Assignment Number : E-IV C-1

February 15, 2016

Problem Statement

Write a Mobile App program using J2ME /Python /Scala /Java /Android to check the palindrome in a given string.

Objective

- To study installation and working of Android Studio.
- To understand how to create Mobile Application for Palindrome program.

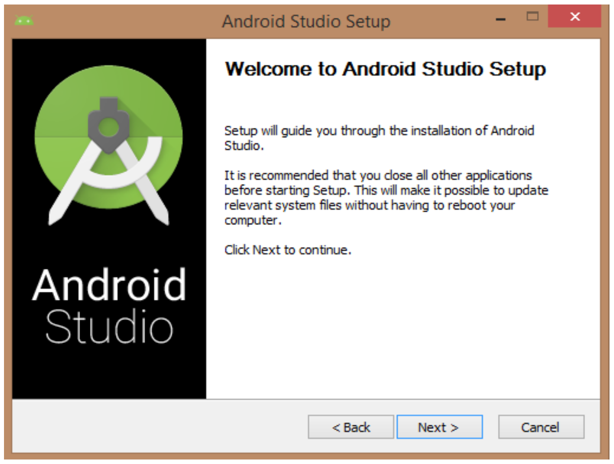
Theory

Android Studio

Android Studio is the official IDE for android application development.It works based on IntelliJ IDEA

Installation

- Launch Android Studio.exe. Make sure before launching Android Studio, Machine should have Java JDK installed.



Welcome to Android Studio Setup

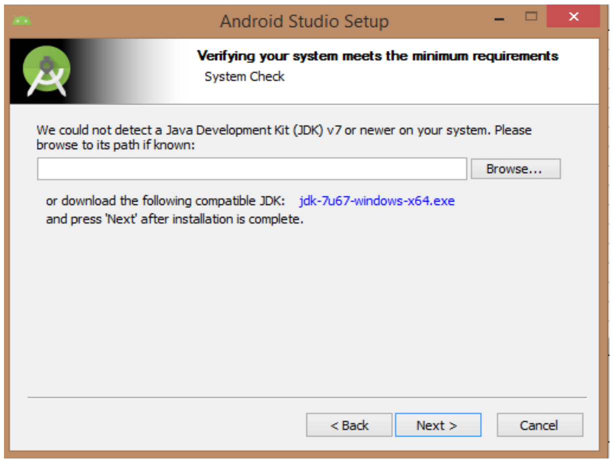
Setup will guide you through the installation of Android Studio.

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Figure 1: Step 1

- Once you launched Android Studio, its time to mention JDK5 path or later version in android studio installer.



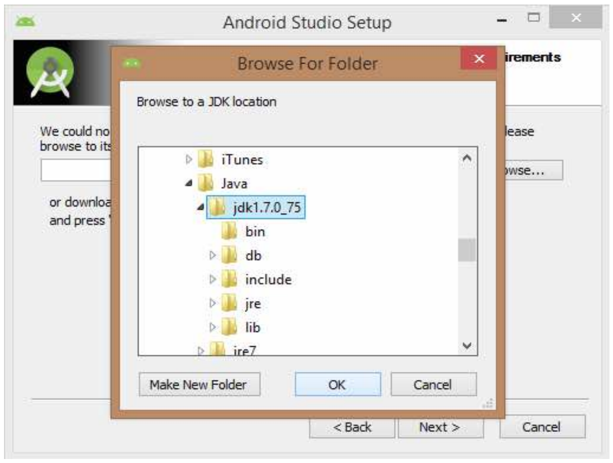
Verifying your system meets the minimum requirements
System Check

We could not detect a Java Development Kit (JDK) v7 or newer on your system. Please browse to its path if known:

or download the following compatible JDK: jdk-7u67-windows-x64.exe
and press 'Next' after installation is complete.

