**TRƯỜNG ĐẠI HỌC KỸ THUẬT - CÔNG NGHỆ CẦN THƠ**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**ĐỒ ÁN HỌC PHẦN**

**ĐỀ TÀI**

**VIẾT MỘT ỨNG DỤNG BẰNG JAVA**

**SỬ DỤNG (DESIGN PATTERN) ADAPTER**

**HỘI ĐỒNG:**

**GVHD: Hà Xuân Sơn**

**GVPB: ………………………………………**

**---o0o---**

**SVTH : Thạch Thắng (1500122)**

**Cần Thơ, ngày 31 tháng 05 năm 2018**

**NHẬN XÉT GIẢNG VIÊN HƯỚNG DẪN**

Cần Thơ, ngày … tháng 05 năm 2018

GIẢNG VIÊN HƯỚNG DẪN

**NHẬN XÉT GIẢNG VIÊN PHẢN BIỆN**

Cần Thơ, ngày … tháng 05 năm 2018

GIẢNG VIÊN PHẢN BIỆN

**LỜI CAM ĐOAN**

**LỜI CẢM ƠN/ LỜI NGỎ**

Sau thời gian thực hiện làm đề tài đến nay em đã hoàn thành đề tài của mình. Để có được kết quả như hôm nay, em thực hiện đề tài đã nhận được rất nhiều sự quan tâm, giúp đỡ và chỉ đạo nhiệt tình của các thầy, cô trong khoa công nghệ thông tin.

Đặc biệt em thực hiện đề tài xin gửi lời cảm ơn chân thành nhất tới thầy “Hà Xuân Sơn”.Thầy là người đã tận tình chỉ bảo, chia sẻ cho em những kiến thức bổ ích hướng dẫn động viên trong suốt thời gian qua, tạo mọi điều kiện tốt nhất cho em trong quá trình thực hiện.

Mặc dù em thực hiện đề tài đã rất cố gắng trong quá trình làm đề tài song đồ án của em không thể tránh khỏi những thiếu sót, hạn chế. Em thực hiện đề tài rất mong nhận được sự đóng góp chân thành từ các thầy, cô và các bạn để đề tài của em được hoàn thiện hơn.

*Em thực hiện đề tài xin chân thành cảm ơn!*

**MỤC LỤC**

[**NHẬN XÉT GIẢNG VIÊN HƯỚNG DẪN** 2](#_Toc515613683)

[**NHẬN XÉT GIẢNG VIÊN PHẢN BIỆN** 3](#_Toc515613684)

[**LỜI CAM ĐOAN** 4](#_Toc515613685)

[1. Tổng quan 8](#_Toc515613686)

[2. Phạm vi nghiên cứu, phạm vi ứng dụng 8](#_Toc515613687)

[3. Đối tượng nghiên cứu 8](#_Toc515613688)

[4. Mục đích nghiên cứu 8](#_Toc515613689)

[5. Nhiệm vụ nghiên cứu 9](#_Toc515613690)

[6. Phương pháp nghiên cứu 9](#_Toc515613691)

[7. Chức năng, công việc đã làm được 9](#_Toc515613692)

[**I.** **Tìm hiểu Java** 9](#_Toc515613693)

[1. **Tổng quan về Java** 9](#_Toc515613694)

[a) Lý do ra đời Java 9](#_Toc515613695)

[b) Java là gì? 9](#_Toc515613696)

[2. Đặc điểm 10](#_Toc515613697)

[2.1 Đơn giản 10](#_Toc515613698)

[2.2 Hướng đối tượng 10](#_Toc515613699)

[2.3 Độc lập phần cứng và hệ điều hành 10](#_Toc515613700)

[2.4 Mạnh mẽ 10](#_Toc515613701)

[2.5 Bảo mật 11](#_Toc515613702)

[2.6 Phân tán 11](#_Toc515613703)

[2.7 Đa luồng 11](#_Toc515613704)

[2.8 Linh động 11](#_Toc515613705)

[**II.** **Tìm hiểu về Desgin Pattern** 12](#_Toc515613706)

[1. Design Pattern là gì ? 12](#_Toc515613707)

