Session: 17

# Android Interface Definition Language

# Objectives

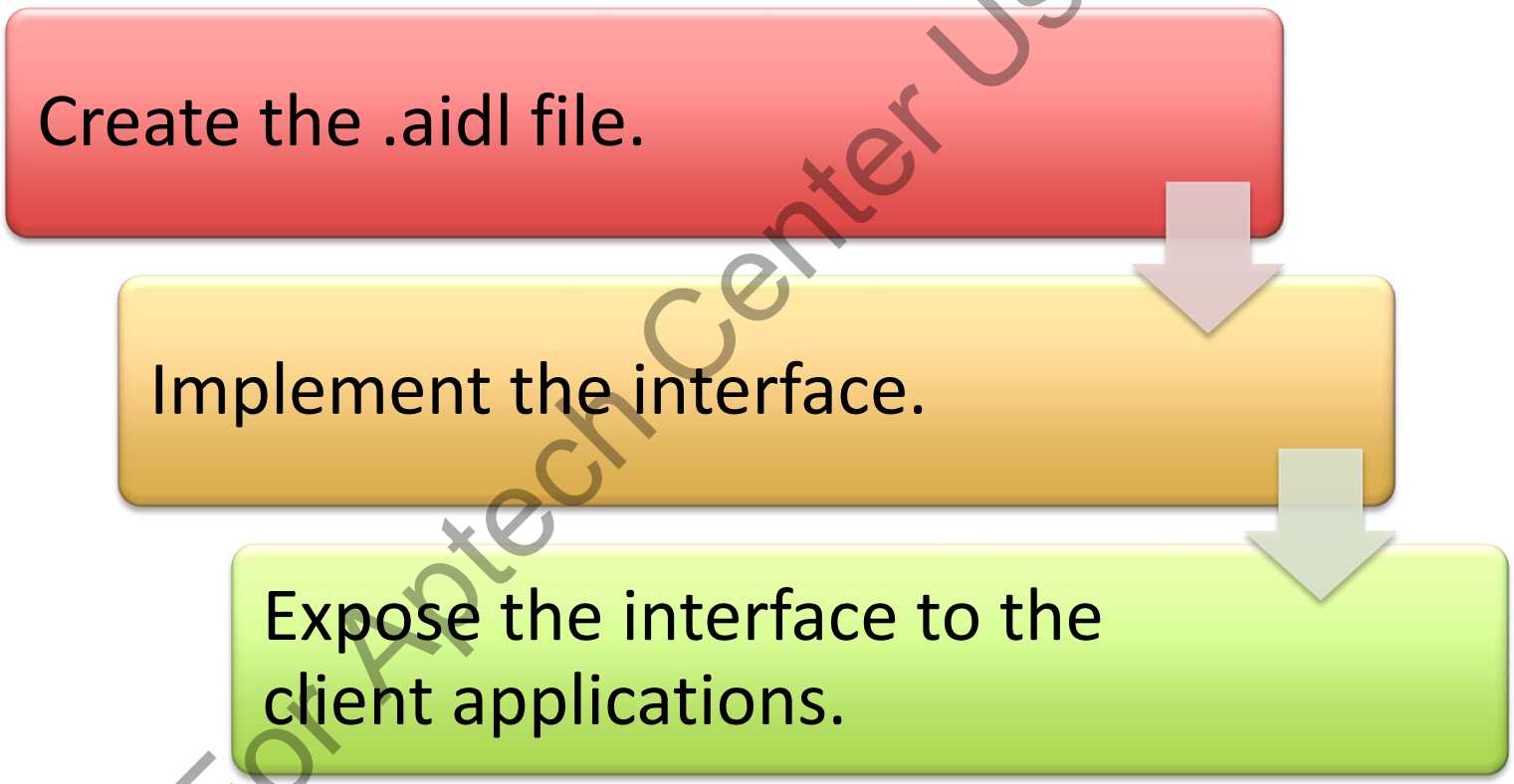
- Explain the AIDL Interface
- Explain how to create an AIDL file
- Explain how to implement the interface
- Explain the process of communication through IPC
- Explain how the IPC method is invoked

