BÁO CÁO LẬP TRÌNH ANDROID

I. Khái niệm về lập trình Android

Lập trình Android là quá trình phát triển ứng dụng và game cho các thiết bị chạy hệ điều hành Android. Android là một hệ điều hành dựa trên Linux được phát triển bởi Google, được sử dụng rộng rãi trên điện thoại di động, máy tính bảng, đồng hồ thông minh và nhiều loại thiết bị di động khác.

Quá trình lập trình Android bao gồm việc sử dụng một ngôn ngữ lập trình như Java hoặc Kotlin, kết hợp với các công cụ phát triển như Android Studio. Các ứng dụng Android có thể được phát triển cho các mục đích khác nhau, từ ứng dụng di động cá nhân đến ứng dụng doanh nghiệp và ứng dụng giải trí.

Lập trình Android không chỉ là một kỹ năng quan trọng trong thế giới công nghệ hiện đại, mà còn mở ra nhiều cơ hội nghề nghiệp và sáng tạo cho những người tham gia.

II. Phương pháp tiếp cận

Đối với những người mới bắt đầu làm quen với môi trường phát triển Android, có những vùng kiến thức trọng điểm cần nắm vững để tạo tiền đề để đi sâu hơn vào các vùng kiến thức khác, bao gồm:

- Ngôn ngữ lập trình: Java hoặc Kotlin
- Công cụ cần thiết: Android Studio, JDK, SDK
- Các kiến thức lập trình Android cơ bản
- Lập trình hướng đối tượng (OOP)

❖ **Bước 1: Chọn ngôn ngữ lập trình Android: Java hay Kotlin?**

Có nhiều ngôn ngữ lập trình có thể phục vụ cho việc phát triển các ứng dụng mobile trên hệ điều hành Android như Java, Kotlin, C++, C#, Dart,… nhưng trong phạm vi bài biết này chỉ đề cập đến hai ngôn ngữ được áp dụng phổ biến nhất là Java và Kotlin.

Java và Kotlin là hai ngôn ngữ lập trình phổ biến nhất được sử dụng để phát triển ứng dụng Android, chiếm ưu thế so với các ngôn ngữ khác nhờ nhiều ưu điểm nổi bật:

- **Lịch sử phát triển**
  - Java: Ngôn ngữ lập trình truyền thống cho Android, ra mắt từ năm 1995, sở hữu cộng đồng rộng lớn, nhiều tài liệu hướng dẫn và thư viện phong phú.
  - Kotlin: Ngôn ngữ trẻ hơn, ra mắt vào năm 2011, được Google chính thức hỗ trợ từ năm 2017, với cú pháp, hiện đại và nhiều tính năng ưu việt.
- **Ưu điểm của cả hai ngôn ngữ trên**

- Hiệu suất: Cả Java và Kotlin đều được biên dịch thành mã bytecode JVM (Java Virtual Machine), mang lại hiệu suất cao và khả năng tương thích tốt với hệ điều hành Android.
- Tính năng: Cung cấp đầy đủ các tính năng cần thiết để phát triển ứng dụng Android, bao gồm lập trình hướng đối tượng, quản lý bộ nhớ, xử lý đa luồng, v.v.
- Cộng đồng: Sở hữu cộng đồng lập trình viên đông đảo, tích cực chia sẻ kiến thức, kinh nghiệm và hỗ trợ lẫn nhau trong quá trình học tập và phát triển.
- Công cụ hỗ trợ: Được cung cấp đầy đủ các công cụ hỗ trợ phát triển như IDE, debugger, test framework, v.v., giúp tối ưu hóa quy trình phát triển ứng dụng.

❖ **Lựa chọn ngôn ngữ nào để học**

- Java: Phù hợp cho người mới bắt đầu, với nhiều tài liệu và hướng dẫn dễ hiểu, cộng đồng hỗ trợ rộng lớn.
- Kotlin: Lựa chọn tối ưu cho lập trình viên có kinh nghiệm, giúp viết code, dễ đọc, dễ bảo trì và phát triển ứng dụng hiệu quả hơn.

Tóm lại, bước đầu tiên và quan trọng nhất trong hành trình lập trình Android của bạn là lựa chọn ngôn ngữ lập trình phù hợp. Mỗi ngôn ngữ đều có ưu và nhược điểm riêng, do đó bạn cần cân nhắc kỹ lưỡng trước khi đưa ra quyết định.

