Field

- A **variable** is used in programming to contain data that changes during the execution of a program
- **Final** variables can be initialized but cannot be changed
    - Attempting to reassign the value results in a compile error when the application runs

String Data Type

- The String type is a class and not a primitive data type
- A string can be a character, word, or phrase

Declaring the Variables

- Typically declared at the beginning of an Activity
- Variables must be declared before you can use them

GetText() Method

- Read data stored in the EditText control with the **GetText()** method
- Data is read in as a string, by default
- A **Parse** class is used to convert strings into numbers

Formatting Numbers

- Currency format requires a dollar sign, a comma (if needed), a decimal point, and two decimal places
- Java has a class called **DecimalFormat** that provides patterns, such as Php ###,###.## for displaying on the Android screen

GetSelectedItem() Method

- The **GetSelectedItem()** method returns the text label of the currently selected Spinner item

SetText () Method

- The **SetText()** method displays text in a TextView control

ICONS, AND DECISION MAKING CONTROLS (MEDICAL CALCULATOR APP)

Decision-Making Controls

- Developers can code Android applications to make decisions based on the input of users or other conditions that occur.
- **Decision making** is one of the fundamental activities of a computer application.

Icon

- An icon is a graphic that takes up a small portion of screen space and provides a quick, intuitive representation of an app.
- As you design a launcher icon, consider that an icon can establish brand identity.
- A unique image logo and program name can communicate your brand to potential customers.

- In the Medical Calculator app, the scale icon clearly communicates that this icon launches a program about weight.
- A simple image with a clear visual cue like the scale has a memorable impact.
- It also helps users find the app on the Google Play site.
- Google Play suggests icons should be simple and bold in design.

Launcher Icon Sizes

- Google Play also specifies the size and format of all launcher icons for uniformity.
- Launcher icons should be saved in the **.png** file format based on your target device.
- When you publish an app to Google Play, you must provide a **512 × 512-pixel**, high-resolution application icon in the developer console as you upload your program
    - This icon is displayed on the Google Play site to provide a description of the app and does not replace your launcher icon
- Google Play's naming convention for launcher icons**: ic_launcher**

An Action bar icon is considered a logo that represents what the program's function is in a single glance.

Changing the Text Color of Android Controls

- The Android platform uses a color system called **hexadecimal color codes** to display different colors.
- A hexadecimal color code is a triplet of three colors.
    - Colors are specified first by a pound sign followed by how much red (00 to FF),
    - how much green (00 to FF), and
    - how much blue (00 to FF) are in the final color.
        - For example, the hexadecimal color of #FF0000 is a true red.
- Use **hexadecimal color codes** to represent RGB (Red, Green, Blue) values
- Codes range from 00 to FF (00 = none, FF = full)
- Codes are identified by a pound sign, followed by the RGB values
    - **#FF0000 is all RED**
    - **#00FF00 is all GREEN**
    - **#0000FF is all BLUE**
    - **#FFFF00 is YELLOW (RED and GREEN = YELLOW)**

You can change a component's alignment after placing it on the UI by adjusting its **Gravity** settings

Changing the Margins

- **Layout:margin** attribute allows for more flexibility in controlling your layout
- Set density-independent pixel (dp) values instead of “eyeballing” to create equal spaces around components
- Using the same specified margins creates a symmetrical layout
- You can change all margins equally or change the left, top, right, and bottom margins individually

Changing the Layout Gravity

- Linear layout is the default setting on the emulator
- The **Gravity** tool changes the alignment
    - Works like the left, center, right, top or bottom buttons on the Microsoft Office ribbon

Changing the Layout Gravity

- The Gravity tool changes the linear alignment.
- Layout gravity is similar to the alignment feature in Microsoft Office that allows a control to snap to the left, center, right, top, or bottom.
- The gravity property in the Properties pane provides more options in addition to the Gravity tool such as **center_vertical, fill_vertical, fill_horizontal, fill, clip_vertical, clip_horizontal, start, and end.**

RadioButton and RadioGroup Controls

- **RadioButton** controls are used to select or deselect an option.
- When a **RadioButton** is placed on the emulator, by default each control is arranged vertically.
- If you prefer the RadioButton controls to be listed horizontally, you can set the orientation property to horizontal.
- RadioButton controls are typically used together in a **RadioGroup** container.
- Checking one radio button unchecks the other radio buttons within the group.
    - In other words, within a RadioGroup control, only one RadioButton control can be selected at a time.
- The keyword, **final** represents that the variable can be assigned only once. This variable can only be referenced within this class.

