SAMSUNG S PEN SDK

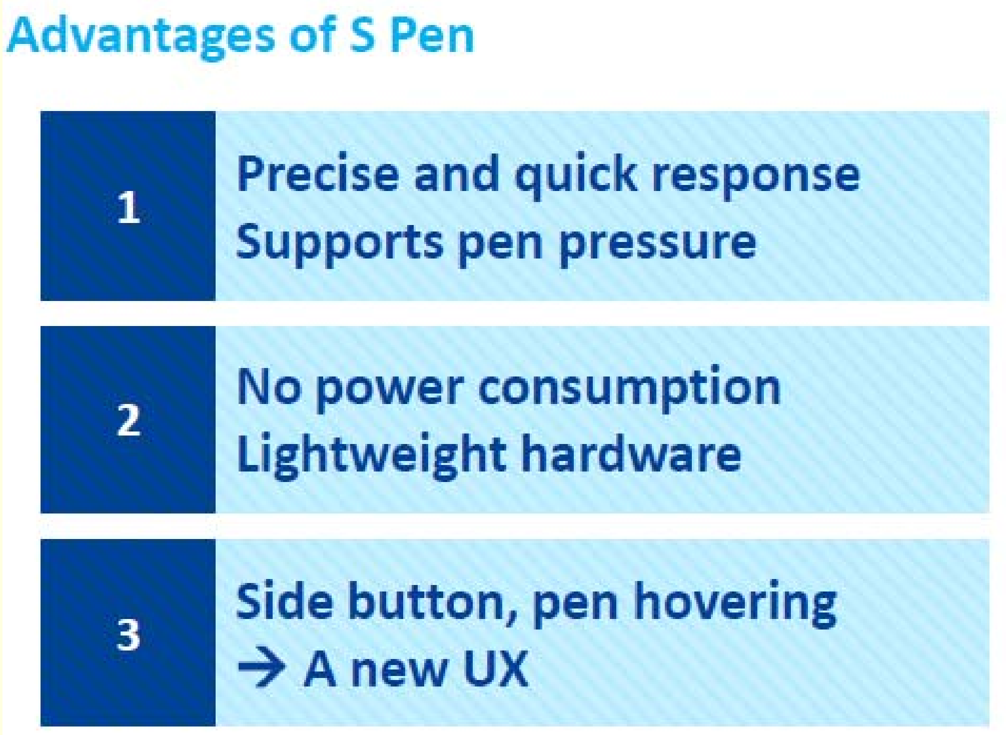
Instructor: Nguyễn Kiêm Hùng

WHAT IS PEN SDK

- **S PEN is a new type of input device available on Note range of devices.**
- **Requirements:**
  - Devices with Android 4.0 Ice Cream Sandwich (API level 14) or higher support Pen.
  - Between minimum resolution of 480x800 and maximum resolution of 1600x2580 are supported.
  - Some functions such as writing pressure or hover can be limited if Pen is not used.

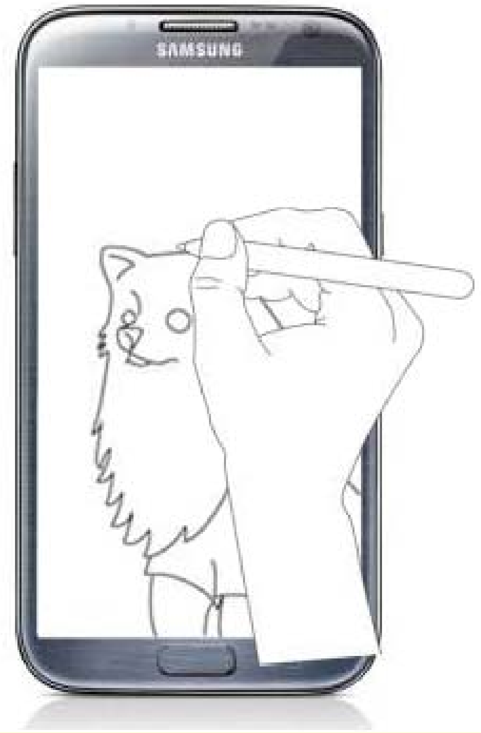
WHAT IS PEN SDK

- allows you to develop applications, such as S Note, that use handwritten input
- uses a pen, finger or other kinds of virtual pens for input



