SAMSUNG CHORD SDK

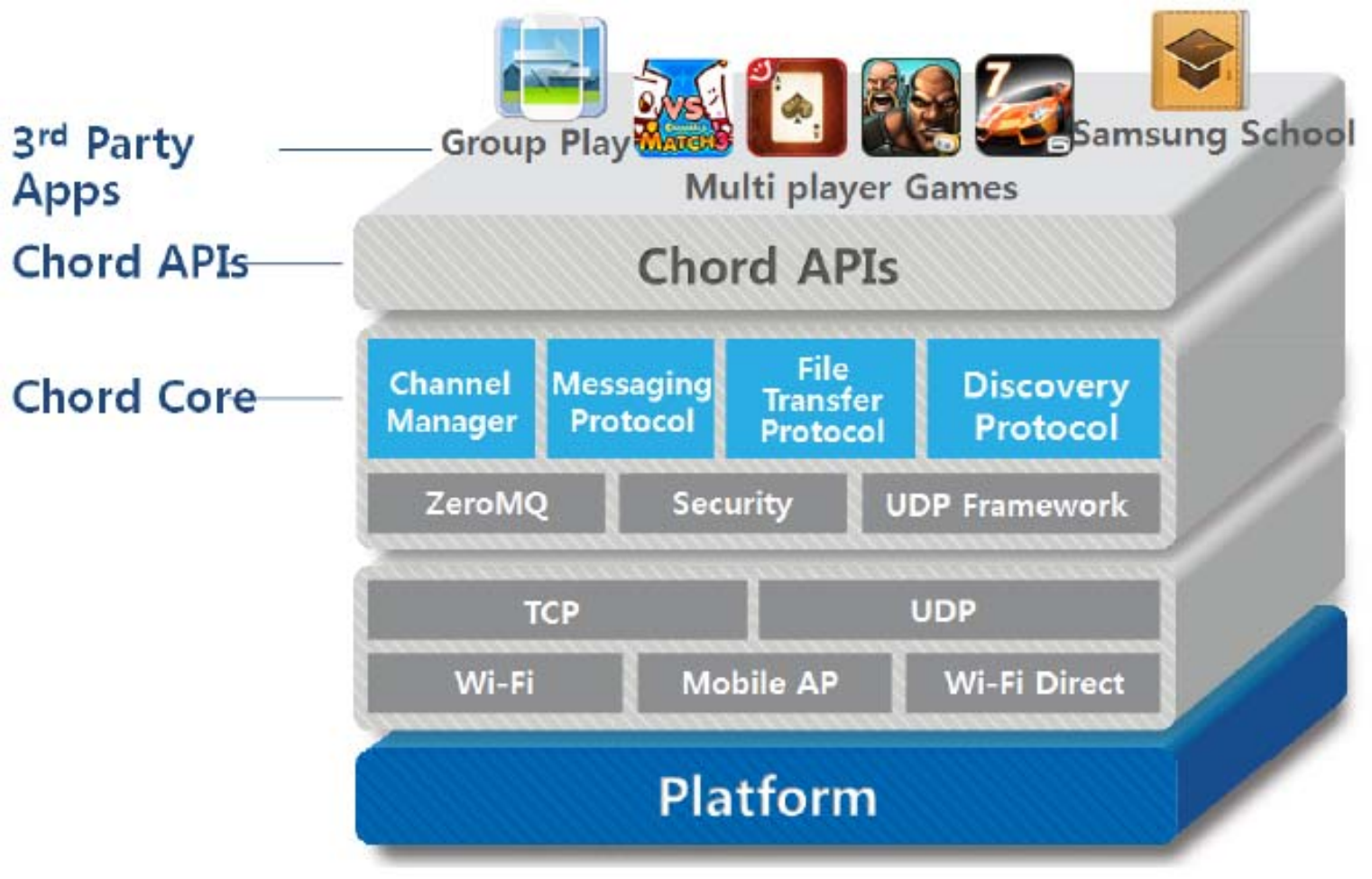
Instructor: Nguyễn Kiêm Hùng

WHAT IS CHORD SDK

- Chord allows you to easily share messages and content between devices in the same (local) subnet in real-time
- Requirements:
  - Devices with Android 4.0 Ice Cream Sandwich (API level 14) or higher support Chord.
  - Chord does not support Atom x86 for Android API 17

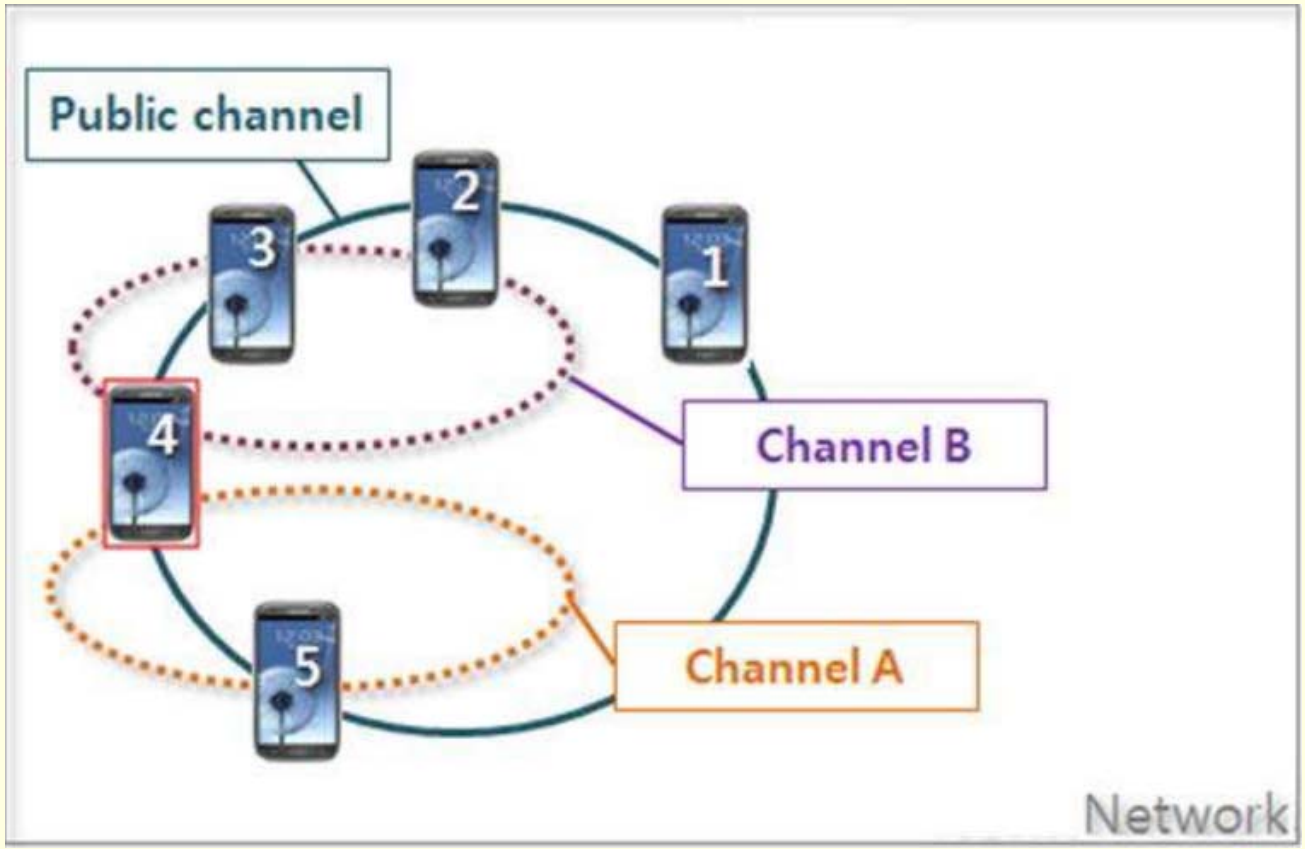
WHAT IS CHORD SDK

- **Chord SDK Architecture**



WHAT IS CHORD SDK

- Every node is part of public Chord channel.
- Only nodes using the same app can interact with each other by a private channel.