# Introduction to AIDL

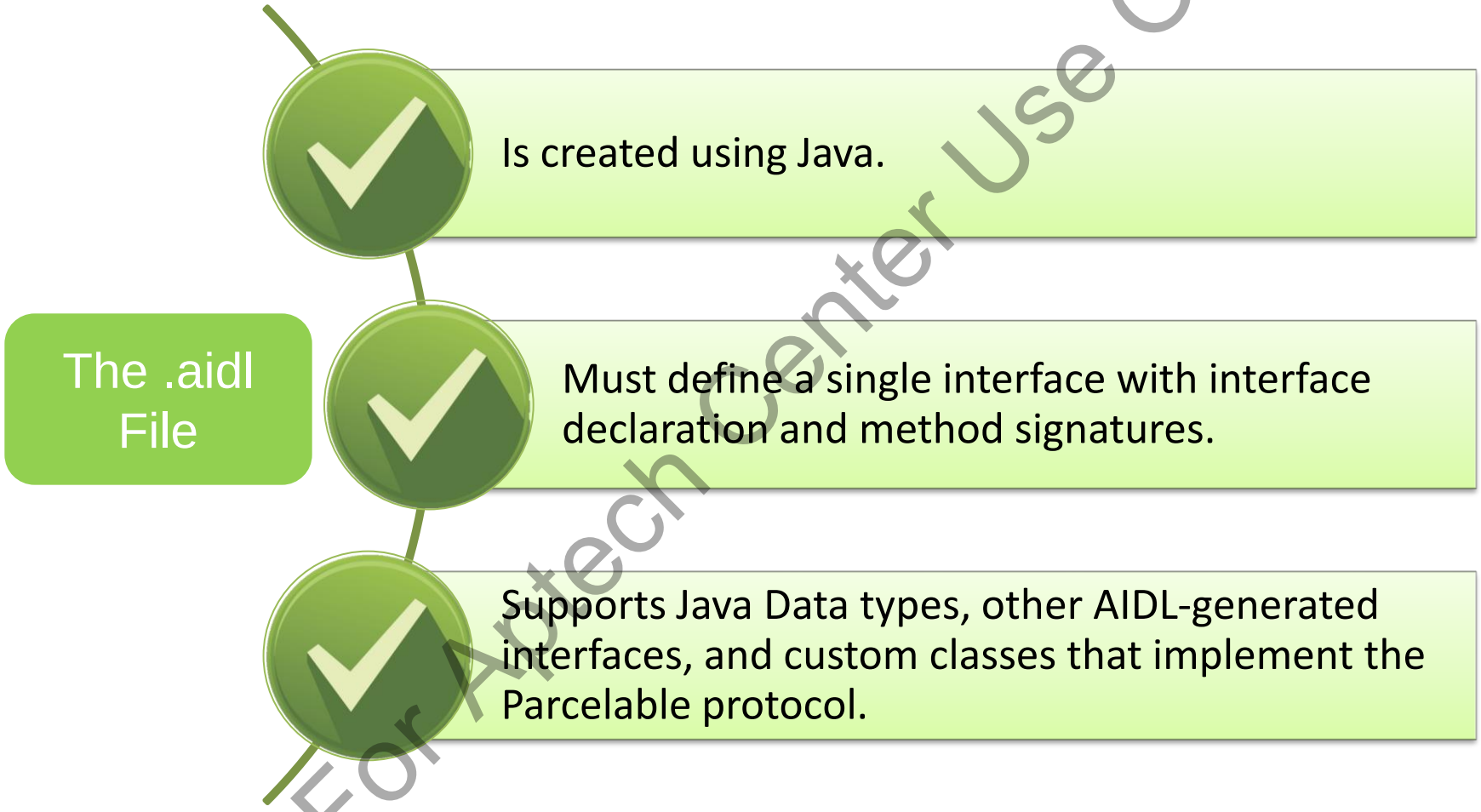
- App1 processes need to connect to the processes in App2.
- App1 is the service provider and App2 is the client.
- AIDL helps in communication between the processes with the help of Interprocess Communication (IPC).