[2. Lợi ích của Design Pattern 12](#_Toc515613708)

[3. Phân loại Design Pattern 12](#_Toc515613709)

[**III.** **Tìm hiểu về android** 13](#_Toc515613710)

[1. Android là gì? 13](#_Toc515613711)

[2. Cài đặt Android Studio 13](#_Toc515613712)

[**IV.** **Phân tích hệ thống** 15](#_Toc515613713)

[1. Sơ đồ Use Case 15](#_Toc515613714)

[2. Các giải thuật 16](#_Toc515613715)

[3. Phân quyền trong Android 16](#_Toc515613716)

[**V.** **Kết luận và hướng phát triển** 20](#_Toc515613717)

[1. Kết quả đạt được: 20](#_Toc515613718)

[2. Hạn chế: 24](#_Toc515613719)

[**VI.** **Tài liệu kham khảo:** 25](#_Toc515613720)

[1. Youtube dạy lập trình hướng Android: 25](#_Toc515613721)

[2. Design Pattern in Java Tutorial 25](#_Toc515613722)

**Tóm tắt dồ án**

1. Tổng quan

* Ngày nay, xu thế của xã hội thông tin đang chuyển dần sang lĩnh vực công nghệ di động nói chung và điện thoại di đông nói riêng. Minh chứng cho điều đó là sự ra đời ngày một đa dạng của 2 hệ điều hành đang nắm giữ thị phần di động hiện nay là Android và iOS. Tuy nhiên, xét về khả năng tùy biến cao thì người ta luôn nghĩ ngay đến hệ điều hành Android.
* Android là một nền tảng phần mềm mở dựa trên nhân hệ điều hành Linux và đánh giá là một nền tảng công nghệ phổ biến trong nhiều lĩnh vực trong tương lai. Androi thực sự làm một mảnh đất màu mỡ mà các công ty phần mềm khai thác. Điều này cũng kéo theo việc cơ hội việc làm cho sinh viên mới ra trường cũng nhiều hơn. Ngoài ra, Android là một nền tảng mở và được viết bằng ngôn ngữ Java nên đây là một điểm thuận lợi cho nhiều người có thể dễ dàng bắt tay vào nghiên cứu, cũng như dễ dàng tiếp cận công nghệ.
* Ngày nay, việc sử dụng một chiếc điện thoại di động không chỉ đơn thuần là để gọi và nhắn tin mà thêm vào đó là rất nhiều chức năng tiện ích khác giúp người dùng có thể quản lý công việc một cách hiệu quả và an toàn nhất. Một trong số đó là việc quản lý các Contact (danh bạ) trong điện thoại.
* Trong đồ án môn học này, em đã tìm hiểu và xây dụng ứng dụng Quản lý danh bạ trên Android có áp dụng mô hình Design Pattern mà cụ thể là Adapter.

1. Phạm vi nghiên cứu, phạm vi ứng dụng

* Phạm vi nghiên cứu : tự nghiên cứu.
* Phạm vi ứng dụng: mỗi người

1. Đối tượng nghiên cứu

* Java
* Chương trình quản lý danh bạ trên Android
* Công cụ xây dựng phần mềm: Android Studio

## Mục đích nghiên cứu

* Xây dựng được tài liệu giúp tìm hiểu về ngôn ngữ Java
* Xây dựng được chương trình quản lý danh bạ áp dụng được mô hình Adapter trong Design Pattern

## Nhiệm vụ nghiên cứu

* Tìm hiểu về Desgin Pattern (Adapter).
* Tìm hiều Java Android để áp dụng vào xây dựng chương trình.

## Phương pháp nghiên cứu

* Tìm hiểu hết tổng hợp kiến thức chung của Java , sau đó đi sâu vào những kiến thức cụ thể để xây dựng.
* Nghiên cứu về mẫu Design Pattern để áp dụng.

