Level/Subject	:	**D3/Programming 6**
Topic	:	Deploying Android apps
Week	:	14th
Activity	:	Preparing to deploy an Android apps with Android Studio
Alocated time	:	120 mins labs & 120 mins self study
Deliverables	:	Project folder
Due date	:	end of week

Competency :

Student expected to be able to deploy Android apps using Android Studio IDE.

Example Practice Task:

Prepare to deploy an Android apps using Android Studio.

1. Lets open again your final version of PhotoGallery apps.

Steps to take before releasing your app

2. Firstly, make sure you have completely tested your application. We recommend testing your application in the following ways:
   a. On a device using the minimum required platform
   b. On a device using the target platform
   c. On a device using the latest available platform
   d. On a real device and not just the emulator
   e. On a variety of screen resolutions and sizes
   f. On a tablet if your application supports it
   g. By switching to landscape mode if you can allow it, both in a mobile device and in a tablet
   h. On different network conditions, such as with no Internet connectivity or low coverage
   i. When the GPS or any location service is not activated on your device (if your application uses GPS or any location service)
   j. When the Back button is pressed
3. Secondly, we have to check the log messages that are printed from our application. Printing some log messages can be considered a security vulnerability. Logs generated by the Android system can be captured and analyzed, so we should avoid showing critical information about the application's internal working. You should also remove the android:debuggable property from the application manifest file. You can also set this property to false.
4. Thirdly, if your application communicates with a server, check whether the configured URL is the production URL. It is possible that during the debug phase, you referenced to a URL of a server in a prerelease environment.

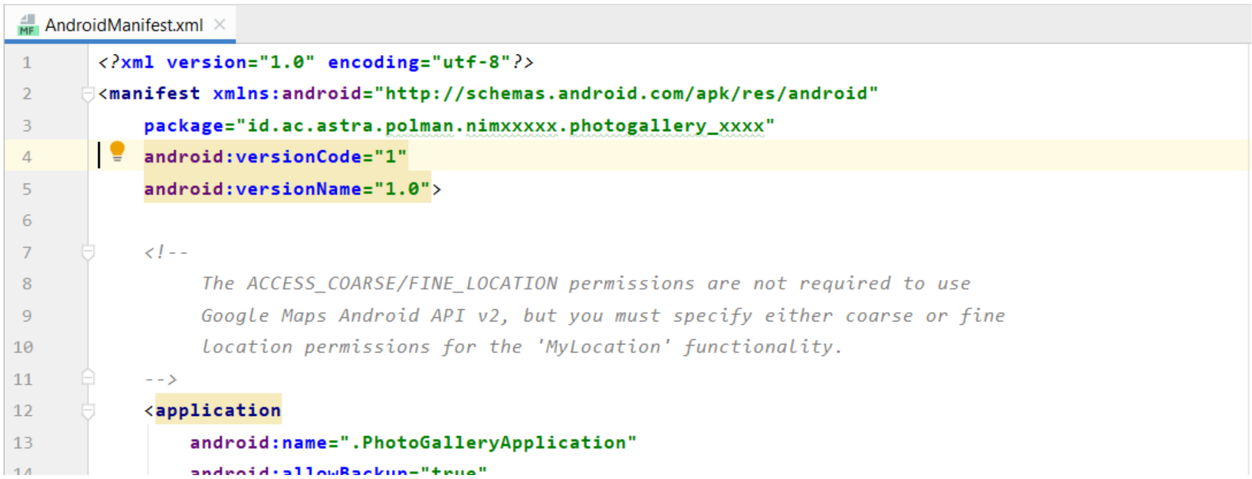
5. Finally, set the correct value for the android:versionCode and android:versionName properties from the application manifest file. The version code is a number (integer) that represents the application version. New versions should have greater version codes. This code is used to determine whether an application installed on a device is of the latest version, or if there is a newer version.
6. The version name is a string that represents the application version. Unlike the version code, the version name is visible to the user and appears in the public information about the application. It is just an informative version name to the user and is not used for any internal purpose. Specify a value of 1 for the version code and 1.0 for the version name. The manifest tag should look like the following:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapplication"
    android:versionCode="1"
    android:versionName="1.0" >
```

7. A new version of our application will have a value of 2 for the version code, and it could have 1.1 for the version name:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapplication"
    android:versionCode="2"
    android:versionName="1.1" >
```