Figure 2: Step 2

- Below the image initiating JDK to android SDK

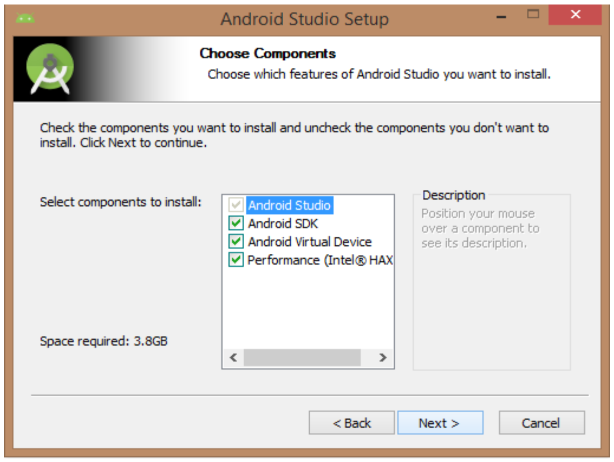


Browse to a JDK location

- iTunes
- Java
  - jdk1.7.0_75
    - bin
    - db
    - include
    - jre
    - lib
  - jre7

Figure 3: Step 3

- Need to check the components, which are required to create applications, below the image has selected Android Studio,Android SDK,Android Virtual Machine and performance(Intel chip).



Choose Components

Choose which features of Android Studio you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

- [✓] Android Studio
- [✓] Android SDK
- [✓] Android Virtual Device
- [✓] Performance (Intel® HAX

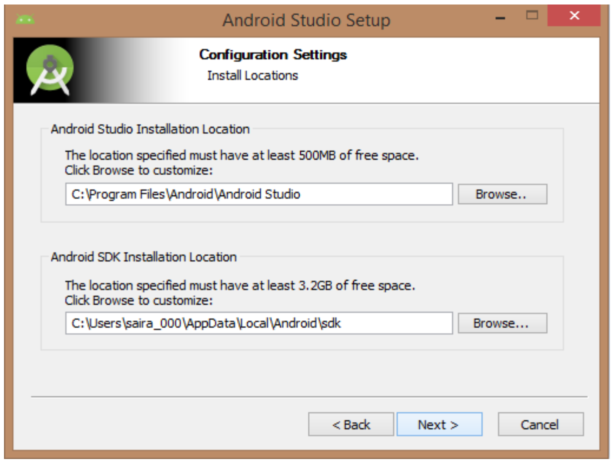
Description

Position your mouse over a component to see its description.

Space required: 3.8GB

Figure 4: Step 4

- Need to specify the location of local machine path for Android studio and Android SDK, below the image has taken default location of windows 8.1 x64 bit architecture.


Android Studio Setup

Configuration Settings
Install Locations

Android Studio Installation Location

The location specified must have at least 500MB of free space.
Click Browse to customize:

C:\Program Files\Android\Android Studio

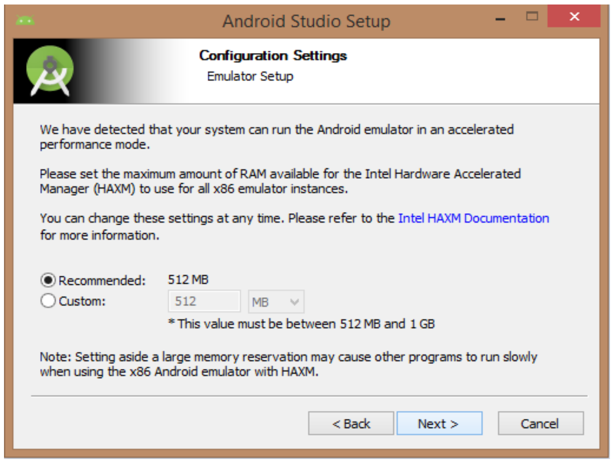
Android SDK Installation Location

The location specified must have at least 3.2GB of free space.
Click Browse to customize:

C:\Users\saira_000\AppData\Local\Android\sdk

Figure 5: Step 5

- Need to specify the ram space for Android emulator by default it would take 512MB of local machine RAM


Android Studio Setup

Configuration Settings
Emulator Setup

We have detected that your system can run the Android emulator in an accelerated performance mode.

Please set the maximum amount of RAM available for the Intel Hardware Accelerated Manager (HAXM) to use for all x86 emulator instances.

You can change these settings at any time. Please refer to the Intel HAXM Documentation for more information.