❖ **Bước 2: Cài đặt các công cụ lập trình Android: Android Studio, JDK, SDK**

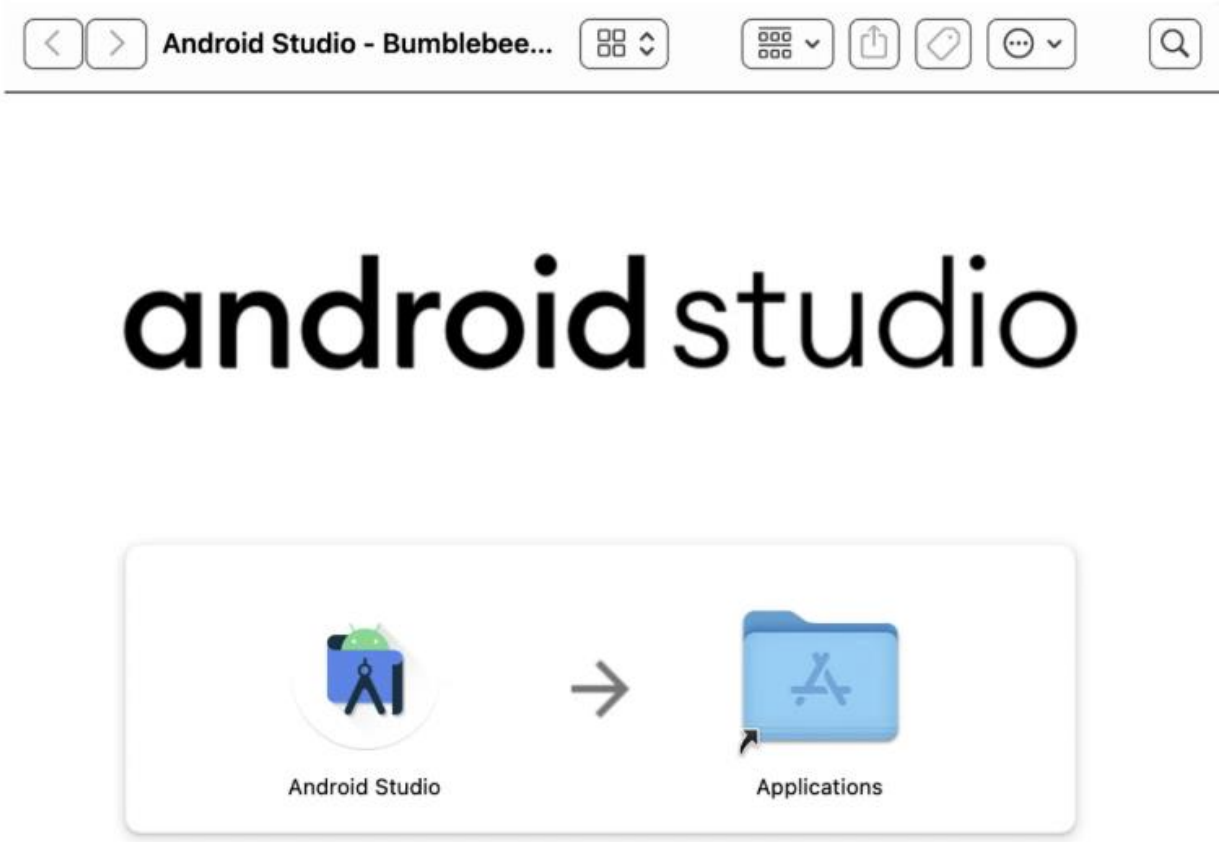
Để bắt đầu hành trình lập trình Android, bạn cần cài đặt các công cụ sau:

- **Android Studio**

Android Studio là môi trường phát triển tích hợp (IDE) chính thức cho hệ điều hành Android được Google phát triển. Nó cung cấp một loạt các công cụ và tính năng giúp bạn xây dựng, thử nghiệm và triển khai ứng dụng Android một cách hiệu quả.

Các bước cài đặt Android Studio:

- Tải xuống: Truy cập trang web chính thức của Android Studio và tải phiên bản phù hợp với hệ điều hành của bạn (Windows, macOS, Linux).
- Cài đặt: Chạy trình cài đặt và làm theo hướng dẫn trên màn hình. Chọn thư mục cài đặt, đồng ý với điều khoản cấp phép và cài đặt các thành phần cần thiết.
- Thiết lập SDK và JDK: Trong quá trình cài đặt Android Studio, bạn sẽ được hướng dẫn cài đặt SDK (Software Development Kit) và JDK (Java Development Kit). Nếu bạn đã cài đặt sẵn, hãy chọn thư mục cài đặt phù hợp.



Giao diện Android Studio.