USING CHORD SDK

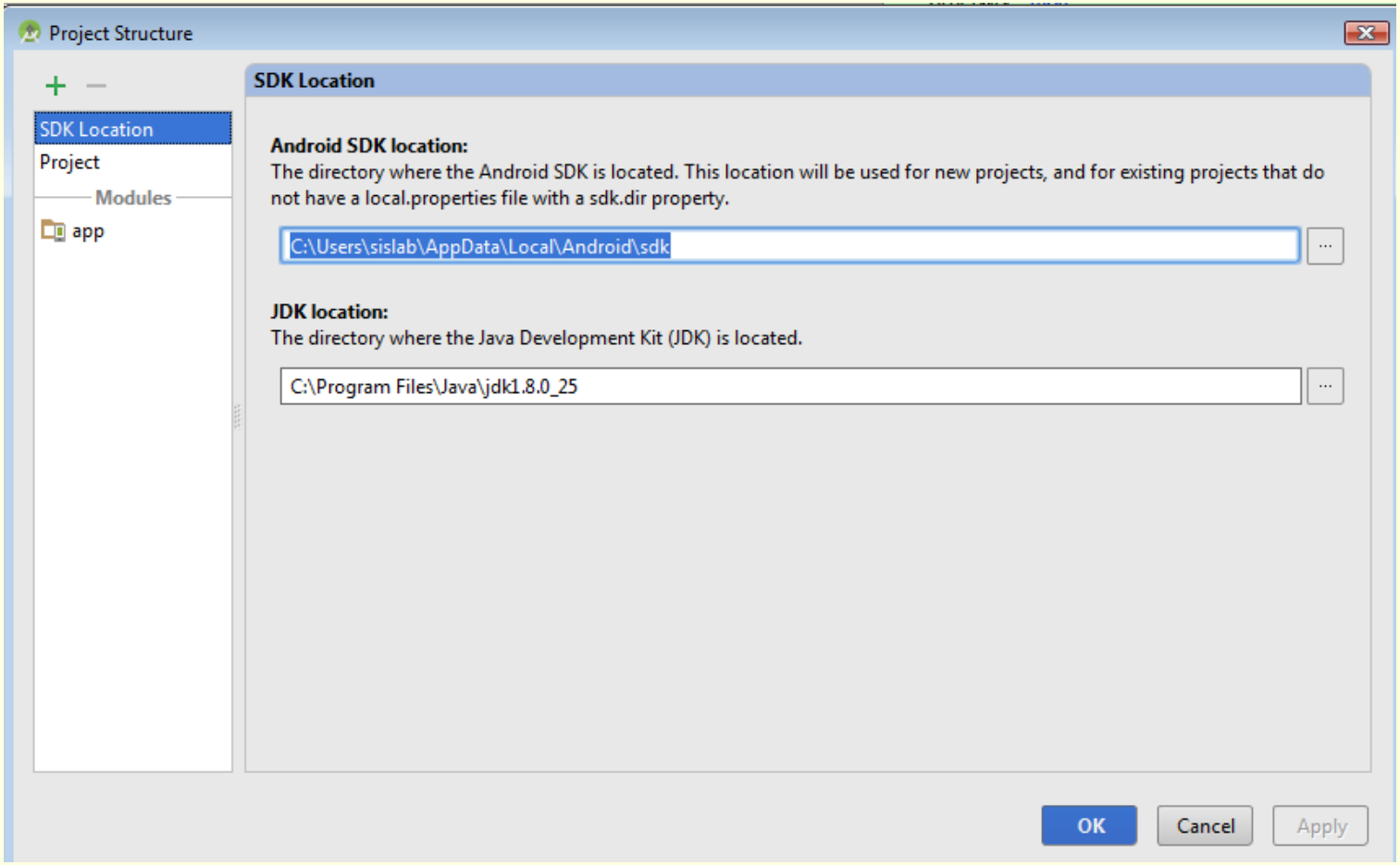
- **Samsung Device Requirements:**
  - Android 4.0 Ice Cream Sandwich (API level 14)

USING CHORD SDK

- **System Requirements**
  - JDK (Java Development Kit): required for developing and running Java applications.
    - Download and Install Java 8 for Microsoft Windows: http://www.oracle.com/technetwork/java/javase/downloads/index.html
  - **Android Studio (Including IDE + ADT plug-in, Emulator …):** provides everything you need to start developing apps for Android
    - Download and install Android Studio: http://developer.android.com/sdk/index.html
- **Downloading the Chord SDK with the Android SDK Manager**

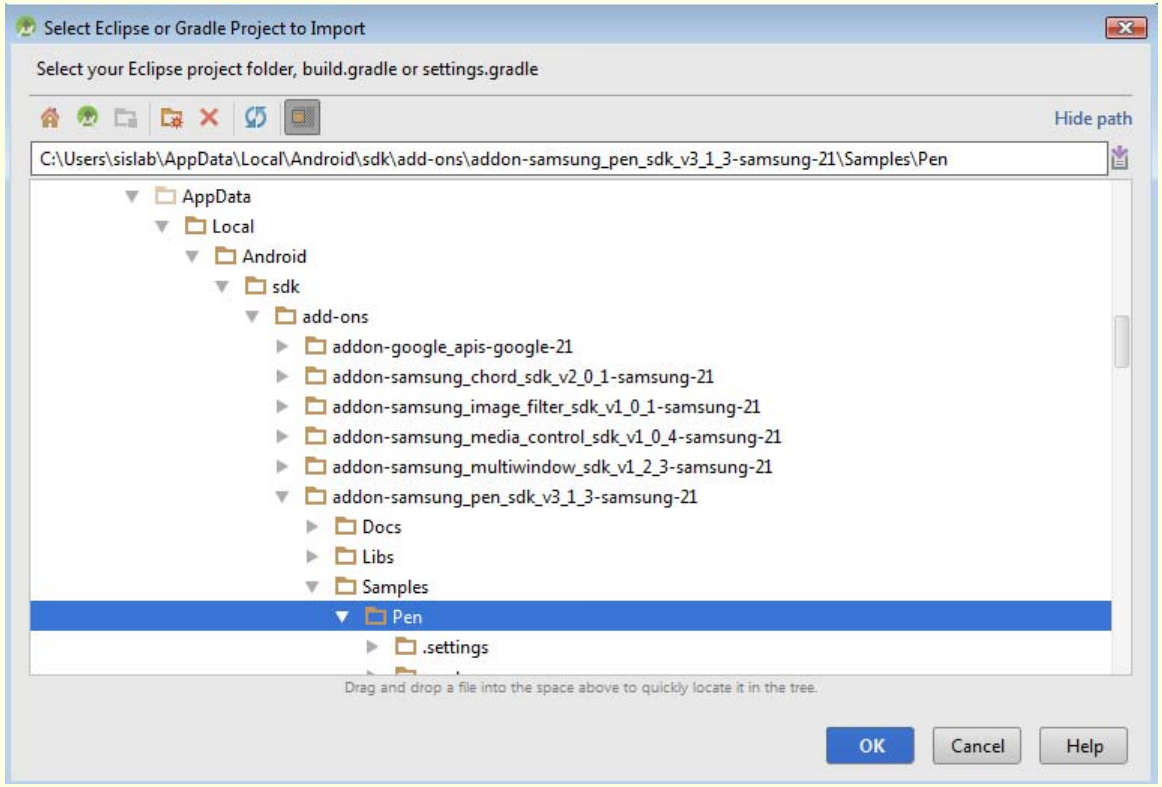
USING CHORD SDK

- **Setup JDK Location in Android Studio**
  - Select **File – Project Structure**



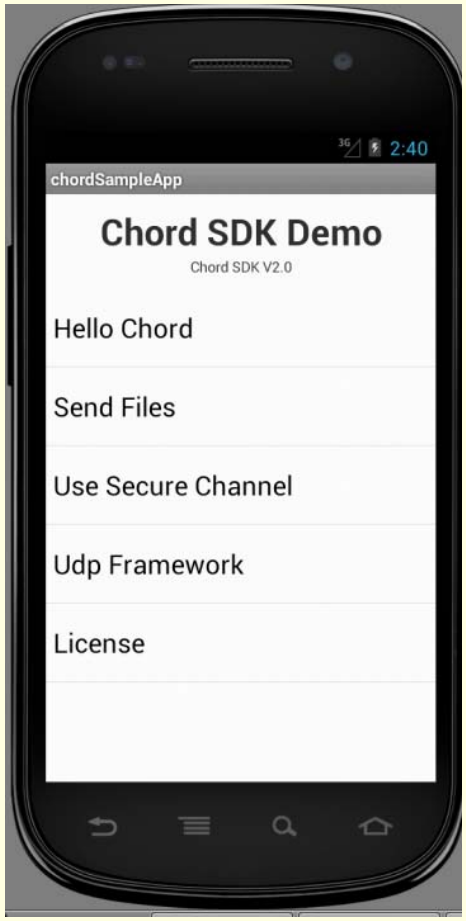
RUNNING SAMPLE APPLICATION

- Run **Android Studio** and click **File → Import Project…** to open the **Import** dialog
  - The sample application for Chord SDK is in the 'Samples' directory of the downloaded SDK package