# AIDL Interface

- Designing the AIDL interface involves the following three steps:



# Create an AIDL file 1-6



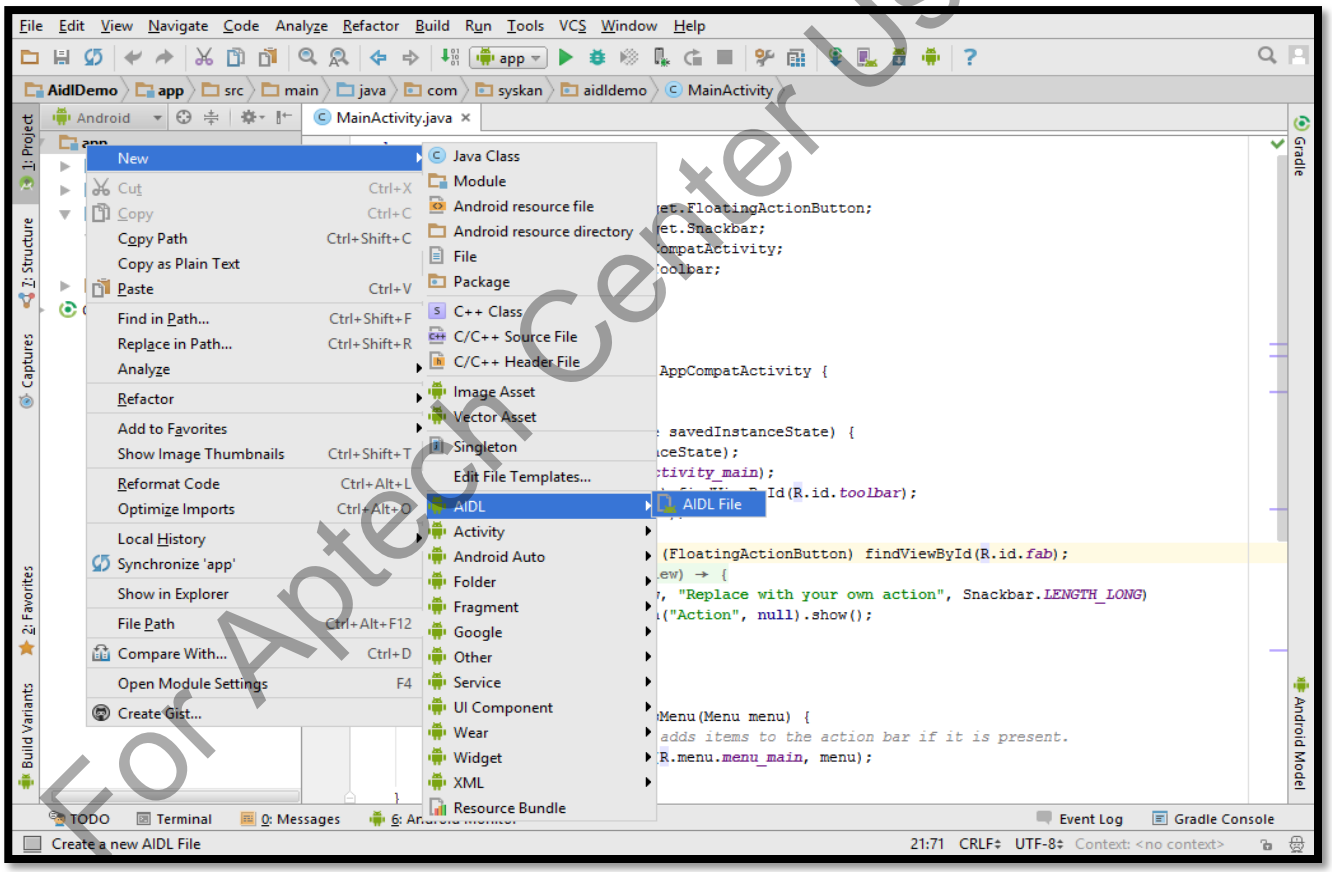
# Create an AIDL file 2-6

## Defining a Service Interface

- Some of the important factors that the user has to understand before defining the service interface are:
  - Parameters of the methods can be zero.
  - The methods can return a value or can be void.