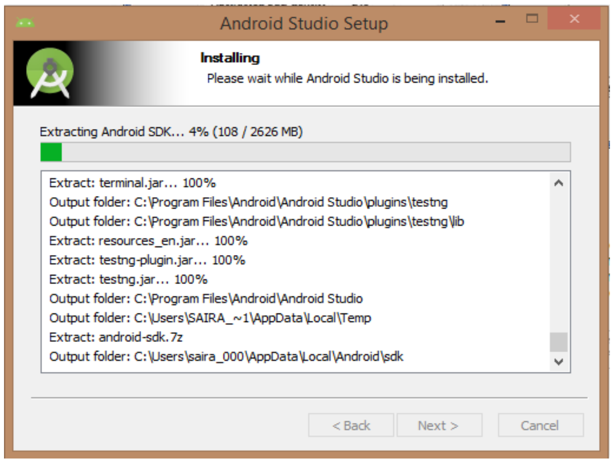
[✓ Recommended:] 512 MB
Custom: 512 MB

* This value must be between 512 MB and 1 GB

Note: Setting aside a large memory reservation may cause other programs to run slowly when using the x86 Android emulator with HAXM.

Figure 6: Step 6

- At final stage, it would extract SDK packages into our local machine, it would take a while time to finish the task and would take 2626MB of Hard disk space.



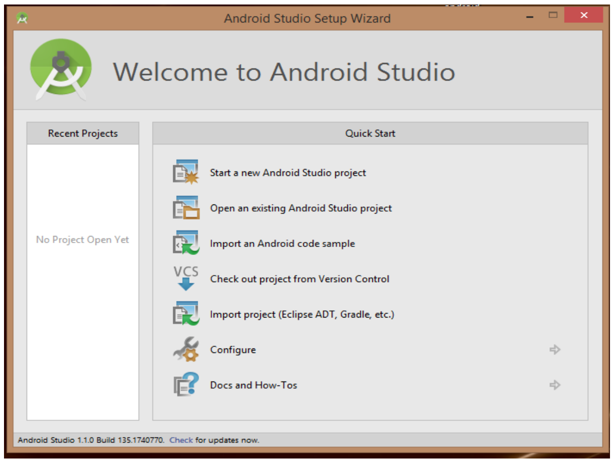
Please wait while Android Studio is being installed.

Extracting Android SDK... 4% (108 / 2626 MB)

```
Extract: terminal.jar... 100%
Output folder: C:\Program Files\Android\Android Studio\plugins\testng
Output folder: C:\Program Files\Android\Android Studio\plugins\testng\lib
Extract: resources_en.jar... 100%
Extract: testng-plugin.jar... 100%
Extract: testng.jar... 100%
Output folder: C:\Program Files\Android\Android Studio
Output folder: C:\Users\SAIRA_~1\AppData\Local\Temp
Extract: android-sdk.7z
Output folder: C:\Users\saira_000\AppData\Local\Android\sdk
```

Figure 7: Step 7

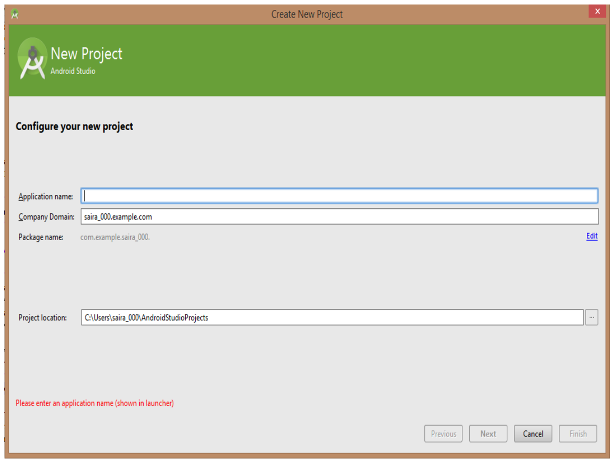
- After done all above steps perfectly, open android studio project with Welcome to android studio message as shown below


Android Studio Setup Wizard
Welcome to Android Studio
Recent Projects
No Project Open Yet
Quick Start
Start a new Android Studio project
Open an existing Android Studio project
Import an Android code sample
Check out project from Version Control
Import project (Eclipse ADT, Gradle, etc.)
Configure
Docs and How-Tos

Android Studio 1.1.0 Build 135.1740770. Check for updates now.

Figure 8: Step 8

- Start your application development by calling start a new android studio project. in a new installation frame should ask Application name, package information and location of the project.