RUNNING SAMPLE APPLICATION

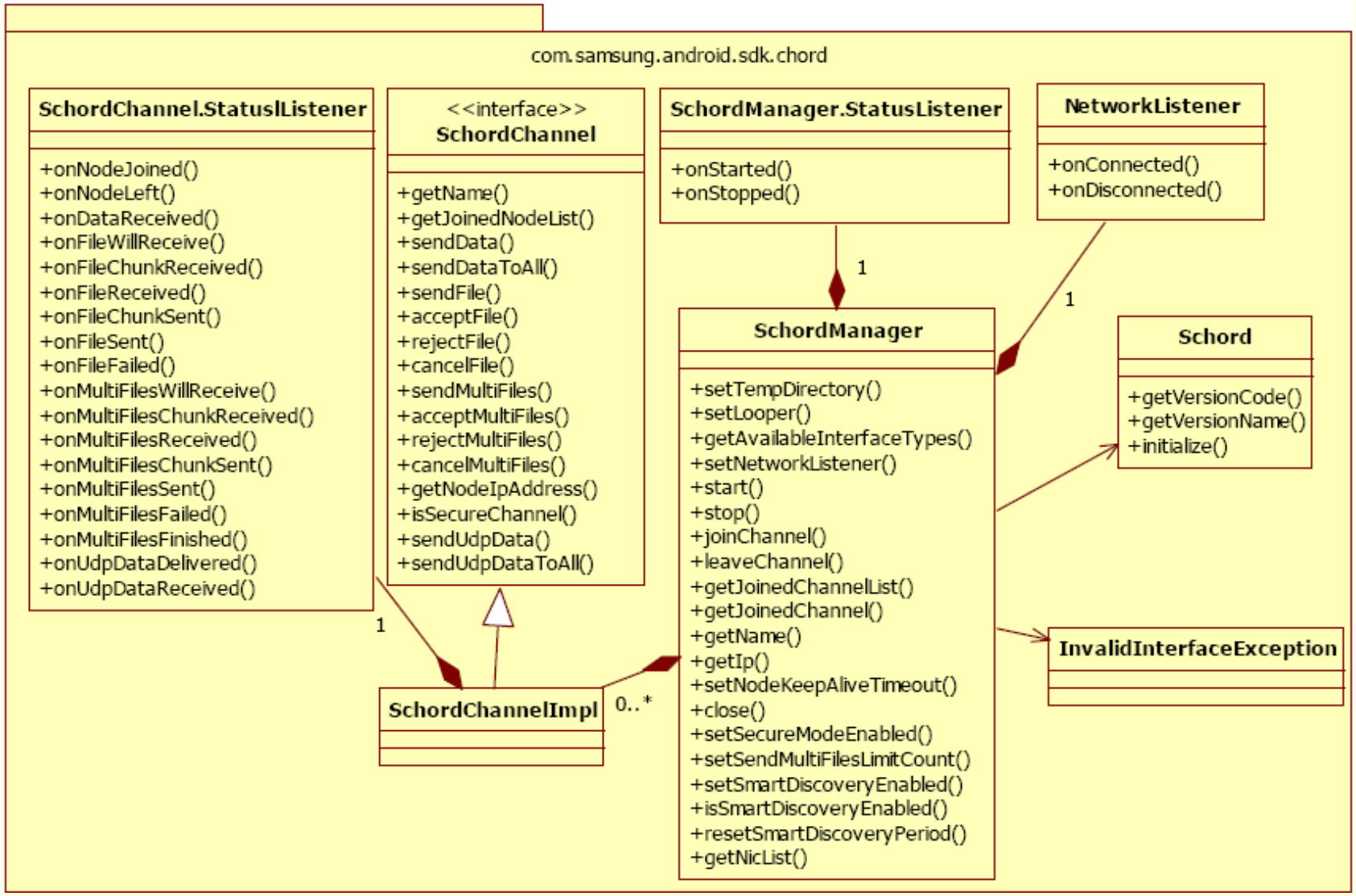
- S Chord sample application include all features of the Chord SDK package.



CHORD PROGRAMING GUIDE

- **Libs' Components**
  - chord-v2.0.1.jar
  - libchord-v2.0.so

CHORD PROGRAMING GUIDE

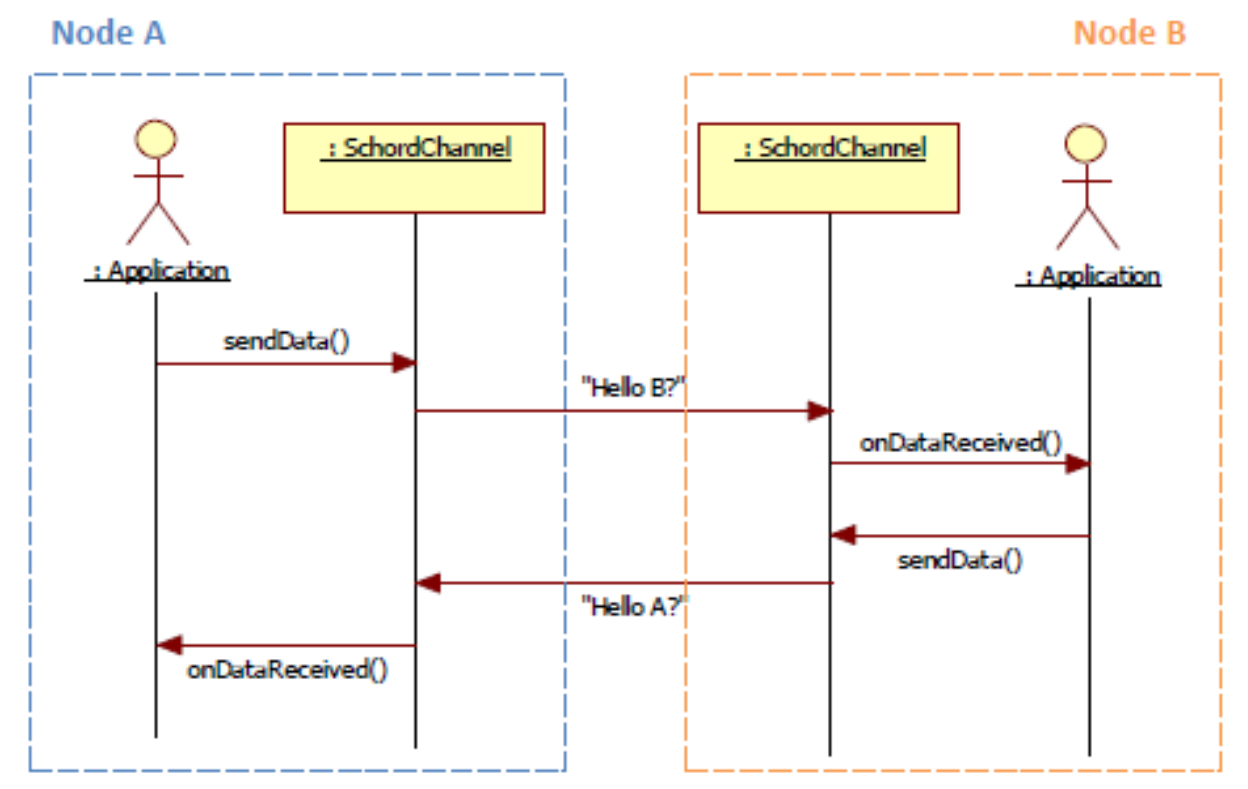


FIRT PROGRAM: Hello Chord

- **Hello Chord is a simple program that:**
  - Joins a channel.
  - Sends the string "Hello Chord!" to another node in the channel.
  - Receives messages and shows on viewport.