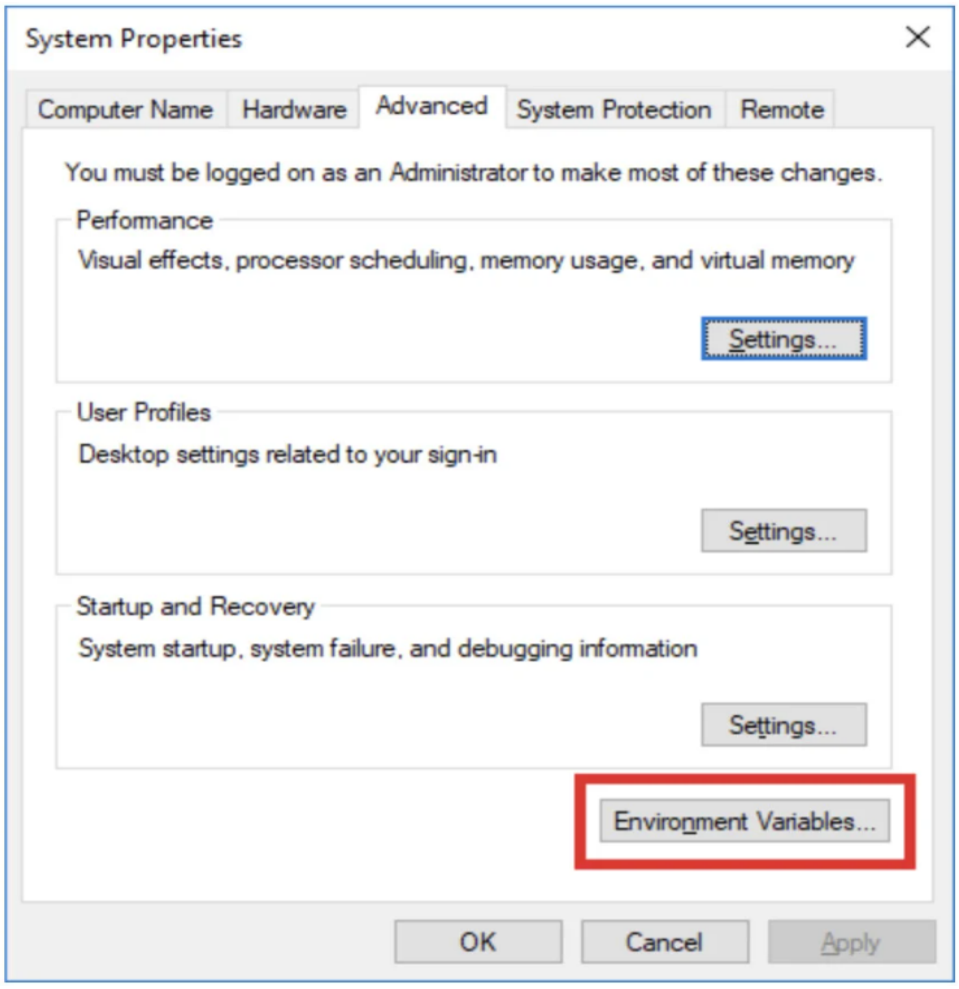
- **JDK (Java Development Kit)**

JDK là viết tắt của Java Development Kit, là một bộ công cụ phần mềm được cung cấp bởi Oracle (trước đây là Sun Microsystems) để phát triển ứng dụng Java. JDK bao gồm các thành phần sau:

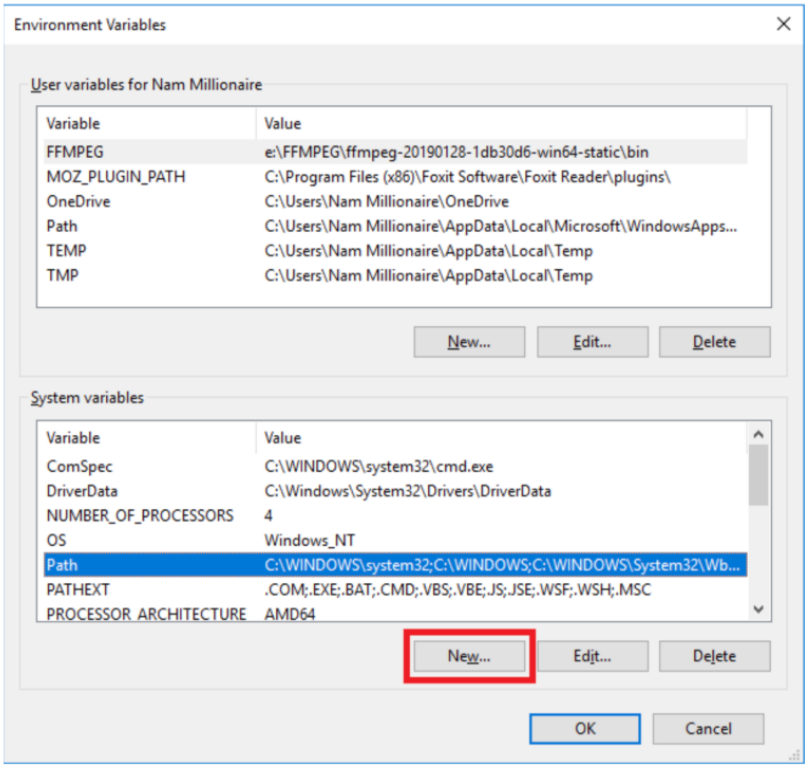
- Trình biên dịch Java (javac): Biên dịch mã nguồn Java (.java) thành mã bytecode (.class).
- Java Runtime Environment (JRE): Cung cấp môi trường để chạy ứng dụng Java.
- Công cụ Java (javadoc, jar): Hỗ trợ tạo tài liệu API, đóng gói ứng dụng.
- Thư viện Java: Cung cấp các lớp và API sẵn có để sử dụng trong phát triển ứng dụng.

Các bước cài đặt JDK:

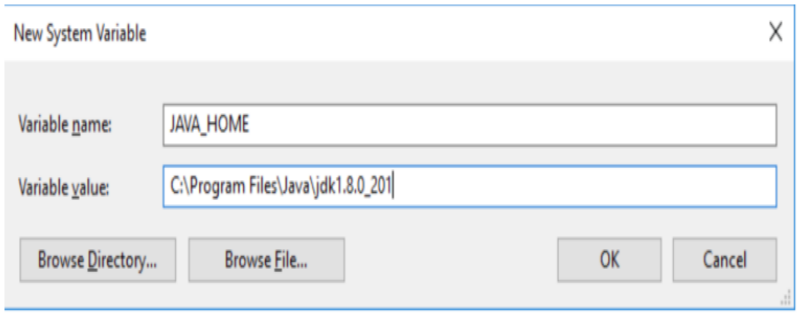
- Tải xuống: Truy cập trang web chính thức của Oracle và tải phiên bản JDK phù hợp với hệ điều hành của bạn (Windows, macOS, Linux).
- Cài đặt: Chạy trình cài đặt và làm theo hướng dẫn trên màn hình. Chọn thư mục cài đặt và cài đặt JDK.
- Thiết lập biến môi trường: Sau khi cài đặt JDK, bạn cần thiết lập biến môi trường JAVA_HOME để Android Studio có thể xác định vị trí cài đặt.
- Windows:
- Nhấp chuột phải vào “This PC” -> “Properties” -> “Advanced system settings” -> “Environment Variables”.