1. Chức năng, công việc đã làm được

|  |  |  |
| --- | --- | --- |
| STT | Chức năng, công việc | Đã làm được |
| 1 | Tạo giao diện thiện, dễ sử dụng. | √ |
| 2 | Thêm danh bạ bất kỳ | √ |
| 3 | Thực hiện gọi | √ |
| 4 | Gửi SMS | √ |

1. **Tìm hiểu Java**
2. **Tổng quan về Java**
3. Lý do ra đời Java

Java được hình thành bởi James Gosling, Patrick Naughton, Chris Warth, Ed Frank, và Mike Sheridan tại Sun Microsystems năm 1991. Sau hơn 18 tháng để phát triển phiên bản đầu tiên. Ngôn ngữ này lúc đầu được gọi là “Oak” nhưng sau đó được đổi tên thành Java vào năm 1995. Ở giữa khoảng thời gian của phiên bản đầu tiên năm 1992 và thông báo công khai của Java vào năm 1995, nhiều người đã đóng góp vào sự thiết kế và tiến hóa của ngôn ngữ này..

1. Java là gì?

Java là ngôn ngữ lập trình hướng đối tượng, do vậy không thể dùng Java để viết một chương trình hướng chức năng. Java có thể giải quyết hầu hết các công việc mà các ngôn ngữ khác có thể làm được.

Java là ngôn ngữ vừa biên dịch vừa thông dịch. Đầu tiên mã nguồn được biên dịch bằng công cụ JavaC để chuyển thành dạng ByteCode. Sau đó được thực thi trên từng loại máy cụ thể nhờ chương trình thông dịch.

## Đặc điểm

Java là ngôn ngữ lập trình được phát triển từ ngôn ngữ lập trình C/C++. Nó kế thừa, phát huy các thế mạnh của ngôn ngữ C/C++ và lược bỏ đi các cú pháp phức tạp của C/C++. Ngôn ngữ Lập trình có một số đặc trưng tiêu biểu: đơn giản, hướng đối tượng, độc lập phần cứng và hệ điều hành, mạnh mẽ, bảo mật, phân tán, đa luồng và linh động

1. Đơn giản

Những người thiết kế mong muốn phát triển một ngôn ngữ dễ học và quen thuộc với đa số người lập trình. Do vậy, Java được loại bỏ các đặc trưng phức tạp của C và C++ như :

Loại bỏ thao tác con trỏ, thao tác định nghĩa chồng toán tử

Không cho phép đa kế thừa mà sử dụng các giao diện

Không sử dụng lệnh “goto” cũng như header (.h)

Loại bỏ cấu trúc “struct” và “union”

1. Hướng đối tượng

Java là ngôn ngữ lập trình thuần hướng đối tượng, mọi chương trình viết trên Java đều phải được xây dựng trên các đối tượng. Nếu trong C/C++ ta có thể tạo ra các hàm (chương trình con không gắn với đối tượng nào) thì trong java ta chỉ có thể tạo ra các phương thức (chương trình con gắn liền với một lớp cụ thể). Trong Java không cho phép các đối tượng có tính năng đa kế thừa mà được thay thế bằng các giao diện (interface).

1. Độc lập phần cứng và hệ điều hành

Đối với các ngôn ngữ lập trình truyền thống C/C++, phương pháp biên dịch được như sau:

* Với mỗi nền phần cứng khác nhau, có một trình biên dịch khác nhau để biên dịch mã nguồn chương trình cho phù hợp với nền phần cứng ấy. Do vậy, khi chạy trên một nền phần cứng khác bắt buộc phải biên dịch lại mã nguồn.
* Đối với các chương trình viết bằng Java, trình biên dich Javac sẽ biên dịch mã nguồn thành bytecode. Sau đó, khi chạy chương trình trên các nền phần cứng khác nhau, máy ảo Java dùng trình thông dịch Java dùng trình thông dịch Java để chuyển mã bytecode thành dạng chạy được trên các nền phần cứng tương ứng. Do vậy, khi thay đổi nền phần cứng, không phải biên dịch lại mã nguồn Java.

1. Mạnh mẽ

Java là ngôn ngữ yêu cầu chặt chẽ về kiểu dữ liệu.

* Kiểu dữ liệu phải khai báo tường minh.
* Java không sử dụng con trỏ và các phép toán con trỏ.
* Java kiểm tra tất cả các truy nhập đến mảng, chuỗi khi thực thi để đảm bảo rằng các truy nhập đó không ra ngoài giới hạn kích thước.