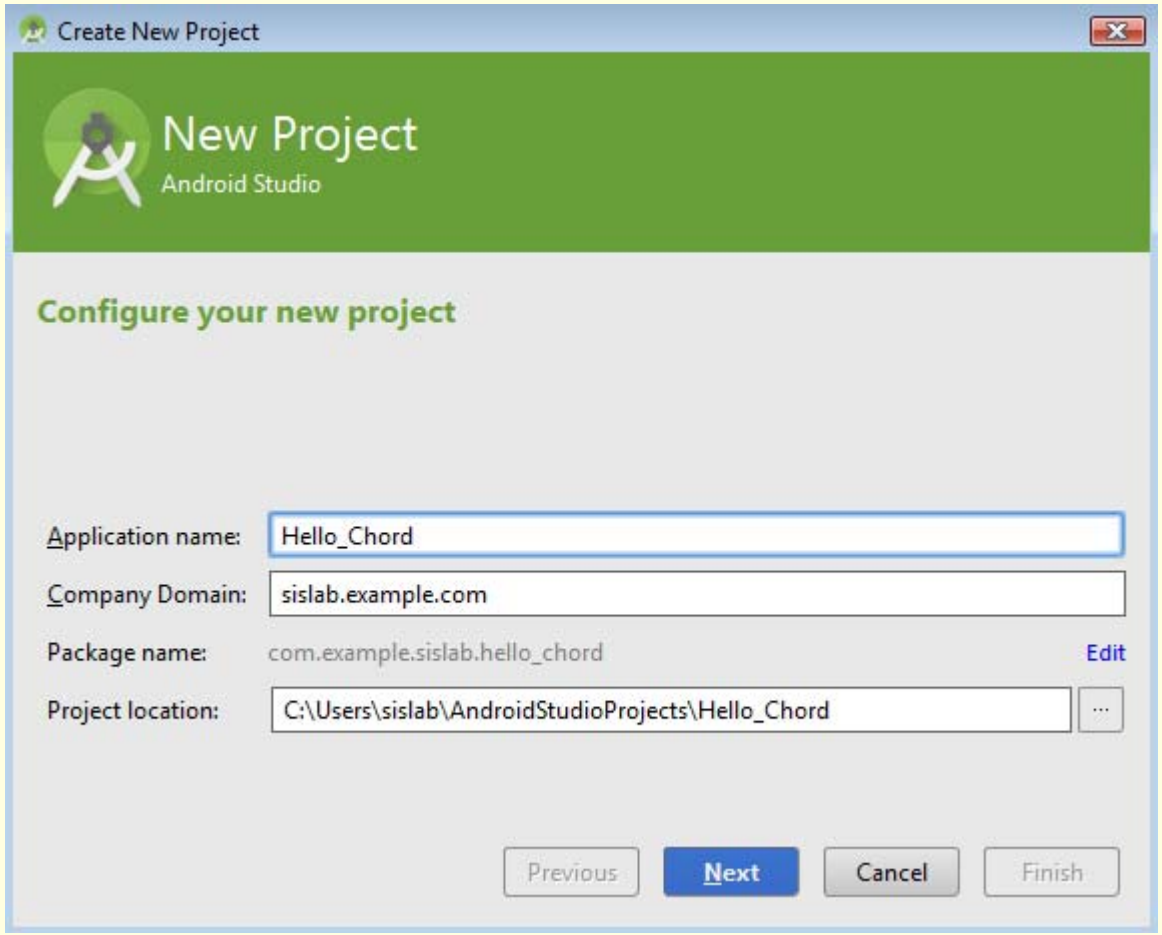
FIRT PROGRAM: Hello Chord

- **Hello Chord is a simple program that:**



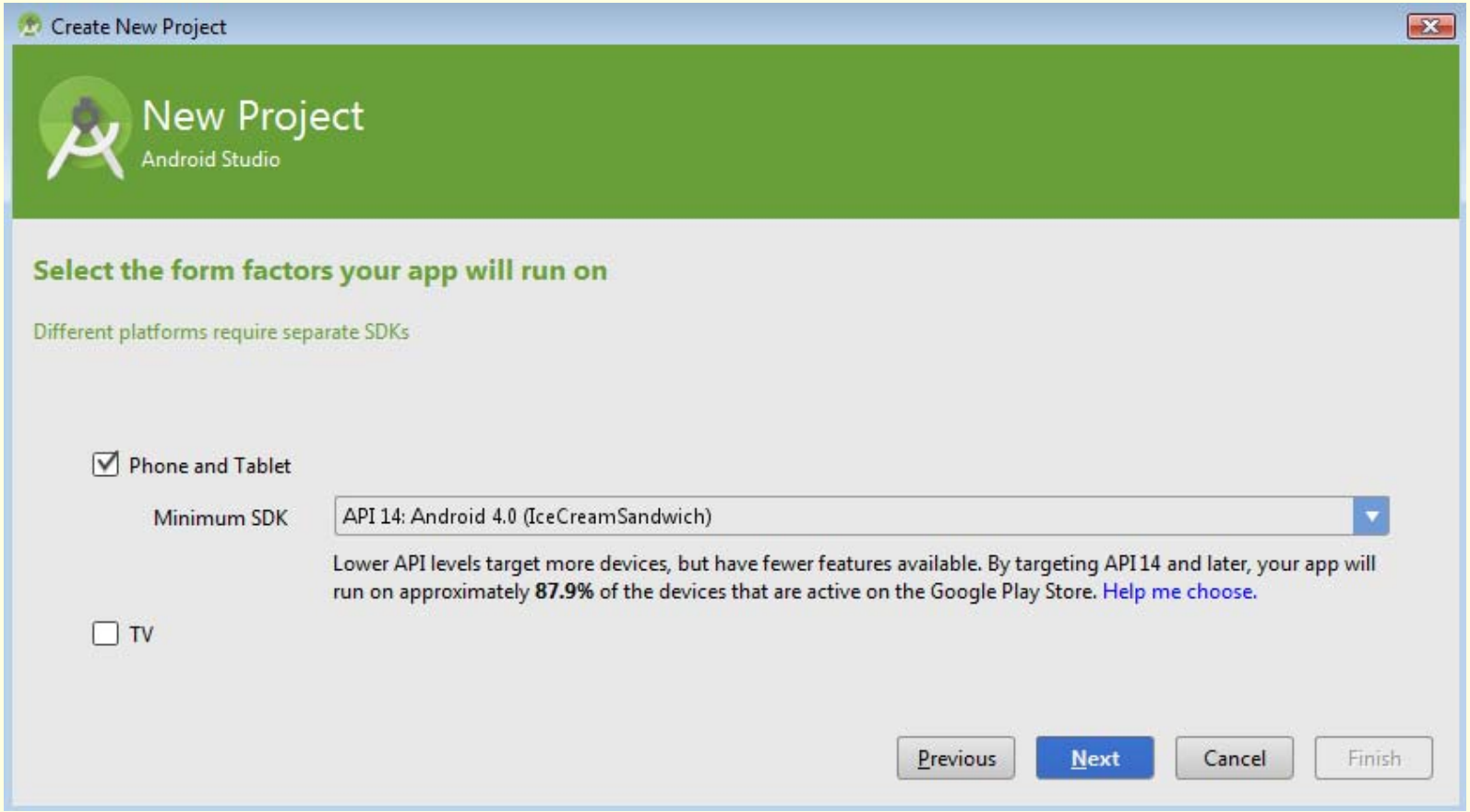
FIRT PROGRAM: Hello Chord

- **Creating a New Project in Android Studio**



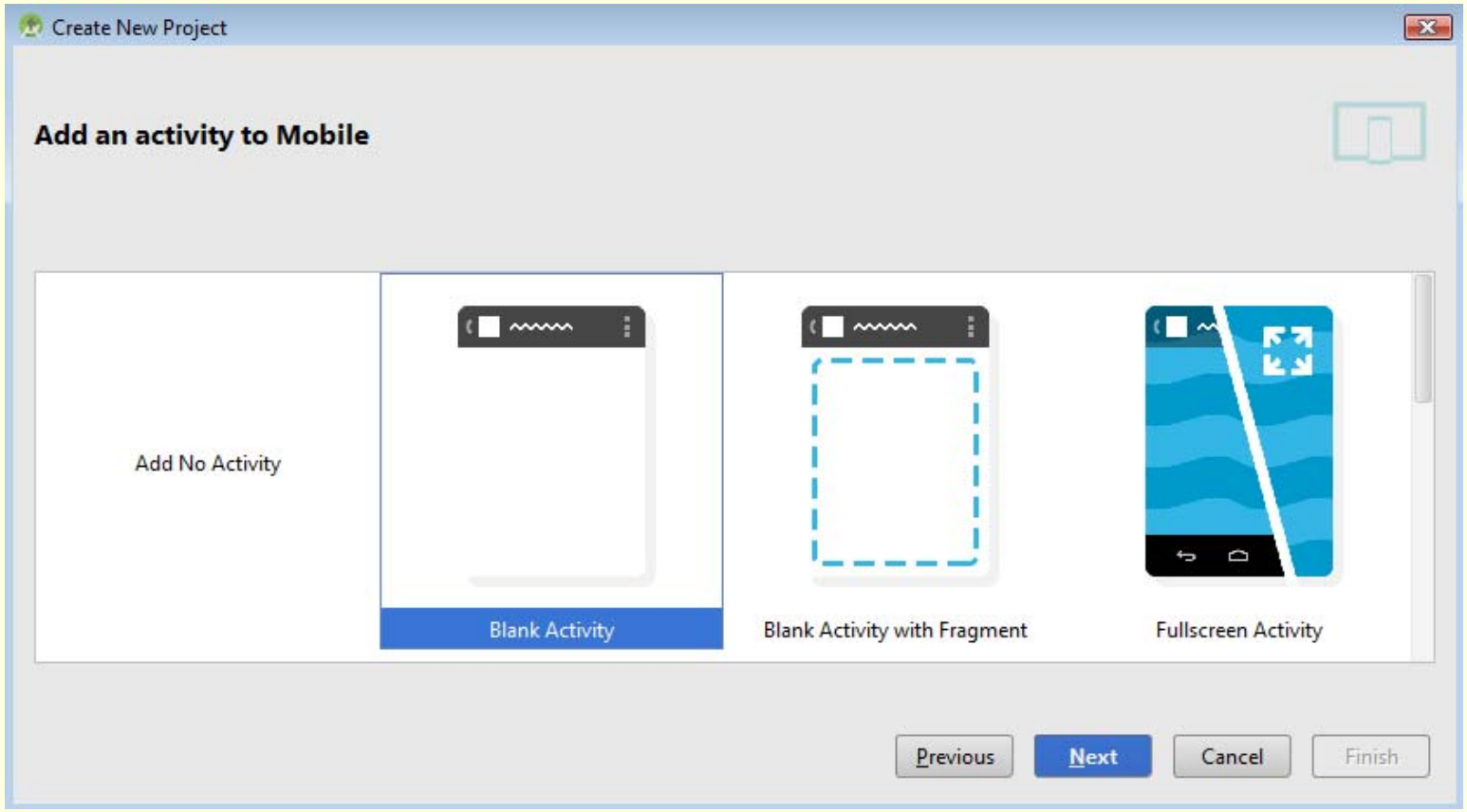
FIRT PROGRAM: Hello Chord

- **Creating a New Project in Android Studio**



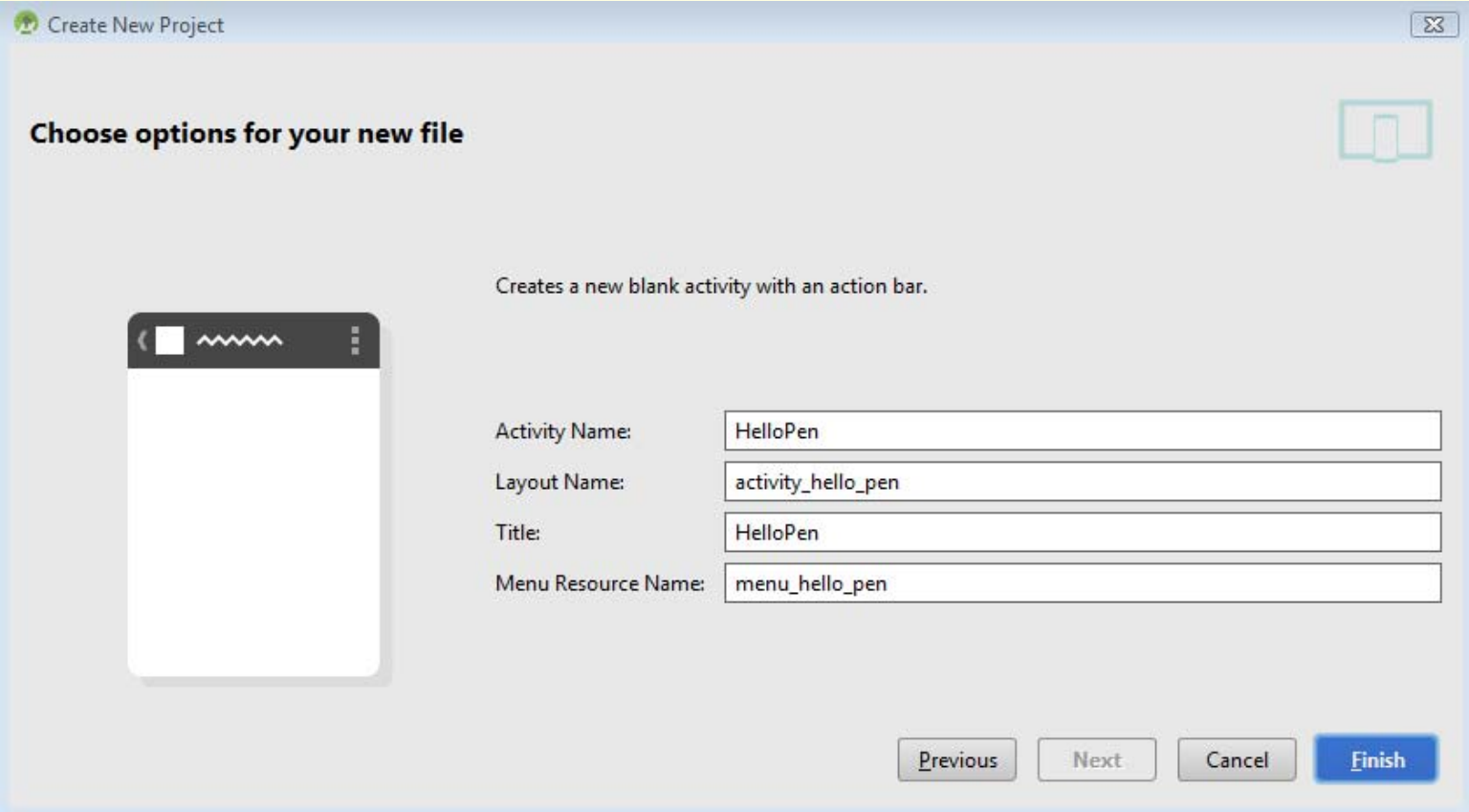
FIRT PROGRAM: Hello Chord