Configure your new project

Application name:

Company Domain: saira_000.example.com

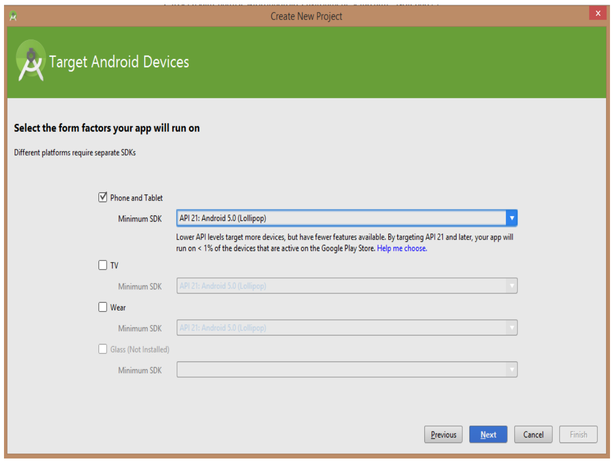
Package name: com.example.saira_000.

Project location: C:\Users\saira_000\AndroidStudioProjects

Please enter an application name (shown in launcher)

Figure 9: Step 9

- After entering application name, select the form factors your application runs on, here need to specify Minimum SDK.



Select the form factors your app will run on

Different platforms require separate SDKs

[✓] Phone and Tablet

Minimum SDK: API 21: Android 5.0 (Lollipop)

Lower API levels target more devices, but have fewer features available. By targeting API 21 and later, your app will run on < 1% of the devices that are active on the Google Play Store. Help me choose.

[ ] TV

Minimum SDK: API 21: Android 5.0 (Lollipop)

[ ] Wear

Minimum SDK: API 21: Android 5.0 (Lollipop)

[ ] Glass (Not Installed)

Minimum SDK:

Figure 10: Step 10

- The next level of installation should contain selecting the activity to mobile, it specifies the default layout for Applications

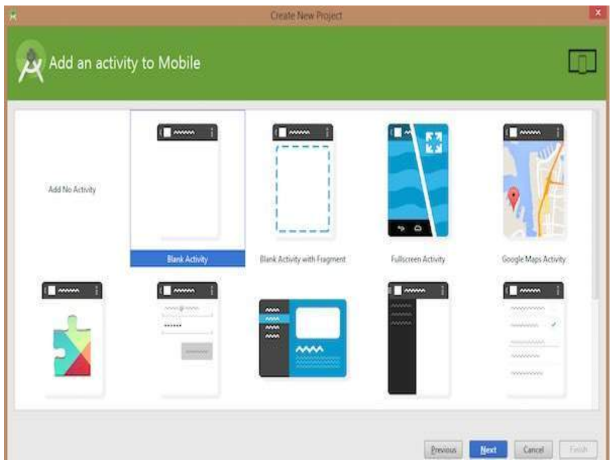


Figure 11: Step 11

- At the final stage it going to be open development tool to write the application code.

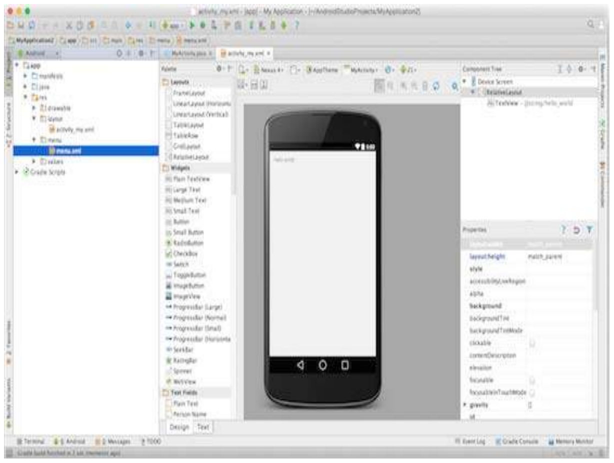
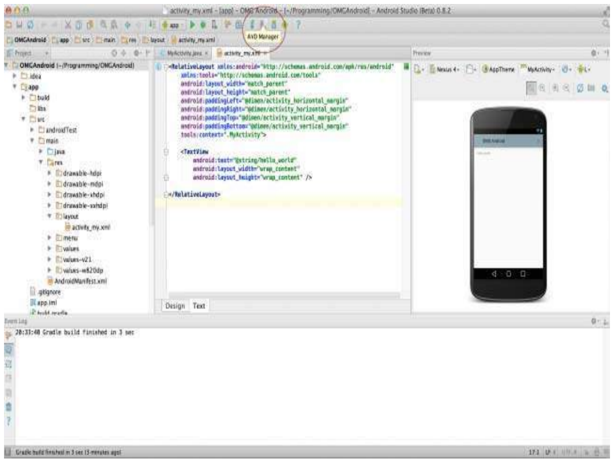