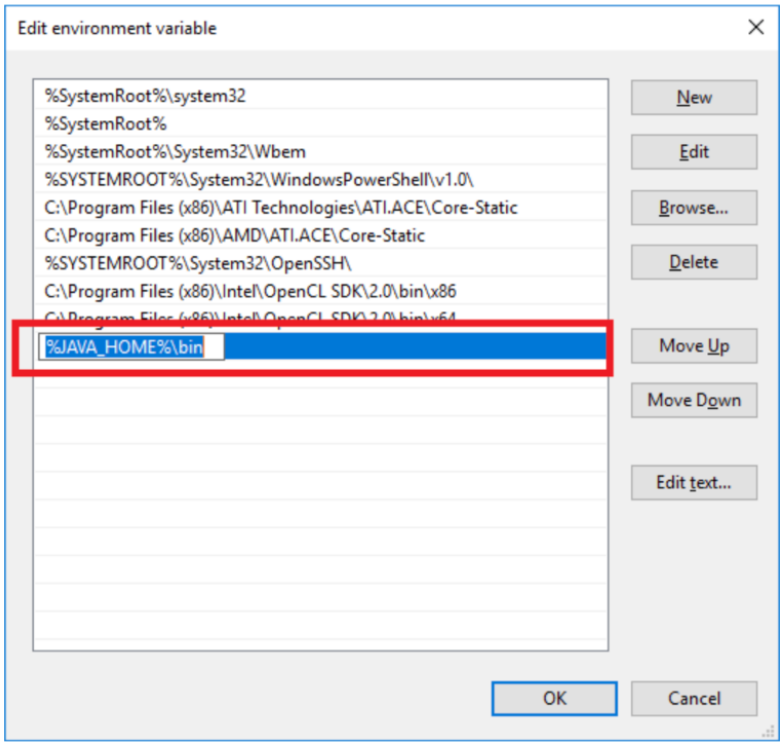
- Trong phần “System variables”, chọn “New”.



- Trong ô "Variable name", nhập "JAVA_HOME".



- Trong ô "Variable value", nhập đường dẫn đến thư mục cài đặt JDK (ví dụ: C:\Program Files\Java\jdk17).
- Nhấp vào "OK" để lưu thay đổi.



Ngoài JDK đối với lập trình viên sử dụng C++ thì cần có thể cần tải NDK. NDK (Native Development Kit) là bộ công cụ để phát triển ứng dụng Android bằng ngôn ngữ C++. NDK cung cấp trình biên dịch C++ (clang) để biên dịch mã nguồn C++ (*.cpp) sang mã máy có thể được thực thi trực tiếp trên CPU của thiết bị Android.

- **SDK (Software Development Kit)**

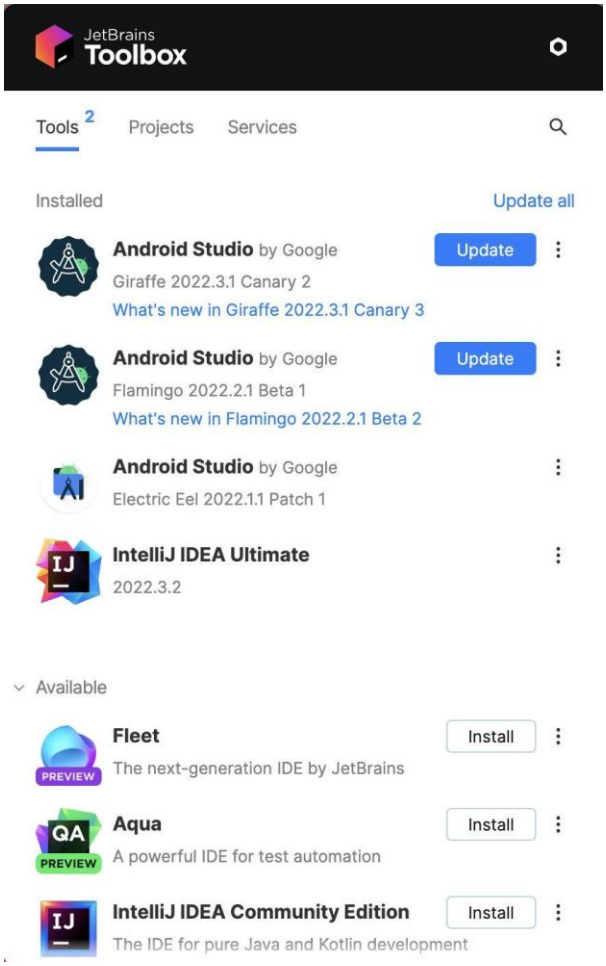
SDK là viết tắt của Software Development Kit, là bộ công cụ phát triển phần mềm cung cấp cho lập trình viên các thành phần cần thiết để xây dựng ứng dụng cho một nền tảng cụ thể. Trong lập trình Android, SDK đóng vai trò quan trọng giúp bạn:

- Truy cập API Android: SDK cung cấp các API (Application Programming Interface) cho phép bạn tương tác với các tính năng của hệ điều hành Android, như camera, GPS, cảm biến, v.v.
- Sử dụng thư viện sẵn có: SDK bao gồm nhiều thư viện mã nguồn mở giúp bạn thực hiện các chức năng phổ biến như xử lý hình ảnh, mạng, lưu trữ dữ liệu, v.v.
- Tích hợp với các dịch vụ Google: SDK hỗ trợ tích hợp với các dịch vụ Google như Google Maps, Google Play Services, v.v.