Using If Else Statements

- In many applications, the logic requires one set of instructions to be executed if a condition is true and another set of instructions to be executed if a condition is false.
- Code Syntax:

```
if (condition) {
        //Statements completed if condition is true
} else {
        //Statements completed if condition is false
}
```

Examples of the equals and compareTo methods

- In addition to numbers, strings can also be compared in a conditional statement.
- A string value comparison compares each character in two strings, starting with the first character in each string.
- All characters found in strings, including letters, numbers, and special characters, are ranked in a sequence from low to high based on how the characters are coded internally on the computer.
- Java strings are compared with the **equals method** of the String class.

Logical Operators

- An **If statement** can test more than one condition within a single statement.

Toast Notification

- A **toast notification** communicates messages to the user.
- These messages pop up as an overlay on the user's current screen, often displaying a validation warning message.
    - In the application, a toast notification displays a message warning the user that an invalid number was entered.
- A toast message only fills the amount of space required for the message to be displayed while the user's current activity remains visible and interactive.
- The notification automatically fades in and out on the screen.

Using the isChecked( ) Method of RadioButton Controls

- The Java code must check each RadioButton to determine if that RadioButton has been selected by the user.
- This checked property can be tested in an If statement using the **isChecked( ) method** to determine if the RadioButton object has been selected.

IMPLEMENTING SPLASH SCREEN API, AND AUDIO IN ANDROID APPS

Creating a Splash Screen

- Many Android applications on the market show splash screens that often include the name of the program, display a brand logo for the application, or identify the author.
- A splash screen opens as you launch your app, providing time for Android to initialize its resources.
- Extending the length of time that your splash screen is displayed enables your app to load necessary files.

The splash screen shows during:

- **Cold Start**
  - A cold start refers to an app's starting from scratch. •
  - Cold starts happen in cases such as your app launching for the first time since the device booted or since the system killed the app.
- **Warm Start**
  - Warm start means picking up where you left off.
  - The user backs out of your app but then re-launches it
  - The system evicts your app from memory and then the user re-launches it.
- **Hot start**
  - It occurs when you switch back to an app that's already open and actively running in the foreground.
  - It's like switching between tabs in a web browser.

Class Variables

- **Local variables** are declared by variable declaration statements within a method, such as a primitive integer variable within an onCreate( ) method.
- The **scope** of a variable refers to the variable's visibility within a class.
- Variables that are accessible only to a restricted portion of a program, such as a single method, are said to have **local scope.**
- Variables that are accessible from anywhere in a class, however, are said to have **global scope.**
- If a variable is needed in multiple methods within a class, the global variable is assigned at the beginning of a class, not within a method.
- This global scope variable is called a **class variable** in Java and can be accessed by multiple methods throughout the program.

Playing Music

- The Android multimedia framework includes support for playing variety of common media types, so that you can easily integrate audio, video and images into your applications.
- You can play audio or video from media files stored in the application's resources (a folder named raw), from stand-alone files in the Android file system of the device, from an SD (Secure Digital) memory card in the phone itself, or from a data stream provided through an Internet connection.
- The most common file type of media supported for audio playback with the MediaPlayer class is **.mp3**, but other audio file types such as **.wav, .ogg,** and **.midi** are typically supported by most Android hardware

Creating a Raw Folder for Music Files

- In an Android project, music files are typically stored in a new resource directory called raw, which is a subfolder created in the res folder.
- The raw resource directory must be created before music files can be placed in that folder.
- The two .mp3 files played in the Aloha Music app are named ukulele.mp3 and drums.mp3, and they should be placed in the raw folder.

Using the MediaPlayer Class

- The MediaPlayer class provides the methods to control audio playback on an Android device.
- At the beginning of the MainActivity.java code, two MediaPlayer class variables are declared.
- After the variables are declared, an instance of the MediaPlayer class is assigned to each variable.

The MediaPlayer State •

- Android uses the MediaPlayer class to control the playing of the audio file.
- What determines whether the music file is playing is called the state of the MediaPlayer.
- The three common states of the audio file are when the music starts, when the music pauses, and when the music stops.
- The state of the music is established by the MediaPlayer's temporary behavior

Changing the Visibility Property Using Code

- The Java property that determines whether a control is displayed on the emulator is the Visibility property.

- By default, the Visibility property is set to display any control you place on the emulator when the program runs.
- To cause the control not to appear, you must code the setVisibility property to change the view to invisible.