Trong các môi trường lập trình truyền thống, lập trình viên phải tự mình cấp phát bộ nhớ, trước khi chương trình kết thúc thì phải tự giải phóng bộ nhớ đã xin cấp trước đó. Trong chương trình Java, lập trình viên không phải bận tâm đến việc cấp phát bộ nhớ. Quá trình cấp phát, giải phóng được thực hiện tự động, nhờ dịch vụ thu nhặt những đối tượng không còn sử dụng nữa (garbage collection).

Cơ chế bẫy lỗi của Java giúp đơn giản hóa quá trình xử lý lỗi và hồi phục sau lỗi.

1. Bảo mật

Java cung cấp một môi trường quản lý thực thi chương trình với nhiều mức để kiểm soát tính an toàn:

ở mức thứ nhất, dữ liệu và các phương thức được đóng gói bên trong lớp. Chúng chỉ được truy xuất thông qua các giao diện mà lớp cung cấp.

ở múc thứ hai, trình biên dịch kiểm soát để đảm bảo mã là an toàn, và tuân theo các nguyên tắc của java

Mức thứ ba được đảm bảo bởi trình thông dịch; chúng kiểm soát xem bytecode có đảm bảo các quy tắc an toàn trước khi thực thi không.

Mức thứ tư kiểm soát việc nạp các lớp vào bộ nhớ để giám sát việc vi phạm giới hạn truy xuất trước khi nạp vào hệ thống.

1. Phân tán

Java được thiết kế để hỗ trợ các ứng dụng chạy trên mạng bằng các lớp mạng (java.net). Hơn nữa, Java hỗ trợ nhiều nền chạy khác nhau nên chúng được sử dụng rộng rãi như là công cụ phát trên Internet – nơi sử dụng nhiều nền khác nhau.

1. Đa luồng

Chương trình Java cung cấp giải pháp đa luồng (Multithreading) để thực thi các công việc đồng thời. Chúng cũng cung cấp giải pháp đồng bộ giữa các luồng. Đặc tình hỗ trợ đa luồng này cho phép xây dựng trên mạng chạy hiệu quả.

1. Linh động

Java được thiết kế như một ngôn ngữ động để đáp ứng cho những môi trường mở. Các chương trình Java chữa rất nhiều thông tin nhằm kiểm soát và truy nhập đối tượng lúc chạy. Điều này cho phép khả năng liên kết mã động.

1. **Tìm hiểu về Desgin Pattern**
2. Design Pattern là gì ?

* Trong kỹ thuật phần mềm, design pattern (mẫu thiết kế) là một giải pháp lặp lại nói chung cho một vấn đề thường xảy ra trong thiết kế phần mềm.
* Một mẫu thiết kế không phải là một ngôn ngữ hay thiết kế hoàn chỉnh có thể được chuyển trực tiếp thành mã. Nó là một mô tả hoặc mẫu cho cách giải quyết vấn đề có thể được sử dụng trong nhiều tình huống khác nhau.
* Design pattern rất phổ biến trong phát triển phần mềm, nó được áp dụng sẵn trong các Framework, các thư viện

1. Lợi ích của Design Pattern

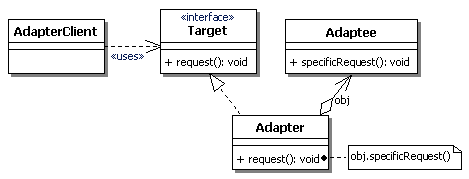
Một số lợi ích của việc sử dụng các mẫu thiết kế là:

* Các mẫu thiết kế đã được xác định và cung cấp cách tiếp cận chuẩn theo ngành để giải quyết vấn đề lặp lại, vì vậy sẽ tiết kiệm được thời gian nếu chúng ta sử dụng mẫu thiết kế hợp lý.
* Sử dụng các mẫu thiết kế thúc đẩy khả năng tái sử dụng dẫn đến phát triển phần mềm nhanh hơn. Nó giúp giảm tổng chi phí của sản phẩm phần mềm.
* Khi áp dụng 1 mẫu thiết kế, nó làm cho code dễ hiểu và gỡ lỗi.