  - The non-primitive data types should have a directional tag indicating the direction of data.
  - AIDL does not support static fields.
  - The code comments in .aidl file are included in the IBinder interface.

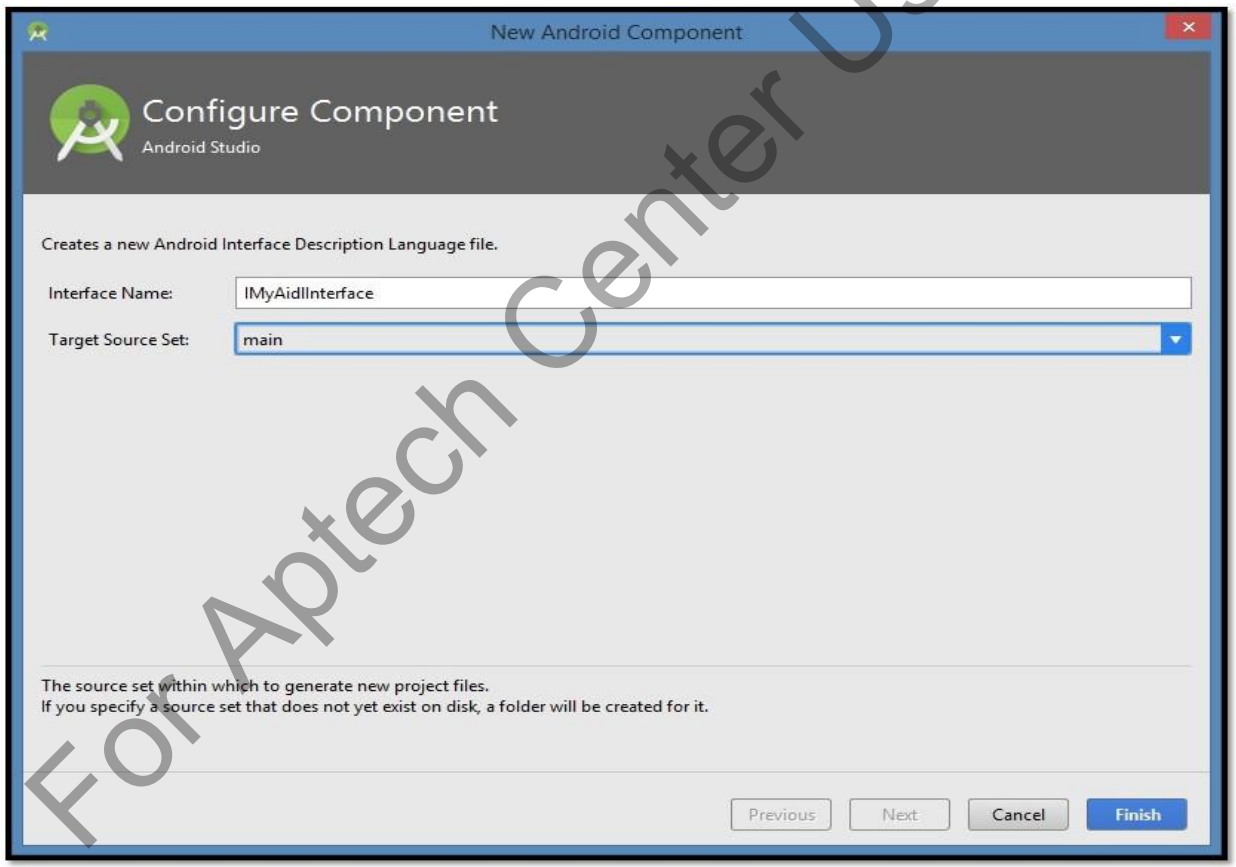
# Create an AIDL file 3-6

- To add an AIDL interface, right-click in the Navigator pane, and then point to **New → AIDL** and click **AIDL File**.