CUSTOM LISTVIEW, ARRAYS, SWITCH STATEMENTS, AND WEB BROWSERS

Creating a List

- Lists are one of the most common designs in mobile apps
- Scrollable
- Selectable
- Programmable to bring up the next Activity (screen)

themes.xml

- Used to define styles and themes for your Android application. Themes control the look and feel of various UI components throughout your app, such as buttons, text fields, and backgrounds.

strings.xml

- Provides a centralized location to store all the string literals used in your application.

activity_main.xml

- The layout file is associated with the main activity of an Android application.

customlistitem.xml

- Used as a layout file to define the appearance of individual items in a custom list. Its purpose is to specify the layout and appearance of each item in the list

Creating an Android Project Using a List

- Opening screen contains a vertical list of Philippine government agencies.
- List is automatically scrollable if it exceeds the window size
- ListView is better than TableLayout View because each row can be selected for further action
- Programmable to bring up the next activity (screen)
- Selecting an item opens up a related Web page or an image of a government agency

Launcher Icon

- The icon that users tap to start your app.

Action Bar

- Also known as the **app bar**, is a dedicated space at the top of the screen that typically contains the app's title, navigation buttons, and other interactive elements.
- Ensure that your activity's theme supports displaying the ActionBar.
- If you're using a theme without an ActionBar, you may need to switch to a theme that includes it, such as **Theme.AppCompat** or one of its variants.

Adding Images to the Resources Folder

- The resources folder **(res/)** is a standardized location for storing various types of resources used by your app.
- Placing images in the **res/drawable** directory organizes them logically and makes them easily accessible to your app's code and layouts.

String Table (Strings.xml)

- By centralizing strings in the **strings.xml** file, you can easily support multiple languages and localize your app for different regions.
- Storing strings in a separate file promotes consistency and makes it easier to manage and update text throughout your app. If you need to modify text content, you can do so in one central location rather than hunting through your codebase for hardcoded strings.

Images must be located in the **drawable -hdpi** folder

- Remember that images may be subject to **copyright laws**

Array: a container object that holds a fixed number of values of a single type

- An **array variable** can store more than one value
    - You can avoid assigning a separate variable for each item in a list for a ListView component
- Each individual item in an array that contains a value is called an **element**
- Arrays provide access to data by using a numeric index, or subscript • Elements of the mainTitle array are accessed as mainTitle[0], mainTitle[1], mainTitle[2], etc.

Declaring an Array

- Declaration includes the array's data type and its name
- Sample arrays of numeric and character values: int[] age={18,21,38,88}; double[] weather={72.3, 65.0, 25.7, 99.5}; char[] initials={'P','N','D’}

MyListAdapter.java

- MyListAdapter acts as a bridge between the data (arrays of titles, sub-titles, and images) and the ListView/RecyclerView, providing a customized view for each item in the list based on the defined layout (customlistitem.xml).
- **Constructor**: This constructor initializes the adapter with the activity context, arrays of main titles, sub-titles, and image resources.
- **super(context, R.layout.customlistitem, mainTitle)** calls the superclass constructor of ArrayAdapter, passing in the activity context, the layout resource for the custom list item **(R.layout.customlistitem),** and the array of main titles.
- The constructor also initializes the context and arrays in the adapter.

ViewHolder Class

- This is a static inner class that holds references to the views of each item in the list.
- It's used to avoid calling **findViewById()** repeatedly when scrolling through the list, thus improving performance.
- It holds references to the views of each item in the list **(titleText, subTitleText, and imageView).**
- A **ViewHolder** pattern is used for efficient view recycling. If the **convertView** is null, a new view is inflated from the **customlistitem.xml** layout, and a ViewHolder object is created to hold references to its views (title text, sub-title text, and image view). The ViewHolder is then set as a tag on the converted view.
- If the **convertView** is not null, it means it's being recycled, so the ViewHolder is retrieved from its tag.
- The data for the current position is retrieved from the arrays and set onto the views held by the ViewHolder

Class Files are needed to display images on the screen when the user selects options

- An **onCreate** method requests that the user interface opens to display an image of the government agency

Switch statement: a decision structure that allows you to choose from many statements based on an integer or single character (char) input

Using the setOnItemClickListener method

- **listView.setOnItemClickListener(...):** This line sets a click listener for items in the ListView named listView.
- **Lambda Expression**: Inside the setOnItemClickListener method, a lambda expression is used to define the behavior when an item in the ListView is clicked.
  • Parameters:

    - **parent**: This parameter refers to the AdapterView where the click happened. In this case, it's the ListView. •
    - **view**: This parameter refers to the specific view within the ListView that was clicked. It represents the item view that was clicked.
    - **position**: This parameter indicates the position of the clicked item in the adapter's data set. It's the position of the item clicked within the ListView.
    - **id**: This parameter represents the row ID of the item that was clicked. It's usually the same as the position parameter, but it can differ if you set custom row IDs in the adapter.