Figure 12: Step 12

Create Android Virtual Device

- To test your Android applications, you will need a virtual Android device. So before we start writing our code, let us create an Android virtual device. Launch Android AVD Manager Clicking $AVD\_Manager$ icon as shown below

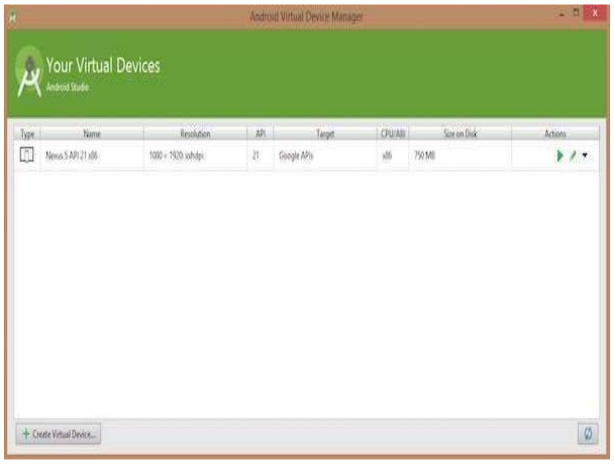


```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MyActivity">

    <TextView
        android:text="@string/hello_world"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</RelativeLayout>
```

```
20:33:48 Gradle build finished in 3 sec
```

Figure 13: Step 13

- After Click on a virtual device icon, it going to be shown by default virtual devices which are present on your SDK, or else need to create a virtual device by clicking Create new Virtual device button



Type	Name	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Nexus 5 API 21 x86	1080 × 1920: xxhdpi	21	Google APIs	x86	750 MB	

Figure 14: Step 14

- If your AVD is created successfully it means your environment is ready for Android application development.

Application Components

Application components are the essential building blocks of an Android application. These components are loosely coupled by the application manifest file AndroidManifest.xml that describes each component of the application and how they interact. There are following four main components that can be used within an Android application:

1. **Activities:**
   An activity represents a single screen with a user interface. For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. If an application has more than one activity, then one of them should be marked as the activity that is presented when the application is launched.
   An activity is implemented as a subclass of Activity class as follows:

```
public class MainActivity extends Activity { }
```

2. **Services:**
   A service is a component that runs in the background to perform long-running operations. For example, a service might play music in the background while the user is in a different application, or it might fetch data over the network without blocking user interaction with an activity.
   A service is implemented as a subclass of Service class as follows:

   ```
   public class MyService extends Service { }
   ```

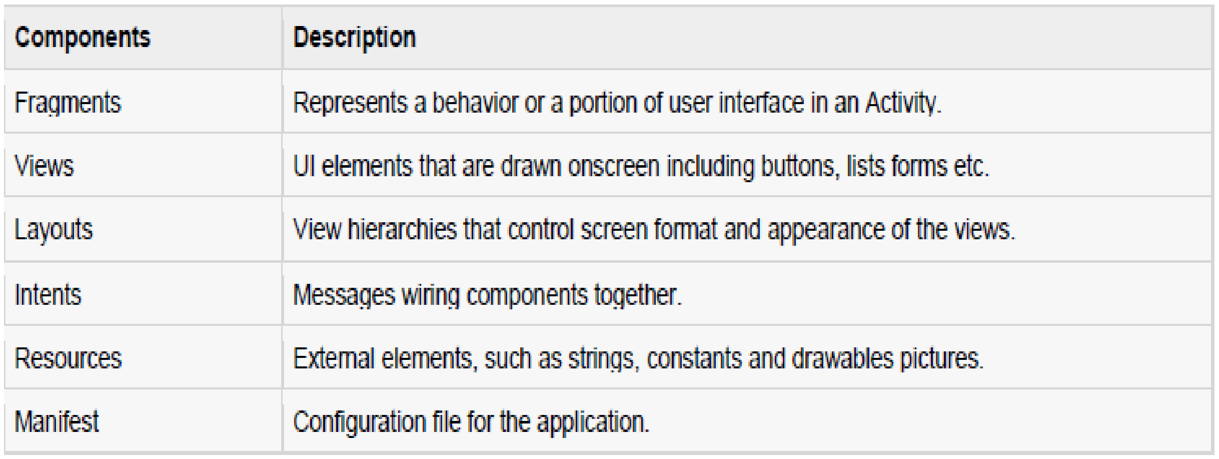
3. **Broadcast Receivers:**
   Broadcast Receivers simply respond to broadcast messages from other applications or from the system. For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use, so this is broadcast receiver who will intercept this communication and will initiate appropriate action.
   A broadcast receiver is implemented as a subclass of BroadcastReceiver class and each message is broadcasted as an Intent object.