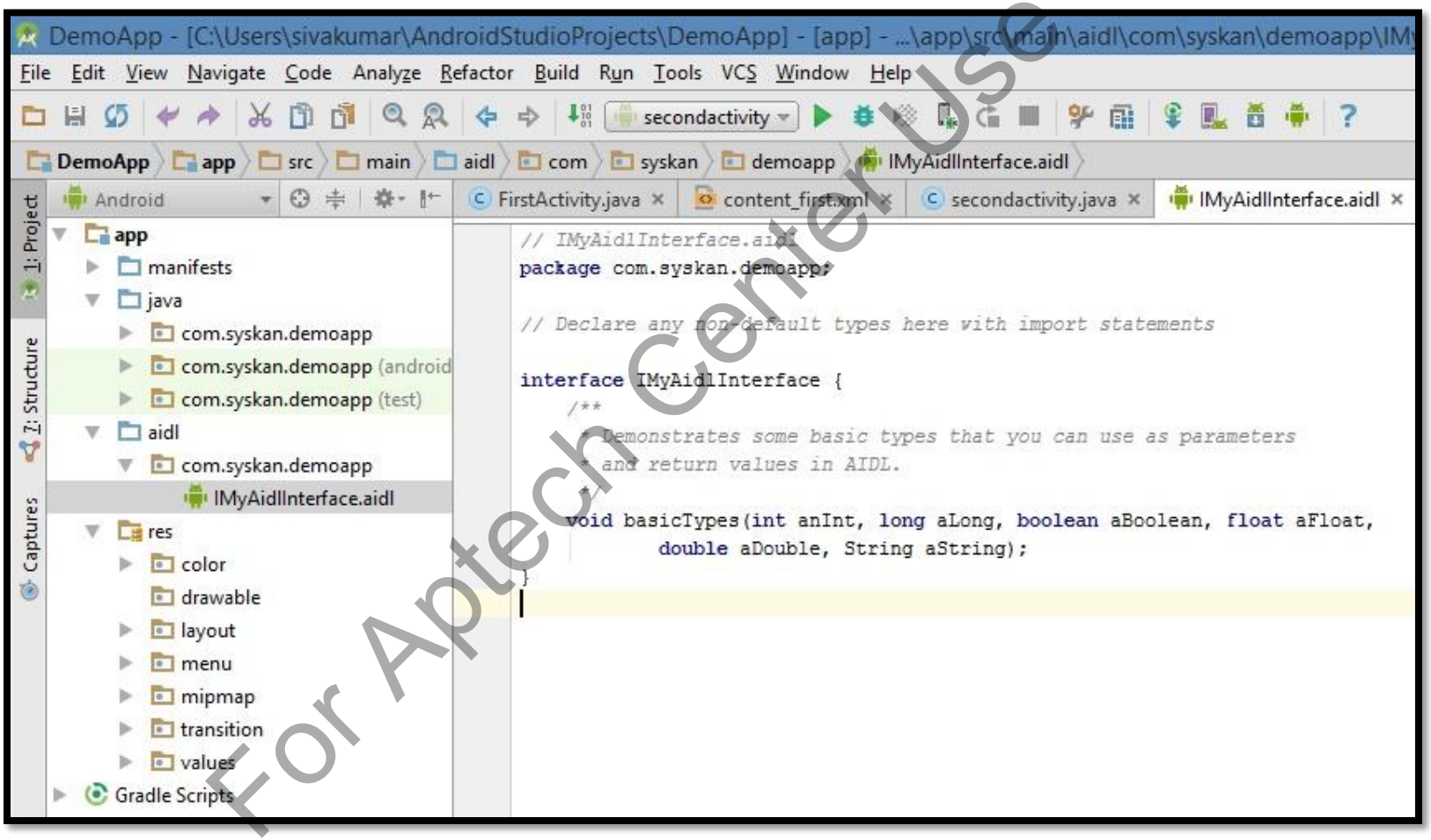
# Create an AIDL file 4-6

- Specify the Name and Target Source Set for the interface and click **Finish**.