- **Creating a New Project in Android Studio**



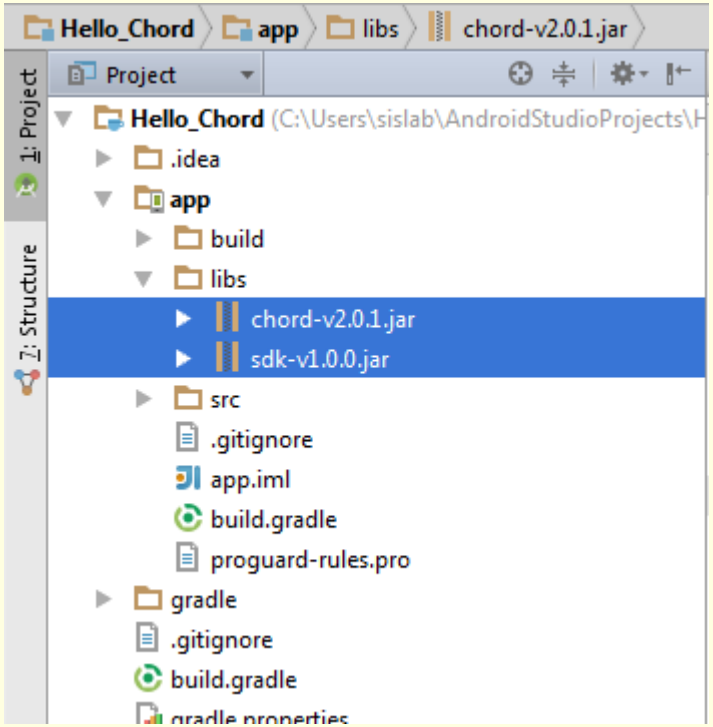
FIRT PROGRAM: Hello Chord

- **Creating a New Project in Android Studio**



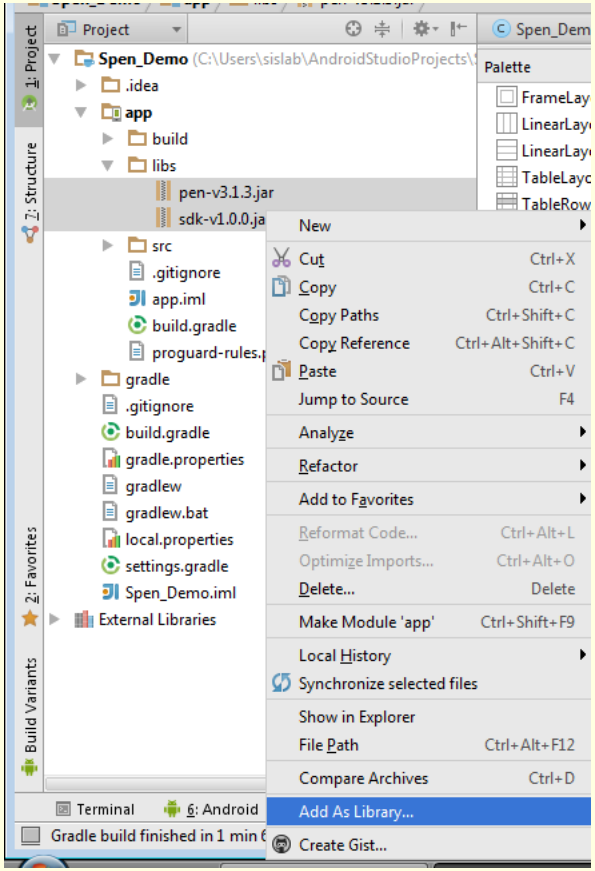
FIRT PROGRAM: Hello Chord

- **Adding a Library** of the downloaded PEN SDK to **"libs"** folder of the newly created project.
  - Copy the **SDK .jar files** to the **'libs'** folder in your new project to use the SDK you need for your application.