8. Lets prepare our manifest like this:

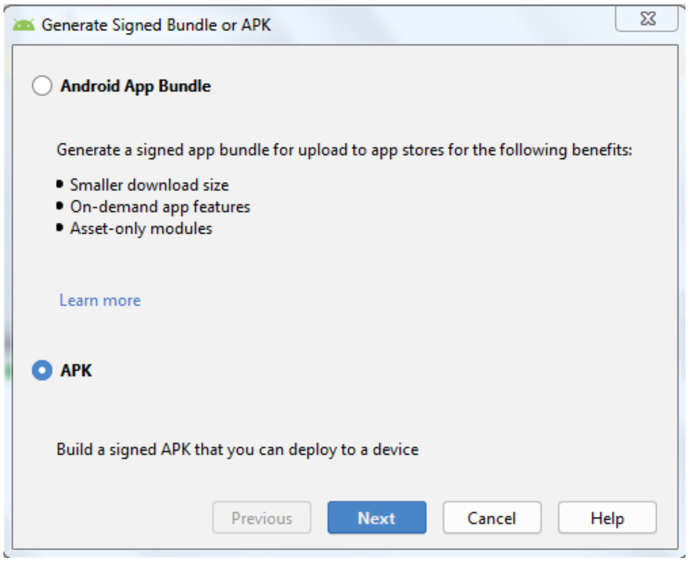


```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="id.ac.astra.polman.nimxxxxx.photogallery_xxxx"
    android:versionCode="1"
    android:versionName="1.0">

    <!--
         The ACCESS_COARSE/FINE_LOCATION permissions are not required to use
         Google Maps Android API v2, but you must specify either coarse or fine
         location permissions for the 'MyLocation' functionality.
    -->
    <application
        android:name=".PhotoGalleryApplication"
        android:allowBackup="true"
```

Generating a signed APK

9. To generate the signed APK, navigate to **Build** | **Generate Signed APK**.



Generate Signed Bundle or APK

○ **Android App Bundle**

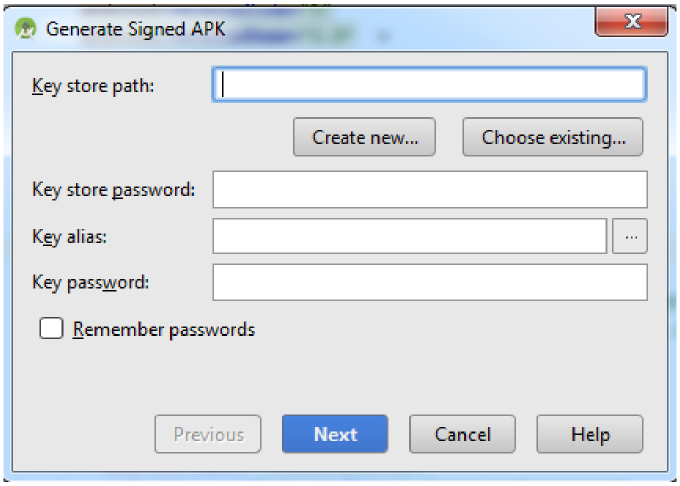
Generate a signed app bundle for upload to app stores for the following benefits:

- Smaller download size
- On-demand app features
- Asset-only modules

Learn more

● **APK**

Build a signed APK that you can deploy to a device



Generate Signed APK

Key store path:

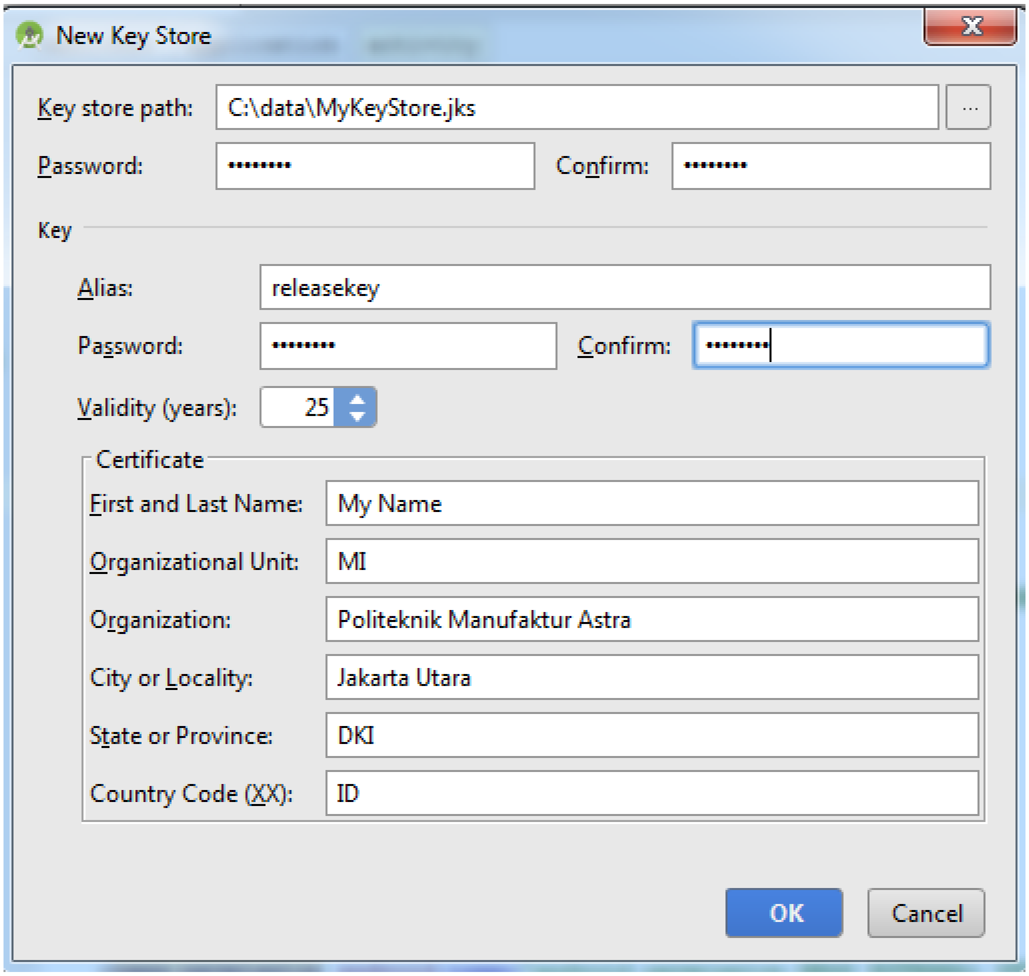
Key store password:

Key alias:

Key password:

□ Remember passwords

10. If this is our first application, we might not have any certificates.
11. Click on the **Create new** button to open the dialog to create a new key store.
12. Now fill in the following information:
    a. **Key store path**: This is the path in your system to create the key store. The key store is a file with the .jks extension, for example, release_keystore.jks.
    b. **Password**: This is the key store password. You have to confirm it.
    c. **Alias**: This is the alias for your certificate and is a pair of public and private keys. Let's name it releasekey.
    d. **Password**: This is the certificate password. You have to confirm it.
    e. **Validity (years)**: This is the certificate that will be valid until the validity date. A value of 25 years or more is recommended.
    f. **Certificate**: This is the personal information contained in the certificate. Type your first and last name, organizational unit, organization, city or locality, state or province, and country code;



New Key Store

Key store path: C:\data\MyKeyStore.jks

Password: •••••••• Confirm: ••••••••

Key

Alias: releasekey

Password: •••••••• Confirm: ••••••••

Validity (years): 25

Certificate

First and Last Name: My Name

Organizational Unit: MI

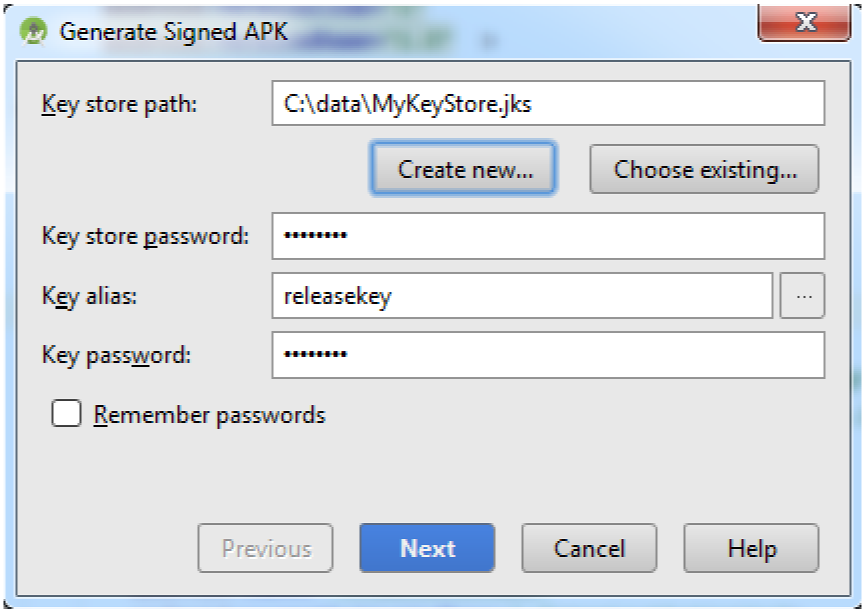
Organization: Politeknik Manufaktur Astra