Understanding Android Intents

- When the user selects one of the first two list items in the project, SSS or PNP, a built-in Android browser launches a website about each government agency.
- A browser is launched with Android code using an **intent**.
- Android intents are powerful features that allow apps to talk to each other in a very simple way.
- Android intents send and receive activities and services including:
    - Opening a Web page in a browser
    - Calling a phone number
    - Locating a GPS position on a map
    - Posting notes to a note-taking program
    - Opening your contacts list
    - Sending a photo
    - Posting to a social network

Launching the Browser from an Android Device

- Android phones have a built-in browser with an intent filter that accepts intent requests from other apps
    - The intent sends the browser a **URI** (Uniform Resource Identifier), a string that identifies web resources
    - URI is similar to **URL** (Uniform Resource Locator) : a website address
    - A URI is a URL with additional information necessary for gaining access to the resources required for posting a web page
- The action called **ACTION_VIEW** (must be in CAPS) is what actually displays the page in the browser
    - ACTION_VIEW is the most common action performed on data

SQLITE DATABASE FOR ANDROID INSERT, DELETE, UPDATE AND VIEW DATA IN SQLITE DATABASE

Advantages to saving local data

- **Local Data Storage**
    - Fast app performance
    - Offline cache keeps app working when user has no cell service. Local storage can be a subset of recently accessed records that is part of a huge collection of data.
- **Cloud Data Storage**
    - All devices have a common set of data

SQLite is a DataStore

- Similar to MS Access •
- All components are contained within one file.
- Embedded Relational Database. No installation or drivers required. It is built-in.
- SQLite is an open-source relational database i.e. used to perform database operations on android devices such as storing, manipulating or retrieving persistent data from the database.

Managing SQLite

- Databases and tables are initially created by statements within the app when the app attempts to use the database.

Data Types in SQLite

- **NULL** – null value
- **INTEGER** – signed integer, stored in 1,2,3,4,6 or 8 bytes
- **REAL** – a floating point value.
- **TEXT** – text string
- **BLOB** – Binary Large Object
    - Bits are stored in a table. BLOBs are sometimes used for JPG, MP3, MOV files etc. It is defined as the chunk of binary data being stored as a single entity in a database system. BLOBs are used primarily to hold multimedia objects like images, videos, and sound, though they can also be used to store programs.

Extending Classes

- Java provides the reserved word extends for specifying a hierarchical relationship between two classes.
- For example, suppose you have a Vehicle class and want to introduce a Car class as a kind of Vehicle.
- **class Car extends Vehicle**
    - codifies a relationship that is known as an "is-a" relationship: a car is a kind of vehicle.
    - In this relationship, Vehicle is known as the base class, parent class, or superclass; and Car is known as the derived class, child class, or subclass.
    - Car is capable of inheriting member declarations from its Vehicle superclass.

SQLiteOpenHelper class

- The android.database.sqlite.SQLiteOpenHelper class is used for database creation and version management.
- For performing any database operation, you have to provide the implementation of onCreate() and onUpgrade() methods of SQLiteOpenHelper class.

getWritableDatabase

- Create and/or open a database that will be used for reading and writing.
- The first time this is called, the database will be opened and **onCreate(SQLiteDatabase), onUpgrade(SQLiteDatabase, int, int)** and/or **onOpen(SQLiteDatabase)** will be called.
- Once opened successfully, the database is cached, so you can call this method every time you need to write to the database. (Make sure to call **close()** when you no longer need the database.)
- Errors such as bad permissions or a full disk may cause this method to fail, but future attempts may succeed if the problem is fixed.

ContentValues

- A key/value store that inserts data into a row of a table.
- In most cases, the keys map to the column names of the table, and the values are the data to enter into the table.
- ContentValues are used to insert new rows into tables. Each Content Values object represents a single table row as a map of column names to values

getReadableDatabase

- Create and/or open a database. This will be the same object returned by getWritableDatabase() unless some problem, such as a full disk, requires the database to be opened readonly.
- In that case, a read-only database object will be returned. If the problem is fixed, a future call to getWritableDatabase() may succeed, in which case the read-only database object will be closed and the read/write object will be returned in the future.

CURSOR

- The result set from a query operation is called the cursor.
- We loop through the cursor items to process each line of a search result.
- Holds the result set from a query on a database. An app can read the data from a cursor and display it to a user or perform business logic based on the data contained in the cursor.

StringBuffer

- A thread-safe, mutable sequence of characters.
- A string buffer is like a **String**, but can be modified.
- For example, if z refers to a string buffer object whose current contents are **"start",** then the method call **z.append("le")** would cause the string buffer to contain **"startle",**
- whereas **z.insert(4, "le")** would alter the string buffer to contain **"starlet".**

AlertDialog

- AlertDialog is a lightweight version of a Dialog.
- **setCancelable (boolean cancelable)**
    - Sets whether the dialog is cancelable or not. Default is true.
- **setMessage**
    - Set the message to display
- **show()**
    - Creates an AlertDialog with the arguments supplied to this builder and immediately displays the dialog.