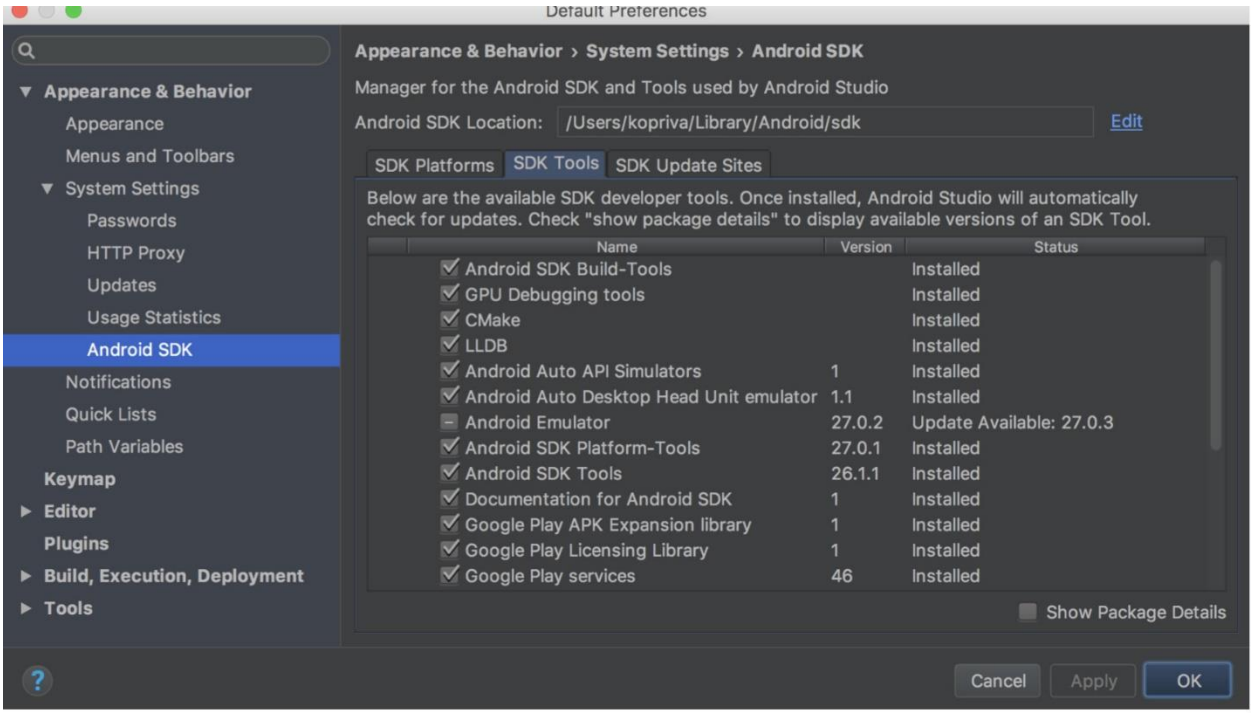
- Thử nghiệm và gỡ lỗi ứng dụng: SDK cung cấp các công cụ để bạn thử nghiệm và gỡ lỗi ứng dụng trên thiết bị ảo hoặc thiết bị thực.

Android Studio tự động cài đặt SDK trong quá trình cài đặt. Bắt đầu sử dụng SDK với các bước sau:

- Mở SDK Manager:
  - Khởi động Android Studio.
  - Chọn “File” -> “Settings” -> “Appearance & Behavior” -> “System Settings” -> “Android SDK”.
  - Nhấp vào “SDK Manager” trong tab “SDK Platforms”.
- Cài đặt các thành phần SDK cần thiết: Trong SDK Manager, chọn các thành phần SDK bạn muốn cài đặt, bao gồm:
  - Platforms: Chọn phiên bản Android mà bạn muốn phát triển ứng dụng (ví dụ: Android 13).
  - Build Tools: Chọn phiên bản build tools mới nhất.



- Android SDK Platform Tools: Cung cấp các công cụ để tương tác với thiết bị Android.
- Android Emulator: Giả lập thiết bị Android để test ứng dụng.



- Google USB Driver: Cần thiết để kết nối thiết bị Android với máy tính.

- Nhấp vào "Install" để cài đặt các thành phần được chọn.

❖ **Bước 3: Tìm hiểu cú pháp Android cơ bản**

Các khái niệm liên quan đến cú pháp Android cơ bản mà bạn cần tìm hiểu là:

- Biến (Variable): Hiểu rõ các loại biến khác nhau (số nguyên, chuỗi, boolean, v.v.) và cách sử dụng chúng để lưu trữ dữ liệu.
- Kiểu dữ liệu (Value Type): Nắm vững các kiểu dữ liệu cơ bản (int, double, String, v.v.) và cách sử dụng chúng một cách chính xác.
- Toán tử (Operators): Khám phá các toán tử toán học, logic và so sánh để thực hiện các phép tính và thao tác dữ liệu.
- Điều kiện (Conditions): Sử dụng các câu lệnh điều kiện (if, else, switch) để kiểm soát luồng chương trình dựa trên các điều kiện nhất định.
- Vòng lặp (Loop): Lặp lại một khối code nhiều lần bằng các vòng lặp (for, while, do-while) để xử lý dữ liệu hiệu quả.