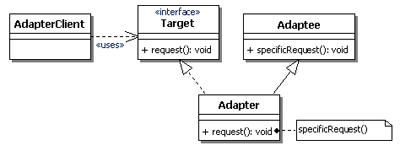
1. Phân loại Design Pattern

Có 3 nhóm chính sau:

* **Creational Pattern** (nhóm khởi tạo) gồm: Abstract Factory, Factory Method, Singleton, Builder, Prototype. Nó sẽ giúp bạn trong việc khởi tạo đối tượng, như bạn biết để khởi tạo bạn phải sử dụng từ khóa new, nhóm **Creational Pattern** sẽ sử dụng một số thủ thuật để khởi tạo đối tượng mà bạn sẽ không nhìn thấy từ khóa này.
* **Structural Pattern** (nhóm cấu trúc) gồm: Adapter, Bridge, Composite, Decorator, Facade, Proxy và Flyweight. Nó dùng để thiết lập, định nghĩa quan hệ giữa các đối tượng.
* **Behavioral Pattern** gồm: Interpreter, Template Method, Chain of Responsibility, Command, Iterator, Mediator, Memento, Observer, State, Strategy và Visitor. Nhóm này dùng trong thực hiện các hành vi của đối tượng.
* Vậy *Adapter Pattern* là pattern giữ vai trò trung gian giữa hai lớp, chuyển đổi giao diện của một hay nhiều lớp có sẵn thành một giao diện khác, thích hợp cho lớp đang viết. Điều này cho phép các lớp có các giao diện khác nhau có thể dễ dàng giao tiếp tốt với nhau thông qua giao diện trung gian, không cần thay đổi code của lớp có sẵn cũng như lớp đang viết. Adapter Pattern còn gọi là Wrapper Pattern do cung cấp một giao diện “bọc ngoài” tương thích cho một hệ thống có sẵn, có dữ liệu và hành vi phù hợp nhưng có giao diện không tương thích với lớp đang viết.



Hình 1. Object Adapter Pattern



Hình 2. Class Adapter Pattern

1. **Tìm hiểu về android**

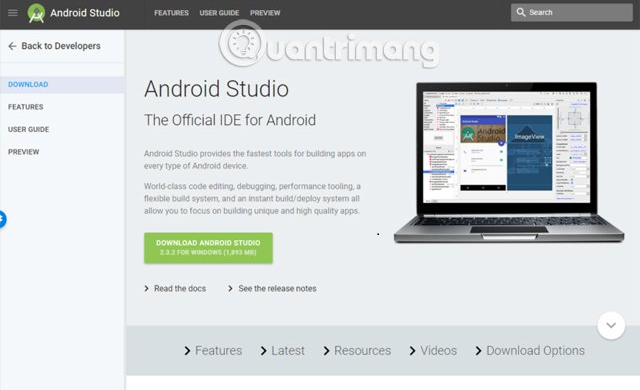
Android là gì?

Có nhiều công cụ để phát triển Android nhưng đến nay công cụ chính thức và mạnh mẽ nhất là Android Studio. Đây là IDE (Môi trường phát triển tích hợp) chính thức cho nền tảng Android, được phát triển bởi Google và được sử dụng để tạo phần lớn các ứng dụng mà bạn có thể sử dụng hàng ngày.

Chức năng của Android Studio là cung cấp giao diện để tạo các ứng dụng và xử lý phần lớn các công cụ quản lý file phức tạp đằng sau hậu trường. Ngôn ngữ lập trình được sử dụng ở đây là [Java](https://quantrimang.com/java-la-gi-tai-sao-nen-chon-java-143130) và được cài đặt riêng trên thiết bị của bạn. Android Studio rất đơn giản, bạn chỉ cần viết, chỉnh sửa và lưu các dự án của mình và các file trong dự án đó. Đồng thời, Android Studio sẽ cấp quyền truy cập vào Android SDK.