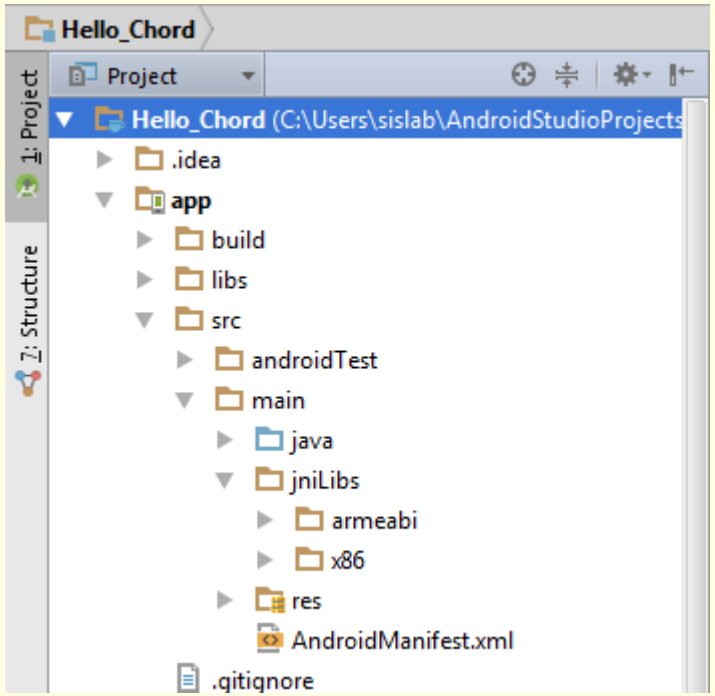
FIRT PROGRAM: Hello Chord

- **Adding a Library** of the downloaded PEN SDK to **"libs"** folder of the newly created project.
  - Select **.jar files** and Right click to show a pop-down menu, select **"Add as libraries"**.



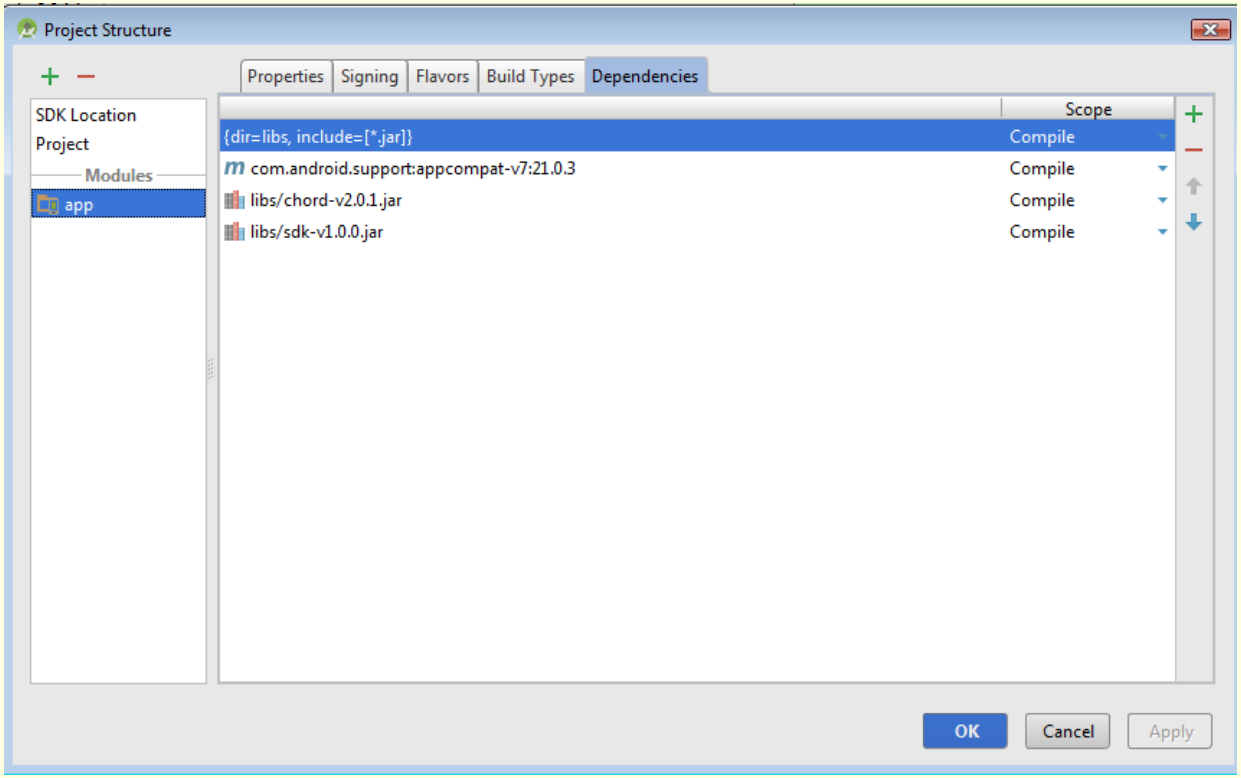
FIRT PROGRAM: Hello Chord

- **Adding a Library** of the downloaded Chord SDK to **"MAIN"** folder of the newly created project.
  - Create a "jnitlibs" folder in **"MAIN"** folder of the newly created project
  - Copy the "**armeabi" and "x86" folders** IN "libs" of the downloaded Chord SDK to the 'jnitlibs ' folder.



FIRT PROGRAM: Hello Chord

- **Adding a Library** of the downloaded PEN SDK to **"libs"** folder of the newly created project.
  - Check **.jar files** have been added into the project as shown in figure.



FIRT PROGRAM: Hello Chord

- **Add the following permission to your Android manifest file** *(\app\src\main\AndroidManifest.xml*):

// add permission

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission
android:name="com.samsung.android.providers.context.permission.WRITE_USE_APP_FEATURE_SURVEY" />
```

FIRT PROGRAM: Hello Chord

- **Edit "activity_hello_chord.xml" file:**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical" >

   <LinearLayout style="@style/Layout_mw_horizontal" >

      <Button
         android:id="@+id/start_stop_btn"
         style="@style/Layout_equalDivision_horizontal"
         android:text="@string/start" />
   </LinearLayout>

   <TextView
      android:id="@+id/myNodeName_textView"
      style="@style/Layout_myNodeName_textView" />
   <TextView
      style="@style/Layout_subTitle_textView"
      android:text="@string/log_viewer"
      android:autoText="false"
      android:textStyle="bold" />

   <LinearLayout style="@style/Layout_equalDivision_vertical" >

      <com.example.sislab.hellochordactivity.ChordLogView
         android:id="@+id/log_textView"
         style="@style/Layout_log_viewer" />

   </LinearLayout>
</LinearLayout>
```

FIRT PROGRAM: Hello Chord

- **Open "HelloChord.java" file and replace *class HelloChord* with the following codes:**

```
package com.example.sislab.hellochordactivity;

import com.samsung.android.sdk.SsdkUnsupportedException;
import com.samsung.android.sdk.chord.Schord;
import com.samsung.android.sdk.chord.SchordChannel;
import com.samsung.android.sdk.chord.SchordManager;
import com.samsung.android.sdk.chord.SchordManager.NetworkListener;

import android.app.Activity;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.util.Log;
import android.util.SparseIntArray;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import java.util.HashMap;
import java.util.List;
```

FIRT PROGRAM: Hello Chord

- **Open "HelloChord.java" file and replace *class HelloChord* with the following codes:**

```
@Override
  protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_hello_chord);


     mDrawableConnected.setBounds(0, 0, mDrawableConnected.getIntrinsicWidth(),
           mDrawableConnected.getIntrinsicHeight());
     mDrawableDisconnected.setBounds(0, 0, mDrawableDisconnected.getIntrinsicWidth(),
           mDrawableDisconnected.getIntrinsicHeight());

     mWifi_startStop_btn = (Button) findViewById(R.id.start_stop_btn);
     mWifi_startStop_btn.setOnClickListener(this);
     mWifi_startStop_btn.setEnabled(false);


     mMyNodeName_textView = (TextView) findViewById(R.id.myNodeName_textView);
     mMyNodeName_textView.setHint(getString(R.string.my_node_name, " "));

     mLogView = (TextView) findViewById(R.id.log_textView);

  }
```

FIRT PROGRAM: Hello Chord

- **Open "HelloChord.java" file and replace *class HelloChord* with the following codes:**

```
@Override
  public void onResume() {
      super.onResume();

      /**
       * [A] Initialize Chord!
       */
      if (mSchordManager_1 == null) {
          // mLogView.appendLog("\n[A] Initialize Chord!");
          initChord();
      }
  }

  @Override
  public void onDestroy() {
      /**
       * [D] Release Chord!
       */
      if (mSchordManager_1 != null) {
          /**
           * If you registered NetworkListener, you should unregister it.
           */
          mSchordManager_1.setNetworkListener(null);

          mSchordManager_1.close();
          mSchordManager_1 = null;
      }
```

FIRT PROGRAM: Hello Chord

- **Open "HelloChord.java" file and replace *class HelloChord* with the following codes:**

```
if (mSchordManager_2 != null) {
        mSchordManager_2.close();
        mSchordManager_2 = null;
    }

    if (mSchordManager_3 != null) {
        mSchordManager_3.close();
        mSchordManager_3 = null;
    }

    mNodeNumberMap.clear();
    mInterfaceMap.clear();

    super.onDestroy();
  }

  @Override
  public void onClick(View v) {
    boolean bStarted = false;
    int ifc = -1;
    switch (v.getId()) {
        case R.id.start_stop_btn:
            bStarted = mWifi_bStarted;
            ifc = SchordManager.INTERFACE_TYPE_WIFI;
         break;
    }
```