City or Locality: Jakarta Utara

State or Province: DKI

Country Code (XX): ID

13. Click on **OK**.
14. The dialog to create the signed APK is now loaded with the key store data.



Generate Signed APK

Key store path: C:\data\MyKeyStore.jks

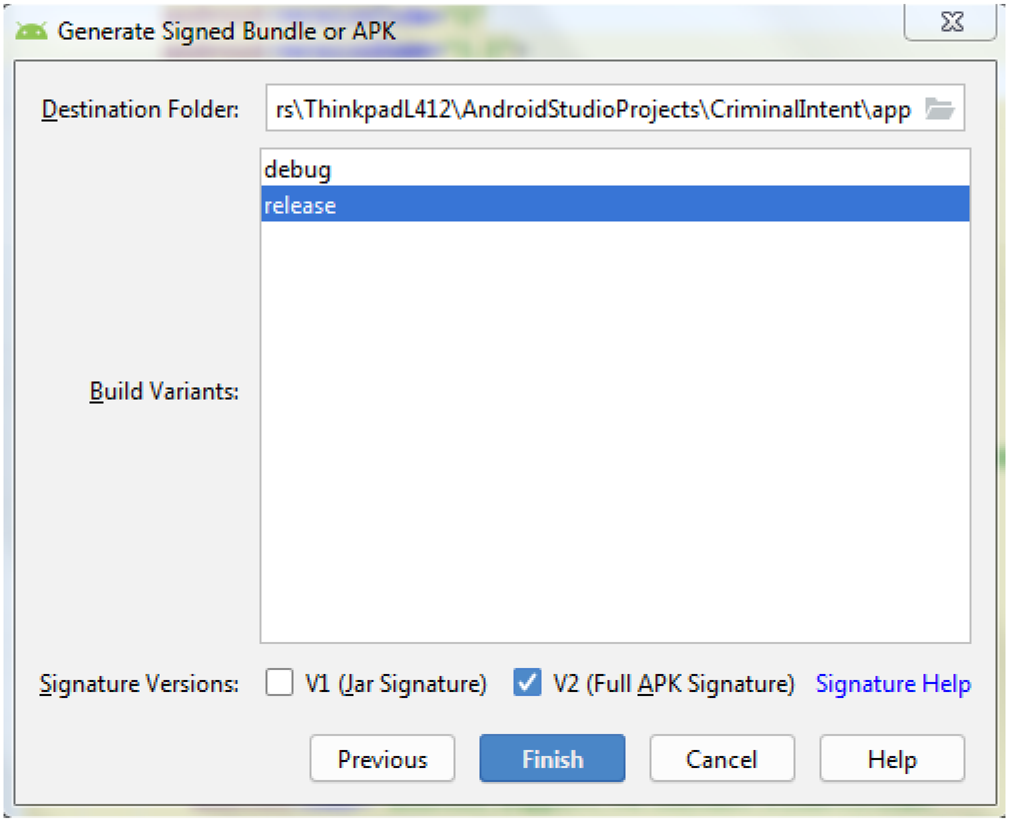
Key store password: ••••••••

Key alias: releasekey

Key password: ••••••••

☐ Remember passwords

15. The next time we create a signed APK, we will already have a certificate, so we will select the **Choose existing** button.
16. Click on the **Next** button.
17. In the next step, select the path to save the APK file, select the release build type, and click on **Finish**.



Generate Signed Bundle or APK

Destination Folder: rs\ThinkpadL412\AndroidStudioProjects\CriminalIntent\app

Build Variants: debug, release (selected)

Signature Versions: □ V1 (Jar Signature) ☑ V2 (Full APK Signature)

18. When the APK is completely generated, we will be informed by a message on the bottom bar of Android Studio. We should have the APK file created in the selected path.
19. Now that you have the APK file ready for release, it is recommended that you test it again in a device before distributing it.
20. Copy the apk to a device of your friend next to you. Try to install it and see if it works as intended.

Task : Deploying Android Apps on Google Play

1. Create a paper of howto in a docx file (MI_PRG6_M14_P1_NIM.docx) about the step to publish an android apps on Google Playstore.
2. Meanwhile, you should also upload your apk to a hosting on the internet (like dropbox) so your apps could be downloaded by the user.

3. Put that URL of your apk on the docx
4. Ask another friend of yours to try to download the apk and install it on his/her device.

Notes:

1. Create folder PRG6_M14_P1_XXXXXXXXXX.
2. Put all your work in that folder.
3. Zip the folder and submit it to the server.