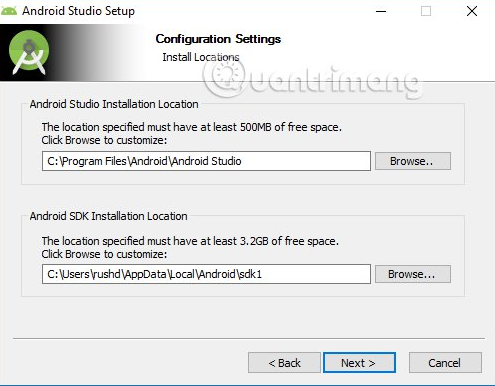
Cài đặt Android Studio

Thiết lập Android Studio khá đơn giản và dễ dàng hơn bao giờ hết nhờ vào trình cài đặt của nó. Khi tải [Android Studio](https://download.com.vn/android-studio/download), ngoài Android Studio, bạn sẽ nhận được Android SDK, SDK manager và nhiều công cụ đi kèm khác. Công cụ duy nhất bạn cần là Java Development Kit.



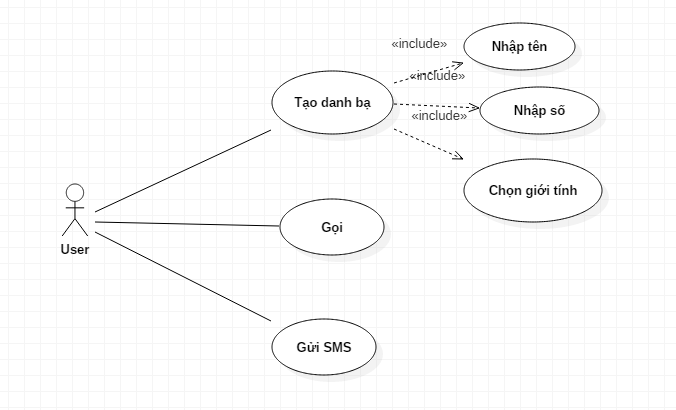
Hình 2: Web tải Android Studio

Thực hiện theo các hướng dẫn đơn giản trong khi cài đặt và nó cũng thiết lập một nền tảng Android mà bạn có thể phát triển tốt nhất.



Hình 3: Cài đặt Android Studio

1. **Phân tích hệ thống**
2. Sơ đồ Use Case

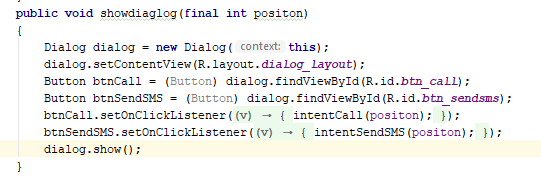


Hình 5: Sơ đồ use case

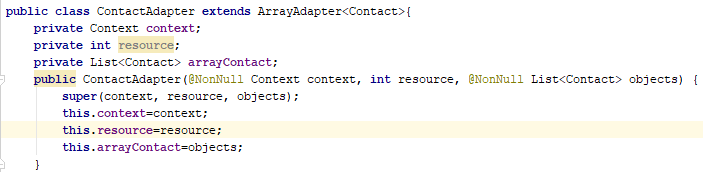
* Người sử dụng phần mềm sẽ tạo danh bạ bằng cách nhập tên, số điện thoại, chọn giới tính (để hiển thị avatar) và nhấn tạo danh bạ.
* Sau khi tạo thành công, người dùng có thể thực hiện gọi và nhắn tin bằng cách nhấn giữ trên đối tượng cần thực hiện.

1. Các giải thuật

* Chương trình sẽ dựa vào position mà người dùng chọn để lọc ra số điện thoại thực hiện hành động