❖ Bước 4: Học về Cấu trúc dữ liệu và giải thuật

Cấu trúc dữ liệu và giải thuật đóng vai trò thiết yếu trong lập trình, giúp bạn tổ chức và quản lý dữ liệu một cách hiệu quả, tối ưu hóa hiệu suất ứng dụng Android. Nắm vững kiến thức về hai lĩnh vực này sẽ giúp bạn xây dựng những ứng dụng Android thông minh, mượt mà và có thể xử lý lượng dữ liệu lớn một cách nhanh chóng.

- **Cấu trúc dữ liệu**

Cấu trúc dữ liệu là cách thức sắp xếp và lưu trữ dữ liệu trong bộ nhớ máy tính. Nắm vững các cấu trúc dữ liệu phổ biến như mảng (Array), danh sách liên kết (Linked List), ngăn xếp (Stack) và hàng đợi (Queue) sẽ giúp bạn lựa chọn cấu trúc phù hợp cho từng bài toán cụ thể, tối ưu hóa việc truy cập và thao tác dữ liệu.

- Mảng (Array): Cấu trúc dữ liệu đơn giản nhất, lưu trữ một tập hợp các phần tử cùng kiểu dữ liệu. Mảng có ưu điểm là cho phép truy cập phần tử nhanh chóng bằng chỉ số, nhưng việc thêm, xóa phần tử giữa mảng có thể tốn kém về thời gian.
- Danh sách liên kết (Linked List): Cấu trúc dữ liệu linh hoạt, lưu trữ các phần tử (node) được kết nối với nhau bằng liên kết. Danh sách liên kết dễ dàng thêm, xóa phần tử ở bất kỳ vị trí nào, nhưng truy cập phần tử bằng chỉ số có thể chậm hơn so với mảng.
- Ngăn xếp (Stack): Với ngăn xếp, cấu trúc dữ liệu hoạt động theo nguyên tắc LIFO (Last In, First Out), nghĩa là phần tử được thêm vào sau sẽ là phần tử đầu tiên được lấy ra.. Ngăn xếp thường được sử dụng trong nhiều ứng dụng, bao gồm việc quản lý ngắn xếp của các hàm trong lời gọi hàm (call stack), duyệt cây trong giải thuật DFS (Duyệt theo chiều sâu), quản lý bộ nhớ và các vấn đề liên quan đến việc lưu trữ và truy xuất dữ liệu theo nguyên tắc LIFO.
- Hàng đợi (Queue): Với hàng đợi, cấu trúc dữ liệu hoạt động theo nguyên tắc FIFO (First In, First Out), nghĩa là phần tử được thêm vào trước sẽ là phần tử đầu tiên được lấy ra. Hàng đợi thường được sử dụng để quản lý danh sách chờ, xử lý các tác vụ theo thứ tự.

- **Giải thuật**

Giải thuật là một tập hợp các hướng dẫn được xác định rõ ràng để giải quyết một vấn đề cụ thể. Nắm vững các thuật toán sắp xếp và tìm kiếm hiệu quả sẽ giúp bạn tối ưu hóa hiệu suất ứng dụng Android, đặc biệt khi xử lý lượng dữ liệu lớn.

- Thuật toán sắp xếp (Sorting algorithm): Sắp xếp các phần tử trong một tập hợp theo một thứ tự nhất định (ví dụ: sắp xếp theo thứ tự tăng dần, giảm dần). Một

số thuật toán sắp xếp phổ biến bao gồm Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, Quick Sort.

- Thuật toán tìm kiếm (Search algorithm): Tìm kiếm một phần tử cụ thể trong một tập hợp dữ liệu. Một số thuật toán tìm kiếm phổ biến bao gồm Linear Search, Binary Search.

❖ **Bước 5: Tìm hiểu các thành phần UI (User Interface – Giao diện người dùng) cơ bản**

Một số khái niệm UI cơ bản trong phát triển ứng dụng Android bao gồm cấu trúc và cú pháp XML để mô tả giao diện người dùng, cũng như các khái niệm về hoạt động (Activity), ý định (Intent), đoạn mã (Fragment), và giao diện người dùng (UI) trong Android:

- XML: XML được sử dụng trong Android để xây dựng giao diện người dùng (UI) của ứng dụng. Cấu trúc và cú pháp XML để mô tả giao diện người dùng Android. Dưới đây là một ví dụ đơn giản về cấu trúc XML cho một layout:

```xml
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, World!" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click me"
        android:layout_below="@id/textView" />

</RelativeLayout>
```

- Activity: Activity là một thành phần của ứng dụng Android, nó chịu trách nhiệm hiển thị giao diện người dùng và tương tác với người dùng. Mỗi Activity thường tương ứng với một màn hình trong ứng dụng.
- Intent: Intent là một đối tượng dùng để kết nối các thành phần của ứng dụng Android trong quá trình chạy. Intent có thể được sử dụng để chuyển dữ liệu giữa các hoạt động (Activity) hoặc khởi chạy các dịch vụ (Service).