FIRT PROGRAM: Hello Chord

- **Open "HelloChord.java" file and replace *class HelloChord* with the following codes:**

```
if (!bStarted) {
        /**
         * [B] Start Chord
         */
        addLogView(ifc, "\n[A] Start Chord!");
        startChord(ifc);
    } else {
        /**
         * [C] Stop Chord
         */
        addLogView(ifc, "\n[B] Stop Chord!");
        stopChord(ifc);
    }

  }

  private void initChord() {

    Schord chord = new Schord();
    try {
        chord.initialize(this);
    } catch (SsdkUnsupportedException e) {
        if (e.getType() == SsdkUnsupportedException.VENDOR_NOT_SUPPORTED) {
            // Vender is not SAMSUNG
            return;
        }
    }
```

FIRT PROGRAM: Hello Chord

- **Open "HelloChord.java" file and replace *class HelloChord* with the following codes:**

```
Log.d(TAG, TAGClass + "initChord : VersionName( " + chord.getVersionName() + " ), VerionCode( " + chord.getVersionCode()+" )");

    mSchordManager_1 = new SchordManager(this);
    mSchordManager_1.setLooper(getMainLooper());
     mSchordManager_1.setNetworkListener(new NetworkListener() {
       @Override
       public void onDisconnected(int interfaceType) {
           Toast.makeText(getApplicationContext(),
                   getInterfaceName(interfaceType) + " is disconnected", Toast.LENGTH_SHORT)
                   .show();
           refreshInterfaceStatus(interfaceType, false);
       }

       @Override
       public void onConnected(int interfaceType) {
           Toast.makeText(getApplicationContext(),
                   getInterfaceName(interfaceType) + " is connected", Toast.LENGTH_SHORT)
                   .show();
           refreshInterfaceStatus(interfaceType, true);
       }
    });
    List<Integer> ifcList = mSchordManager_1.getAvailableInterfaceTypes();
    for (Integer ifc : ifcList) {
       refreshInterfaceStatus(ifc, true);
    }

}
```

FIRT PROGRAM: Hello Chord

- **Open "HelloChord.java" file and replace *class HelloChord* with the following codes:**

```
private void refreshInterfaceStatus(int interfaceType, boolean bConnected) {

        if (!bConnected) {
            if (interfaceType == SchordManager.INTERFACE_TYPE_WIFI) {
           mWifi_startStop_btn.setEnabled(false);

            } else if (interfaceType == SchordManager.INTERFACE_TYPE_WIFI_P2P) {

            } else if (interfaceType == SchordManager.INTERFACE_TYPE_WIFI_AP) {

            }
        } else {
            if (interfaceType == SchordManager.INTERFACE_TYPE_WIFI) {
              mWifi_startStop_btn.setEnabled(true);

            } else if (interfaceType == SchordManager.INTERFACE_TYPE_WIFI_P2P) {

            } else if (interfaceType == SchordManager.INTERFACE_TYPE_WIFI_AP) {
}
        }

    }
```

FIRT PROGRAM: Hello Chord

- **Open "HelloChord.java" file and replace *class HelloChord* with the following codes:**

```
private void startChord(int interfaceType) {

    int managerIndex = 0;
    SchordManager startManager = null;
    if (mInterfaceMap.get(interfaceType) == 0) {
        managerIndex = mInterfaceMap.size() + 1;
        mInterfaceMap.put(interfaceType, managerIndex);
    } else {
        managerIndex = mInterfaceMap.get(interfaceType);
    }

    switch (managerIndex) {
        case 1:
            startManager = mSchordManager_1;
            break;
        case 2:
            mSchordManager_2 = new SchordManager(this);
            startManager = mSchordManager_2;
            break;
        case 3:
            mSchordManager_3 = new SchordManager(this);
            startManager = mSchordManager_3;
            break;
    }
```

FIRT PROGRAM: Hello Chord

- **Open "HelloChord.java" file and replace *class HelloChord* with the following codes:**

```
try {
        Log.d(TAG, TAGClass + "start(" + getInterfaceName(interfaceType)
            + ") with the SchordManager number : " + managerIndex);

        startManager.setLooper(getMainLooper());

        switch (interfaceType) {
          case SchordManager.INTERFACE_TYPE_WIFI:
            startManager.start(interfaceType, mWifi_ManagerListener);
            mWifi_startStop_btn.setEnabled(false);
            break;
        }
        addLogView(interfaceType, "    start(" + getInterfaceName(interfaceType) + ")");

    } catch (Exception e) {
        addLogView(interfaceType, "    Fail to start -" + e.getMessage());
        mInterfaceMap.delete(interfaceType);
    }
}
```

FIRT PROGRAM: Hello Chord

- **Open "HelloChord.java" file and replace *class HelloChord* with the following codes:**

```
// ************************************************
  // ChordManagerListener
  // ************************************************
  private SchordManager.StatusListener mWifi_ManagerListener = new SchordManager.StatusListener() {

     @Override
     public void onStarted(String nodeName, int reason) {
              //4. Chord has started successfully
      mWifi_bStarted = true;
     mLogView.setVisibility(View.VISIBLE);
     mWifi_startStop_btn.setText(R.string.stop);
     mWifi_startStop_btn.setEnabled(true);

     if (reason == STARTED_BY_USER) {
        // Success to start by calling start() method
        mLogView.setText("    > onStarted(STARTED_BY_USER)");
        joinTestChannel(SchordManager.INTERFACE_TYPE_WIFI);
     } else if (reason == STARTED_BY_RECONNECTION) {
        // Re-start by network re-connection.
        mLogView.setText("    > onStarted(STARTED_BY_RECONNECTION)");
     }

    }
```

FIRT PROGRAM: Hello Chord

- **Open "HelloChord.java" file and replace *class HelloChord* with the following codes:**

```
 @Override
      public void onStopped(int reason) {
         /**
          * 8. Chord has stopped successfully
          */
         mWifi_bStarted = false;
/*
         if (!mWifiDirect_bStarted) {
            mMyNodeName_textView.setText("");
            mMyNodeName_textView.setHint(getString(R.string.my_node_name, " "));
         }
*/
         mWifi_startStop_btn.setText(R.string.start);

         if (STOPPED_BY_USER == reason) {
            // Success to stop by calling stop() method
            mLogView.setText("    > onStopped(STOPPED_BY_USER)");
            mWifi_startStop_btn.setEnabled(true);

         } else if (NETWORK_DISCONNECTED == reason) {
            // Stopped by network disconnected
            mLogView.setText("    > onStopped(NETWORK_DISCONNECTED)");
            mWifi_startStop_btn.setEnabled(false);
         }
      }
   };
```

FIRT PROGRAM: Hello Chord

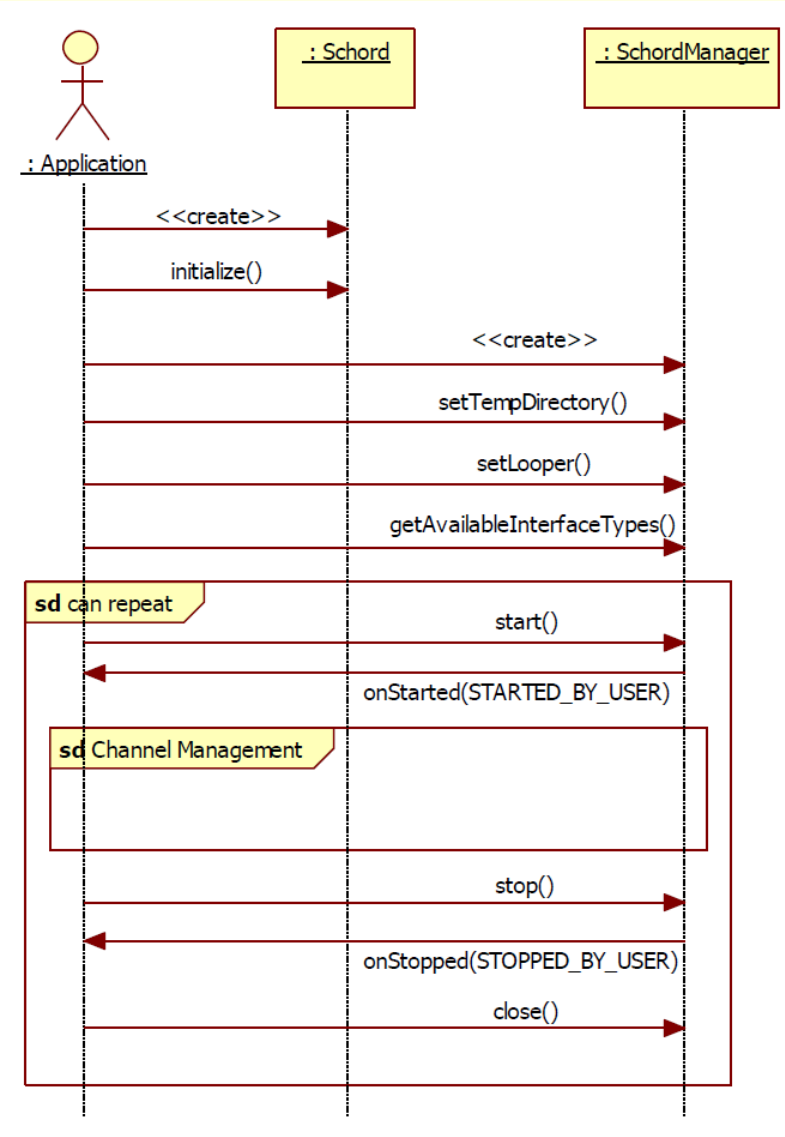
- **Open "HelloChord.java" file and replace *class HelloChord* with the following codes:**

```
private void joinTestChannel(int interfaceType) {
    /**
     * 5. Join my channel
     */
    addLogView(interfaceType, "    joinChannel()");

    SchordChannel channel = null;
    SchordManager currentManager = null;

    currentManager = getSchordManager(interfaceType);

    switch (interfaceType) {
      case SchordManager.INTERFACE_TYPE_WIFI:
        channel = currentManager.joinChannel(CHORD_HELLO_TEST_CHANNEL,
              mWifi_ChannelListener);
        break;

    }

    if (channel == null) {
      addLogView(interfaceType, "    Fail to joinChannel");
    }

  }
```