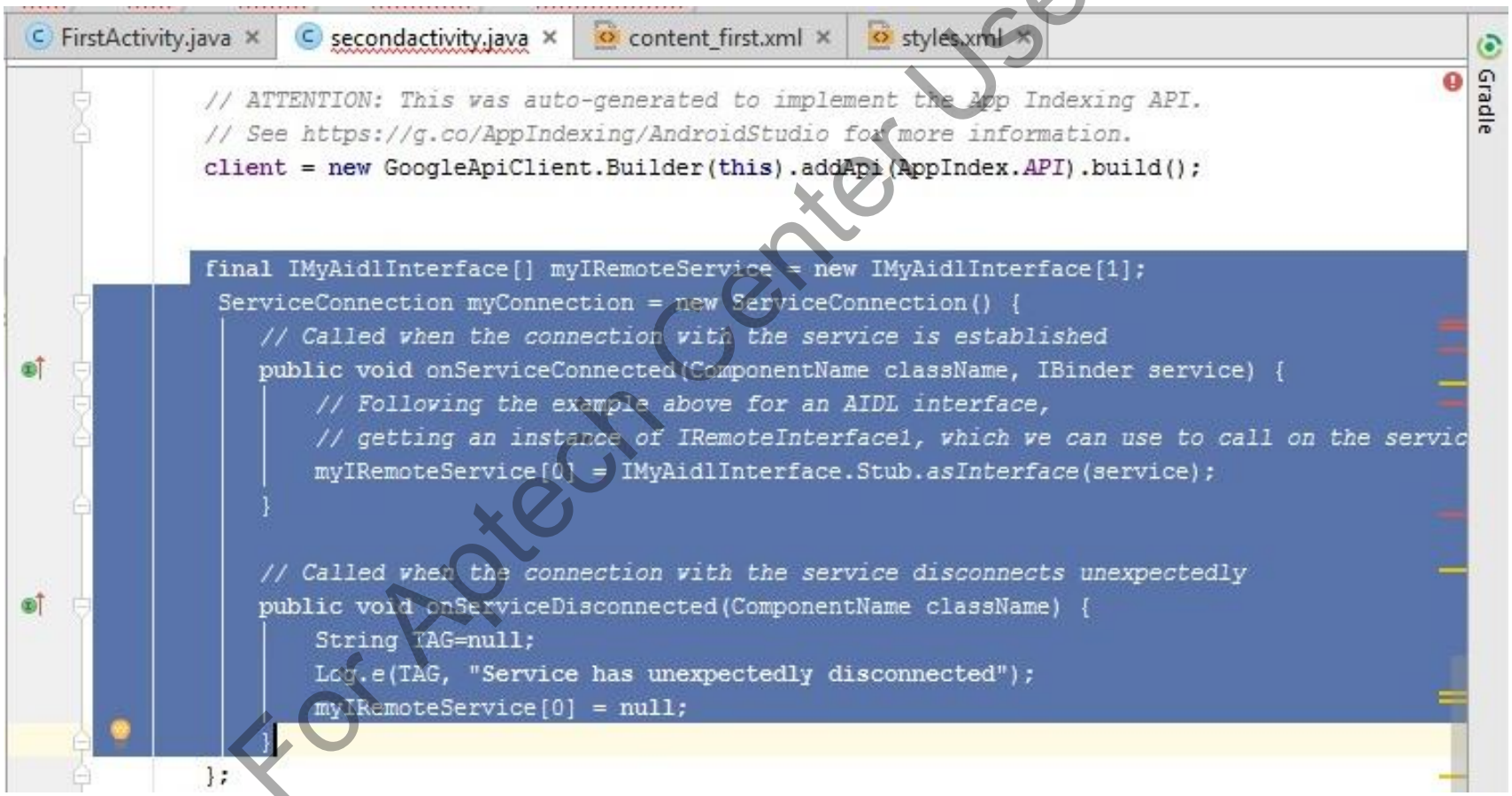
# Create an AIDL file 5-6

- .aidl file with autogenerated code.