- Fragment: Fragment là một phần của giao diện người dùng hoạt động một cách độc lập, có thể được thêm vào hoặc loại bỏ trong một Activity. Fragment giúp tách biệt giao diện người dùng thành các phần nhỏ hơn, dễ quản lý hơn.
- Lifecycle của ứng dụng Android: Lifecycle của ứng dụng Android bao gồm một loạt các trạng thái mà ứng dụng có thể chuyển qua từ khi khởi động đến khi kết thúc. Các phương thức lifecycle quan trọng nhất bao gồm onCreate(), onStart(), onResume(), onPause(), onStop(), và onDestroy().

- **Giao diện người dùng (UI):**
  - Thiết kế layout bằng XML và Java/Kotlin: Layout XML được sử dụng để xây dựng giao diện người dùng, trong khi Java hoặc Kotlin được sử dụng để tương tác và điều khiển các thành phần trong layout.
  - Sử dụng các widget và control cơ bản: Android cung cấp một loạt các widget và control như TextView, Button, EditText, ListView, RecyclerView, và nhiều hơn nữa để xây dựng giao diện người dùng.
  - Xử lý sự kiện người dùng: Sự kiện người dùng như nhấn nút, chạm vào màn hình, hoặc nhập liệu được xử lý thông qua các phương thức callback như onClick(), onTouchListener(), và onKeyListener().

❖ **Bước 6: Tìm hiểu về các Cơ sở dữ liệu**

Trong lập trình Android, cơ sở dữ liệu thường được sử dụng để lưu trữ và quản lý dữ liệu của ứng dụng. Các cơ sở dữ liệu phổ biến nhất trong Android là SQLite và Room.

- **SQLite**

SQLite là giải pháp lưu trữ dữ liệu phổ biến cho ứng dụng Android. Nó cho phép bạn lưu trữ dữ liệu cấu trúc một cách hiệu quả, truy vấn dữ liệu bằng SQL và thực hiện các thao tác CRUD (Create, Read, Update, Delete).

  - Tạo và quản lý cơ sở dữ liệu: Sử dụng SQLiteOpenHelper để tạo, mở, truy cập và đóng cơ sở dữ liệu.
  - Thực hiện truy vấn SQL: Sử dụng SQLiteDatabase để thực hiện các truy vấn SQL như SELECT, INSERT, UPDATE, DELETE.
  - Sử dụng ContentProvider: Cung cấp truy cập dữ liệu cho các ứng dụng khác.

- **SharedPreferences**

SharedPreferences là giải pháp lưu trữ dữ liệu đơn giản, nhẹ và dễ sử dụng. Nó phù hợp cho việc lưu trữ các cài đặt ứng dụng, thông tin người dùng đơn giản.

  - Lưu trữ dữ liệu: Sử dụng các phương thức put(), get(), remove() để lưu trữ, truy xuất và xóa dữ liệu.

- Theo dõi các thay đổi của dữ liệu: Sử dụng SharedPreferences.OnSharedPreferenceChangeListener để theo dõi thay đổi dữ liệu.

❖ **Bước 7: Hiểu cách sử dụng các thư viện hỗ trợ**

Học Android không chỉ là việc hiểu về cú pháp của ngôn ngữ lập trình Kotlin hoặc Java và cách xây dựng giao diện người dùng, mà còn là hiểu cách sử dụng các thư viện hỗ trợ để giảm thiểu thời gian phát triển, tăng tính bảo mật, cải thiện hiệu suất và tạo ra các ứng dụng mạnh mẽ và linh hoạt.

- **Retrofit**

Retrofit là thư viện mạng phổ biến dành cho Android, giúp bạn truy cập và quản lý các API một cách đơn giản và hiệu quả. Với Retrofit, bạn có thể:

- Tạo các request API: Xác định URL, phương thức HTTP, header, body và các tham số cho request.
- Gửi request API: Gửi request đến server và nhận response.
- Xử lý response API: Phân tích dữ liệu JSON, chuyển đổi dữ liệu sang object, xử lý lỗi API.

Retrofit cung cấp nhiều tính năng hữu ích như:

- Hỗ trợ nhiều phương thức HTTP: GET, POST, PUT, DELETE,…
- Hỗ trợ định dạng dữ liệu JSON và XML.
- Hỗ trợ converter để chuyển đổi dữ liệu JSON sang object.
- Hỗ trợ interceptor để chèn header, token, logging,…

- **Room**

Room là một thư viện của Android Jetpack giúp làm việc với cơ sở dữ liệu SQLite dễ dàng hơn và an toàn hơn. Nó cung cấp một lớp trừu tượng hóa cơ sở dữ liệu, các DAO (Data Access Object) để thực hiện các thao tác dữ liệu, và xử lý mọi thứ trên một luồng không chính xác (background thread).

Room giúp giảm bớt công việc lặp đi lặp lại khi làm việc với SQLite và giúp bảo vệ dữ liệu của bạn trước các lỗi phổ biến như SQL injection.