   ```
   public class MyReceiver extends BroadcastReceiver { }
   ```

4. **Content Providers:**
   A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the ContentResolver class. The data may be stored in the file system, the database or somewhere else entirely.
   A content provider is implemented as a subclass of ContentProvider class and must implement a standard set of APIs that enable other applications to perform transactions

   ```
   public class MyContentProvider extends ContentProvider { }
   ```

There are additional components which will be used in the construction of above mentioned entities, their logic, and wiring between them.
These components are:



Components	Description
Fragments	Represents a behavior or a portion of user interface in an Activity.
Views	UI elements that are drawn onscreen including buttons, lists forms etc.
Layouts	View hierarchies that control screen format and appearance of the views.
Intents	Messages wiring components together.
Resources	External elements, such as strings, constants and drawables pictures.
Manifest	Configuration file for the application.

Figure 15: Additional Components

Important Application Files

- **The Main Activity File:**
  The main activity code is a Java file MainActivity.java. This is the actual application file which ultimately gets converted to a Dalvik executable and runs your application.

- **The Manifest File:**
  Whatever component are developed as a part of application, all its components must be declared in a manifest file called AndroidManifest.xml which resides at the root of the application project directory. This file works as an interface between Android OS and the application.

- **The Strings File:**
  The strings.xml file is located in the res/values folder and it contains all the text that the application uses. For example, the names of buttons, labels, default text, and similar types of strings go into this file. This file is responsible for their textual content.

- **The R File:**
  The R.java file is the glue between the activity Java files like MainActivity.java and the resources like strings.xml. It is an automatically generated file and one should not modify the content of the R.java file.

- **The Layout File:**
  The *activity_main.xml* is a layout file available in res/layout directory, that is referenced by the application when building its interface.

Building Android Application

Event Handling

Events are a useful way to collect data about a user's interaction with interactive components of your app, like button presses or screen touch etc. The Android framework maintains an event queue into which events are placed as they occur and then each event is removed from the queue on a first-in, first-out (FIFO) basis. One can capture these events in program and take appropriate action as per requirements.
There are following three concepts related to Android Event Management:

- Event Listeners:
  The View class is mainly involved in building up a Android GUI, same View class provides a number of Event Listeners. The Event Listener is the object that receives notification when an event happes.

- Event Listeners Registration:
  Event Registration is the process by which an Event Handler gets registered with an Event Listener so that the handler is called when the Event Listener fires the event.

- Event Handlers:
  When an event happens and have registered the event, the event listener calls the Event Handlers, which is the method that actually handles the event.
  Example:

  1. onClick():
     OnClickListener() is called when the user either clicks or touches or focuses upon any widget like button, text, image etc. It uses onClick() event handler to handle such event.

Toast

A toast provides simple feedback about an operation in a small popup. It only fills the amount of space required for the message and the current activity remains visible and interactive. For example, navigating away from an email before we send it triggers a "Draft saved" toast to let us know that we can continue editing later. Toasts automatically disappear after a timeout.

First, instantiate a Toast object with one of the makeText() methods. This method takes three parameters: the application Context, the text message, and the duration for the toast. It returns a properly initialized Toast object. One can display the toast notification with show(), as shown in the following example:

```
Context context = getApplicationContext();
CharSequence text = "Hello toast!";
int duration = Toast.LENGTH_SHORT;

Toast toast = Toast.makeText(context, text, duration);
toast.show();
```

Intent

An intent is an abstract description of an operation to be performed. It can be used with startActivity to launch an Activity, broadcastIntent to send it to any interested BroadcastReceiver components, and startService(Intent) or bindService(Intent, ServiceConnection, int) to communicate with a background Service.
An Intent provides a facility for performing late runtime binding between the code in different applications. Its most significant use is in the launching of activities, where it can be thought of as the glue between activities. It is basically a passive data structure holding an abstract description of an action to be performed.