# Create an AIDL file 6-6

- Auto generated IBinder interface



# Implement AIDL Interface 1-2

- The interface generated by the .aidl file with .java extension includes a subclass Stub that declares all the methods from the .aidl file.

- The Stub consists of helper methods such as asInterface() which implements the IBinder interface and returns the instance of the interface, which is used to call the RPC methods.

# Implement AIDL Interface 2-2

- Following code snippet shows how to extend the Binder interface and implement the interface using an anonymous interface:

```
private final IMyAidlInterface.Stub myBinder = new
IMyAidlInterface.Stub()
{
       public void basicTypes(int IntegerData, long LongData, boolean
BooleanData,
        float FloatData, double DoubleData, String StringData)
       {
        // Does nothing
        }
};
```

# Expose AIDL Interface to Clients 1-3

- Expose the interface or publish the service to the client, the user needs to inherit Service and implement the Service.onBind () method.
- This method returns an instance of the class that implements the generated Stub.

# Expose AIDL Interface to Clients 2-3

```
import android.app.Service;
. . . // Other import statements
import com.syskan.aidldemo.IMyAidlInterface;
public class MyFirstRemoteService extends Service {
@Override
public void onCreate() {
super.onCreate();
}
@Override
public IBinder onBind(Intent intent) {
// Return the interface
return myBinder;
}
private final IMyAidlInterface.Stub myBinder = new
IMyAidlInterface.Stub() {
public void basicTypes(int IntegerData, long LongData, boolean
BooleanData, float FloatData, double DoubleData, String StringData) {
// Does nothing
} };
}
```

# Expose AIDL Interface to Clients 3-3

```
IMyAidlInterface iMyAidlInterface;
protected ServiceConnection mConnection = new ServiceConnection() {
// Called when the connection with the service is established
public void onServiceConnected(ComponentName className, IBinder
service) {
//Following the example given earlier for an AIDL interface,
// this gets an instance of the IRemoteInterface, which
// we can use to call on the service
iMyAidlInterface = IMyAidlInterface.Stub.asInterface
(service);
}
String TAG;
// Called when the connection with the service disconnects
// unexpectedly
public void onServiceDisconnected(ComponentName className) {
Log.e(TAG, "Service has unexpectedly disconnected");
iMyAidlInterface = null;
}};
```

# Communication Through IPC 1-4

To facilitate communication through IPC and to create a class that supports parcelable protocol:

1. Implement the Parcelable interface in the class.
2. Write the current state of the object to Parcel, implement writetoParcel.
3. Implement the Parcelable.Creator interface, use a static field CREATOR in the class.
4. Ensure that the .aidl file that is created, declares the Parcelable class.

# Communication Through IPC 2-4

- Following code snippet shows how to create a parcelable class in the existing or a new .aidl file:

```
// Declare Triangle so AIDL can find it & knows that it implements
// the parcelable protocol.
parcelable Triangle;
```

# Communication Through IPC 3-4

```
import android.os.Parcel;
import android.os.Parcelable;
public final class Triangle implements Parcelable {
public int leftpos;
public int toppos;
public int rightpos;
public static final Parcelable.Creator<Triangle> CREATOR = new
Parcelable.Creator<Triangle>() {
public Triangle createFromParcel(Parcel in) {
return new Triangle(in); }
public Triangle[] newArray(int size) { return new
Triangle[size];
}
};
```