Đoạn mã dưới đây mô tả ngắn gọn 1 ví dụ là việc triển khai Room trong việc lưu trữ dữ liệu trong các ứng dụng Android:

```kotlin
// Định nghĩa entity (bảng) trong Room
@Entity
data class User(
    @PrimaryKey val uid: Int,
    @ColumnInfo(name = "first_name") val firstName: String?,
    @ColumnInfo(name = "last_name") val lastName: String?
)

// Định nghĩa DAO (Data Access Object) trong Room
@Dao
interface UserDao {
    @Query("SELECT * FROM user")
    fun getAll(): List<User>

    @Insert
    fun insertAll(vararg users: User)
}

// Khai báo cơ sở dữ liệu với RoomDatabase
@Database(entities = [User::class], version = 1)
abstract class AppDatabase : RoomDatabase() {
    abstract fun userDao(): UserDao
}
```

Trên đây là một cái nhìn tổng quan về những kiến thức cơ bản trong lập trình Android, được đúc kết từ quá trình học và phát triển các sản phẩm, dự án lớn nhỏ khác nhau.

Tiếp theo, việc áp dụng những kiến thức này vào thực tế thông qua việc thực hiện các dự án và sản phẩm Android sẽ giúp bạn củng cố và mở rộng hiểu biết của mình, đồng thời phát triển kỹ năng thực hành và tư duy lập trình.

Bước 8: Xây dựng các ứng dụng Android đơn giản

Xây dựng ứng dụng Android đơn giản là một cách tuyệt vời để áp dụng các kiến thức đã học vào thực tế. Bạn có thể chọn ba ứng dụng rất phổ biến và hữu ích để bắt đầu: To-do list, danh bạ, và máy tính.

- ❖ **Bước 9: Tìm hiểu về Lập trình hướng đối tượng (OOP)**

```
class Car {
    String color;
    int year;

    void drive() {
        System.out.println("Driving");
    }
}

Car myCar = new Car();
myCar.color = "Red";
myCar.year = 2022;
myCar.drive();
```

Trên đây là một ví dụ mô tả việc khởi tạo một đối tượng (Object) tên là "Car" có hai thuộc tính là "Color" đại diện cho màu xe, "year" là năm xe được sản xuất, phương thức "drive" đại diện cho trạng thái xe đang chạy.

- **Điểm chính của OOP**
  - Lớp (Class): Khuôn mẫu để tạo ra các đối tượng, đóng gói dữ liệu (thuộc tính) và hành vi (phương thức) của đối tượng.
  - Đối tượng (Object): Phiên bản cụ thể của một lớp, đại diện cho một thực thể trong thế giới thực.
  - Tính đóng gói (Encapsulation): Ẩn giấu chi tiết triển khai bên trong lớp, chỉ để lộ giao diện thông qua các phương thức.
  - Tính kế thừa (Inheritance): Khả năng cho phép một lớp kế thừa các thuộc tính và phương thức từ lớp cha, giúp tái sử dụng code và phát triển các lớp mới dễ dàng hơn.
  - Tính đa hình (Polymorphism): Khả năng cho phép các đối tượng thuộc các lớp khác nhau nhưng có cùng giao diện thực hiện cùng một hành động theo cách riêng của chúng.
  - Tính trừu tượng (Abstraction): Tập trung vào các đặc điểm thiết yếu của một đối tượng, bỏ qua các chi tiết phức tạp, giúp đơn giản hóa code và dễ dàng sử dụng hơn.
- **Lợi ích của OOP**
  - Dễ quản lý: Việc chia nhỏ code thành các lớp và đối tượng giúp dễ dàng quản lý và sửa đổi code, đặc biệt là cho các dự án lớn.
  - Tái sử dụng: Các lớp có thể được tái sử dụng trong nhiều dự án khác nhau, tiết kiệm thời gian và công sức cho lập trình viên.

- Dễ bảo trì: Việc sửa lỗi và cập nhật code trở nên dễ dàng hơn nhờ tính đóng gói và tính trừu tượng.
- Mở rộng: Dễ dàng mở rộng ứng dụng bằng cách thêm các lớp và đối tượng mới.

- **Ví dụ về OOP trong lập trình Android**
  - Lớp Activity: Đại diện cho một màn hình trong ứng dụng Android.
  - Lớp View: Đại diện cho các thành phần giao diện người dùng như Button, TextView, ImageView,…
  - Kế thừa: Lớp Button có thể kế thừa từ lớp View, kế thừa các thuộc tính và phương thức chung của các thành phần giao diện người dùng.
  - Đa hình: Phương thức onClick() được định nghĩa trong lớp View có thể được thực thi theo cách khác nhau trong các lớp Button, TextView,…

OOP là một chủ đề rộng lớn và cần thời gian để học hỏi. Bạn có thể tham khảo tài liệu online, video hướng dẫn và sách để học về OOP, hoặc bạn có thể tham gia các cộng đồng lập trình để trao đổi kiến thức và kinh nghiệm với những người khác.

III. Cách thức thực hiện

Để tự học lập trình Android cho người mới bắt đầu hiệu quả, bạn cần lập kế hoạch học tập phù hợp với bản thân. Dưới đây là một số bước giúp bạn lập kế hoạch tự học lập trình Android:

❖ Xác định mục tiêu

- Xác định lý do bạn muốn học lập trình Android.
- Xác định mục tiêu cụ thể bạn muốn đạt được trong thời gian bao lâu.

❖ Đánh giá kiến thức hiện tại

- Xác định bạn đã biết gì về lập trình và Android.
- Xác định những kiến thức bạn cần học thêm.

❖ Lên lịch học tập

- Chia nhỏ mục tiêu thành các mục tiêu nhỏ hơn và dễ thực hiện hơn.
- Lên lịch học tập cụ thể cho từng ngày, bao gồm thời gian học tập và nội dung học tập.