* Mô hình Adapter Pattern



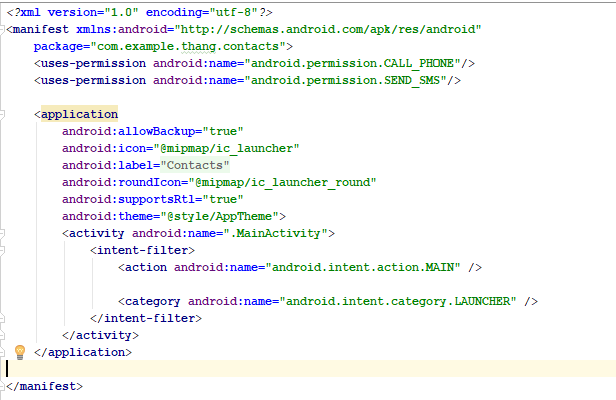
Trong MainActivity

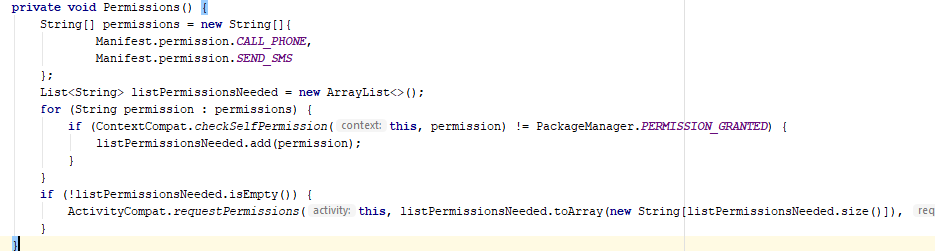


1. Phân quyền trong Android

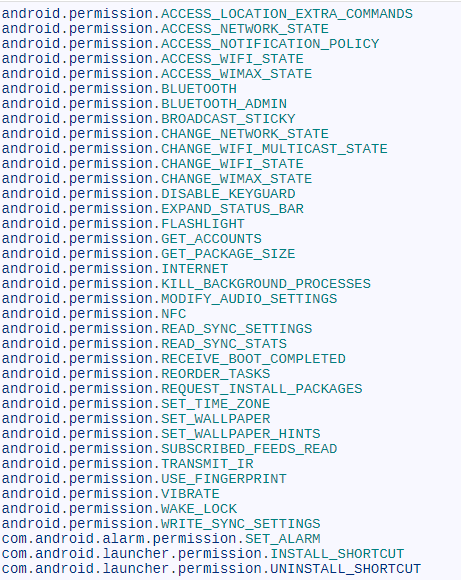
* Đối với các phiên bản từ Android 5.1 (Lollipop) trở về trước thì phân quyền là không cần thiết và đã bị Google làm ẩn đi
* Đối với các phiên bản từ Android 6 (Marshmallow) cho phép bạn kiểm soát quyền truy cập (Permission) của từng app một. Người dùng có thể kiểm soát thiết bị của bản thân một cách tốt nhất.

Để khai báo ứng dụng của bạn cần 1 quyền, đặt 1 thẻ <use - permission> trong file Manifest, như là phần tử con của thẻ gốc <manifest>



Vì gọi và gửi SMS được xem là quyền nguy hiểm nên ta phải xin cấp quyền một lần nữa ở MainActivity

* Các quyền cơ bản của Android:



* Các quyền nguy hiểm:

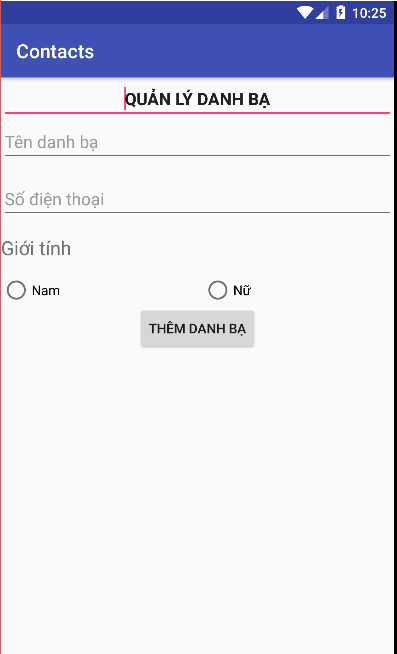


1. **Kết luận và hướng phát triển**

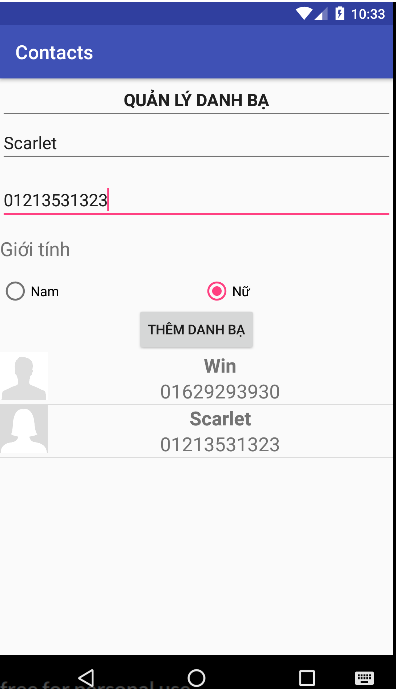
Xây dựng được chương trình quản lý danh bạ cơ bản giúp người dùng Android quản lý danh bạ thuận tiện hơn

1. Kết quả đạt được:

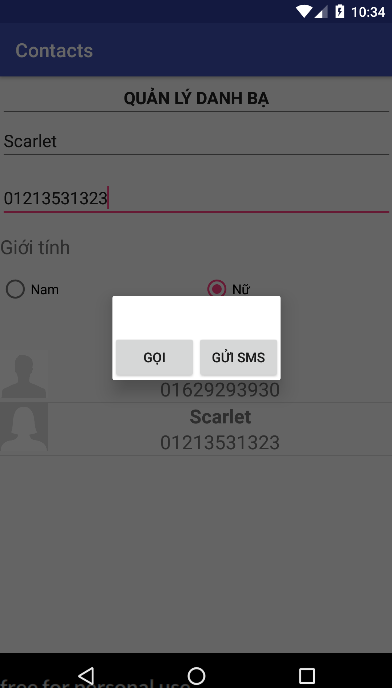
Xây dựng chương trình cụ thể



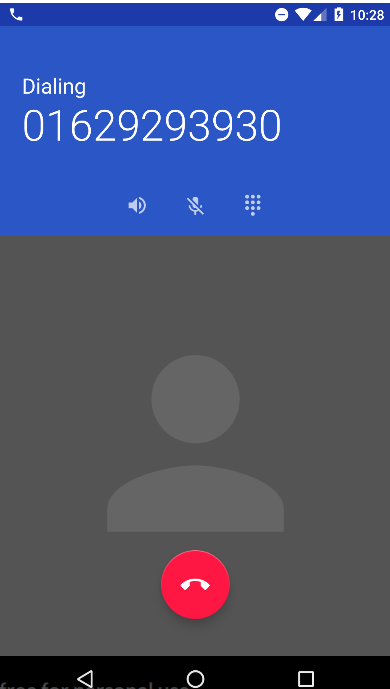
Hình 4: Giao diện chính



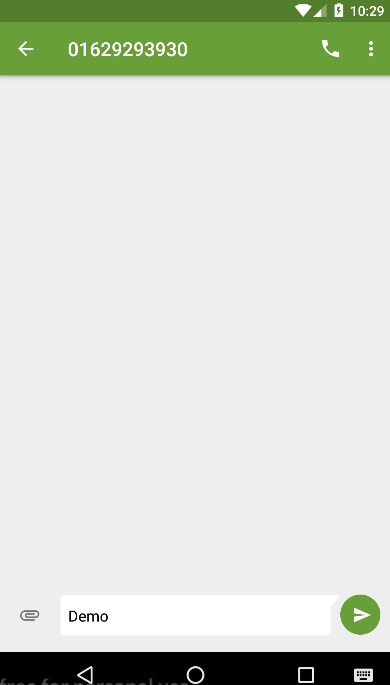
Hình 5: Thêm thành công số liên lạc



Hình 6: Showdialog tùy chọn



Hình 7: Màn hình thực hiện cuộc gọi



Hình 8: Màn hình gửi SMS

1. Hạn chế:

* Chưa cài đặt thử nghiệm trên máy thật để thực hiện các tính năng

1. **Tài liệu kham khảo:**
2. Youtube dạy lập trình hướng Android:

<https://www.youtube.com/watch?v=n1iOKgVhMoM&list=PLzrVYRai0riTlWPxOEhi1-2QmvLiw0DCb>

1. Design Pattern in Java Tutorial

<https://www.tutorialspoint.com/design_pattern/index.htm>