# Communication Through IPC 4-4

```
public Triangle() { }
private Triangle(Parcel in) {
readFromParcel(in);
}
public void writeToParcel(Parcel out) {
out.writeInt(leftpos);
out.writeInt(toppos);
out.writeInt(rightpos);
}
public void readFromParcel(Parcel in) {
leftpos = in.readInt();
toppos = in.readInt();
rightpos = in.readInt();
}
}
```

# IPC Method Invocation 1-5

- The IPC method is invoked when the calling class performs the following steps:
  1. It copies the .aidl file to the src directory of the project.
  2. It creates an instance of the `IBinder` interface.
  3. It implements `ServiceConnection`.
  4. It calls `context.bindservice ()` in the `ServiceConnection` implementation.
  5. It calls `IMyAidlInterface.Stub.asInterface (IBinder)` service to cast the returned parameter to `IMyAidlInterface` type.
  6. It calls the methods defined in the interface.
  7. It calls the `Context.unbindService ()` method with the instance of the service to disconnect.

# IPC Method Invocation 2-5

```
import android.content.ComponentName;
import android.content.Intent;
import android.content.ServiceConnection;
import android.os.Bundle;
import android.os.IBinder;
import android.os.RemoteException;
import android.support.v7.app.AppCompatActivity;
import android.widget.TextView;
public class MyFirstBinding extends AppCompatActivity
private IMyAidlInterface mService;
private TextView mLog;
@Override
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mLog = (TextView) findViewById(R.id.log);
```

# IPC Method Invocation 3-5

```
    Intent serviceIntent = new Intent()
    .setComponent(new ComponentName(
    "com.syskan.aidlexamplereceiver",
    "com.syskan.aidlexamplereceiver.MainService"));
    mLog.setText("Starting service…\n");
    startService(serviceIntent);
    mLog.append("Binding service…\n");
    bindService(serviceIntent, mConnection, BIND_AUTO_CREATE);
}
private ServiceConnection mConnection = new ServiceConnection()
{
    @Override
    public void onServiceConnected(ComponentName className, IBinder
    service)
    {
        mLog.append("Service binded!\n");
        mService = IMyAidlInterface.Stub.asInterface(service);
        performListing();
    }
```

# IPC Method Invocation 3-5

```
    @Override
    public void onServiceDisconnected(ComponentName className)
    {
        mService = null;
        // This method is only invoked when the service quits from the //
        other end or gets killed
        // Invoking exit() from the AIDL interface makes the Service
        // kill itself, thus invoking this.
        mLog.append("Service disconnected.\n");
}};
private void performListing()
{
    mLog.append("Requesting file listing…\n");
    long start = System.currentTimeMillis();
    long end = 0;
```

# IPC Method Invocation 4-5

```
try
{
    MainObject[] results = mService.listFiles("/sdcard/testing");
    end = System.currentTimeMillis();
    int index = 0;
    mLog.append("Received " + results.length + " results:\n");
    for (MainObject o : results)
    {
        if (index > 20)
        {
            mLog.append("\t -> Response truncated!\n");
            break;
        }
        mLog.append("\t -> " + o.getPath() + "\n");
        index++;
    }
}
```

# IPC Method Invocation 5-5

```
    catch (RemoteException e)
    {
        e.printStackTrace();
    }
    mLog.append("File listing took " + (((double) end - (double) start) /
    1000d) + " seconds, or " + (end - start) + " milliseconds.\n");
    try
    {
        mService.exit();
    }
    catch (RemoteException e)
    {
        e.printStackTrace();
    }
  }
}
```

# Summary

- Android Interface Definition Language allows communication between the client and service provider through IPC.
- AIDL interface is defined in the .aidl file using Java programming language.
- .aidl file defines the programming interface that includes method signatures.
- The programming interface contains an inner abstract class Stub, that implements methods from the AIDL interface.
- To publish the service to the clients, expose the interface to the clients.
- IPC interface allows a parcelable class to be sent from one process to another.
- The calling class must follow certain steps to call the remote AIDL interface.