Intent Structure:

The primary pieces of information in an intent are:

- **Action:**
  The general action to be performed, such as *ACTION_VIEW*, *ACTION_EDIT*, *ACTION_MAIN*, etc.

- **Data:**
  The data to operate on, such as a person record in the contacts database, expressed as a Uri.

Some examples of action/data pairs are:

- *ACTION_VIEW* content: //contacts/people/1
  Display information about the person whose identifier is "1".

- *ACTION_DIAL* content: //contacts/people/1
  Display the phone dialer with the person filled in.

In addition to these primary attributes, there are a number of secondary attributes that one can also include with an intent:

- **Category:**
  Gives additional information about the action to execute. For example, *CATEGORY_LAUNCHER* means it should appear in the Launcher as a top-level application, while *CATEGORY_ALTERNATIVE* means it should be included in a list of alternative actions the user can perform on a piece of data.

- **Type:**
  Specifies an explicit type (a MIME type) of the intent data. Normally the type is inferred from the data itself. By setting this attribute, you disable that evaluation and force an explicit type.

- **Component:**
  Specifies an explicit name of a component class to use for the intent. Normally this is determined by looking at the other information in the intent (the action, data/type, and categories) and matching that with a component that can handle it. If this attribute is set then none of the evaluation is performed, and this component is used exactly as is. By specifying this attribute, all of the other Intent attributes become optional.

- **Extras:**
  This is a Bundle of any additional information. This can be used to provide extended information to the component. For example, if we have a action to send an e-mail message, we could also include extra pieces of data here to supply a subject, body, etc.

Mathematical Model

Let S be the System that represents the Palindrome Application.
Initially,

$$S = \{\phi\}$$

Let,

$$S = \{I, O, F\}$$

Where:-

I	= Represents Input Set
O	= Represents Output Set.
F	= Represents Function set.

Input Set -I

String to be checked if it is Palindrome or not.

Output Set -O

Result, whether the given String is Palindrome or not.

Function Set -F

$$F = \{F1, F2\}$$

Where:-

F1	= Represents the onClick function for Check button.
F2	= Represents the MakeToast function for displaying result.

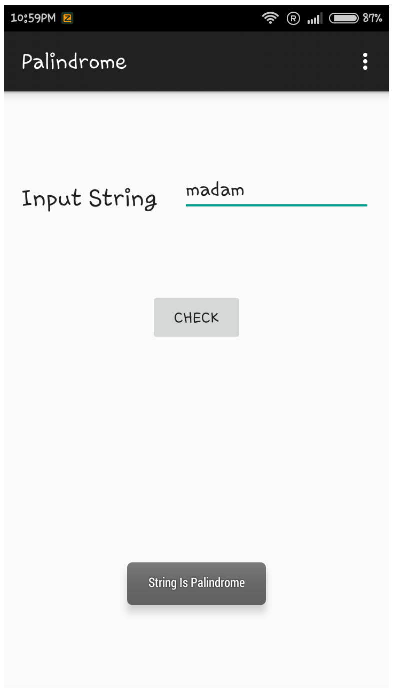
Finally,

$$S = \{I, O, F\}$$

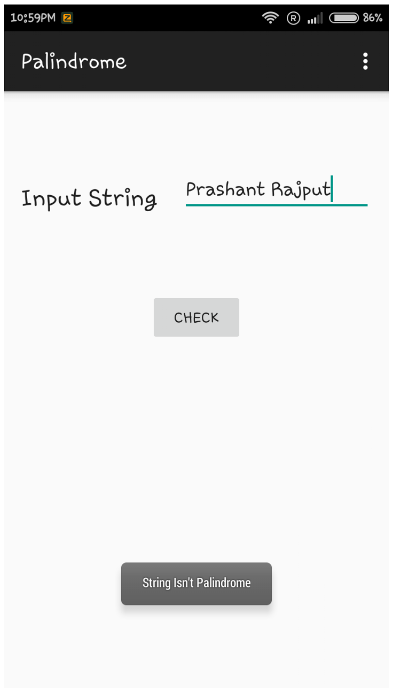
Conclusion

Thus, studied how to create Android Mobile Application to check whether the given string is a Palindrome or not.

Output:



(a) Output



(b) Output