FIRT PROGRAM: Hello Chord

- **Open "HelloChord.java" file and replace *class HelloChord* with the following codes:**
  - **Refere to Hello_Chord.pdf for more details**

Using the Chord Class



Using the Chord Class

- Create an instance of Schord.
- Initialize the Chord by the following method:

  void initialize (Context context) throws SsdkUnsupportedException

  - If the device does not support Chord, sdkUnsupportedException is thrown.

Using the Chord Class

```
// Initialize Schord
Schord chord = new Schord();
try {
    // Initialize an instance of Schord.
    chord.initialize(this);
    } catch (SsdkUnsupportedException e) {
    if(e.getType()==SsdkUnsupportedException.VENDOR_NOT_SUPPORTED) {
    // Vendor is not Samsung

}
}
```

Using the Chord Class

- Create an instance of SchordManager.
- Call the following methods:
  - **setTempDirectory() sets a temporary directory for Chord functions.**
  - **setLooper() sets a looper object associated with the thread for processing callbacks.**
  - **getAvailableInterfaceTypes() gets the list of available network interface types.**
  - **start() starts the Chord.**
- Once Chord starts, SchordManager calls the following callback method on the application:
  - onStarted(STARTED_BY_USER) indicates that Chord has started.
- At this point, Channel Management takes over. When the application is closed, use the following methods in your application:
  - stop() stops Chord.
  - onStopped(STOPPED_BY_USER) indicates that Chord has stopped.
  - close() releases the instance.

Using the Chord Class

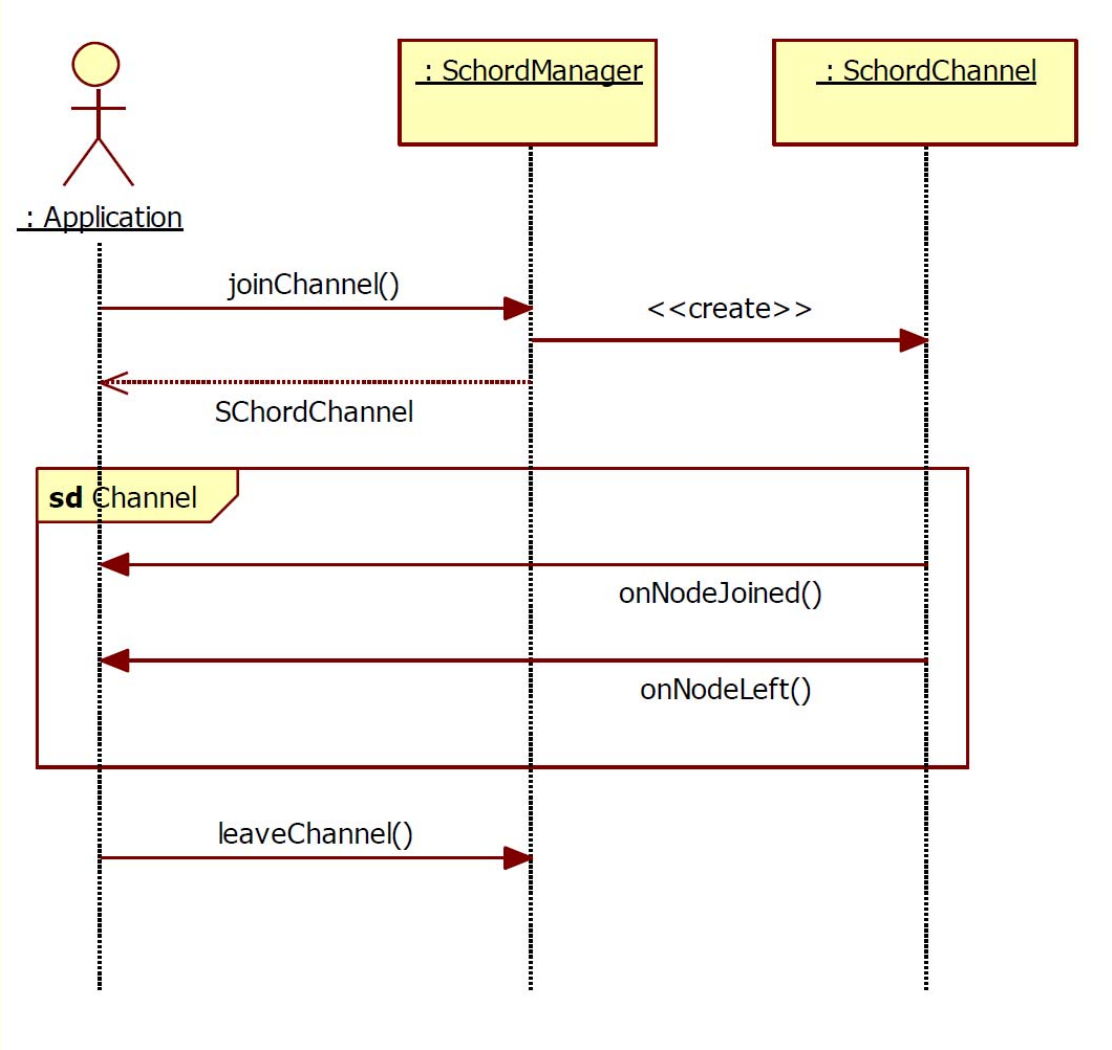
```
// create a instanse of SchordManager
        chordManager.setTempDirectory(tempDir);
        chordManager.setLooper(getMainLooper());
        List<Integer> interfaceList = mChordManager.getAvailableInterfaceTypes();
        if (interfaceList.isEmpty()) {
                // There is no connection.
                return;
        }
        chordManager.start (interfaceList.get(0).intValue(), new SchordManager.StatusListener() {
        @Override
        public void onStarted(String name, int reason) {

            if (STARTED_BY_USER == reason) {
            // Started
            }
        }
        @Override
        public void onStopped(int reason) {
            if (STOPPED_BY_USER == reason) {
            // Stopped
            }
        }
        @Override
        public void onDestroy {
        chordManager.stop();
        chordManager.close();
        }
```

Using the Chord Class

- **Joining and Leaving channels:**
  - Once SchordManager is running and the node has been created, call joinChannel() in your application to join the channel,
  - and use the returned SchordChannel to send and receive data on that channel.
  - To leave the channel, call leaveChannel().

Using the Chord Class

- **Joining and Leaving channels:**



Using the Chord Class

- **Sending and Receiving Data and Files:**
  - sendData() sends data to a specific node on a channel
  - sendDataToAll()sends data to all nodes on a channel.
  - sendFile()sends a file to a specific node on a channel.
  - sendUdpData()sends data using UDP to a specific node on a channel.
  - Etc.

Using the Chord Class

```
mChordManager.joinChannel (CHORD_HELLO_TEST_CHANNEL, new SchordChannel.StatusListener()
{
  @Override
public void onNodeJoined(String fromNode, String fromChannel) {
byte[][] payload = new byte[1][];
          payload[0] = "Hello A!".getBytes();
SchordChannel channel = mChordManager.getJoinedChannel(fromChannel);
// Send simple data.
channel.sendData(fromNode, CHORD_SAMPLE_MESSAGE_TYPE, payload);
       }
 @Override
public void onDataReceived(String fromNode, String fromChannel, String payloadType,
byte[][] payload) {
String receivedData = new String(payload[0]);
Massage_textView.setText("Received: "+ receivedData);

// Send “Hello B?”
byte[][] data = new byte[1][];
          data[0] = "Hello B?".getBytes();
SchordChannel channel = mChordManager.getJoinedChannel(fromChannel);
channel.sendData(fromNode, CHORD_SAMPLE_MESSAGE_TYPE,data);
} };
```

Using the Chord Class

Wifi Pass: e28a74d48b3c

Reference:

ProgrammingGuide_Chord.pdf