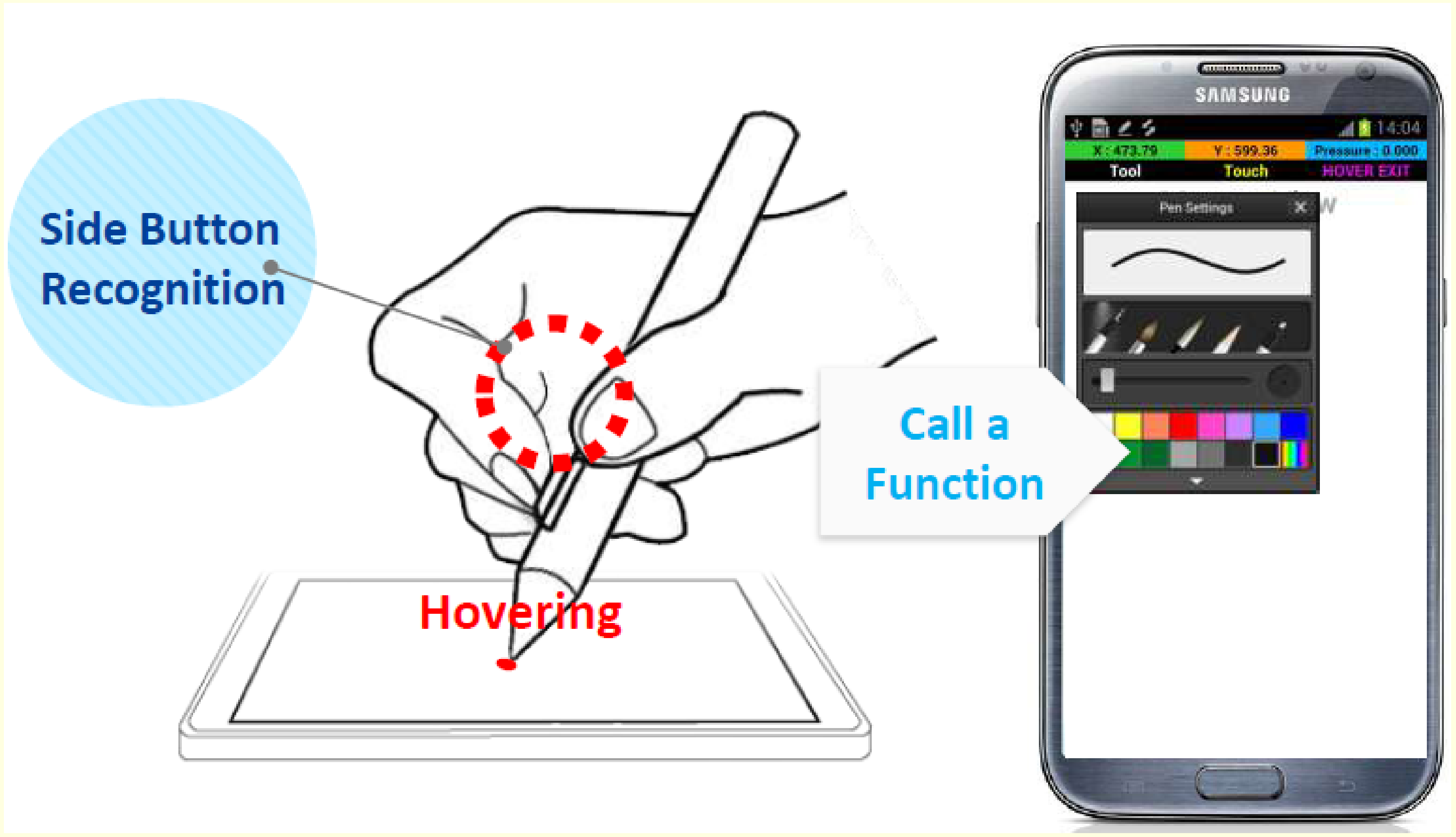
Advantages of S Pen

1	Precise and quick response Supports pen pressure
2	No power consumption Lightweight hardware
3	Side button, pen hovering → A new UX



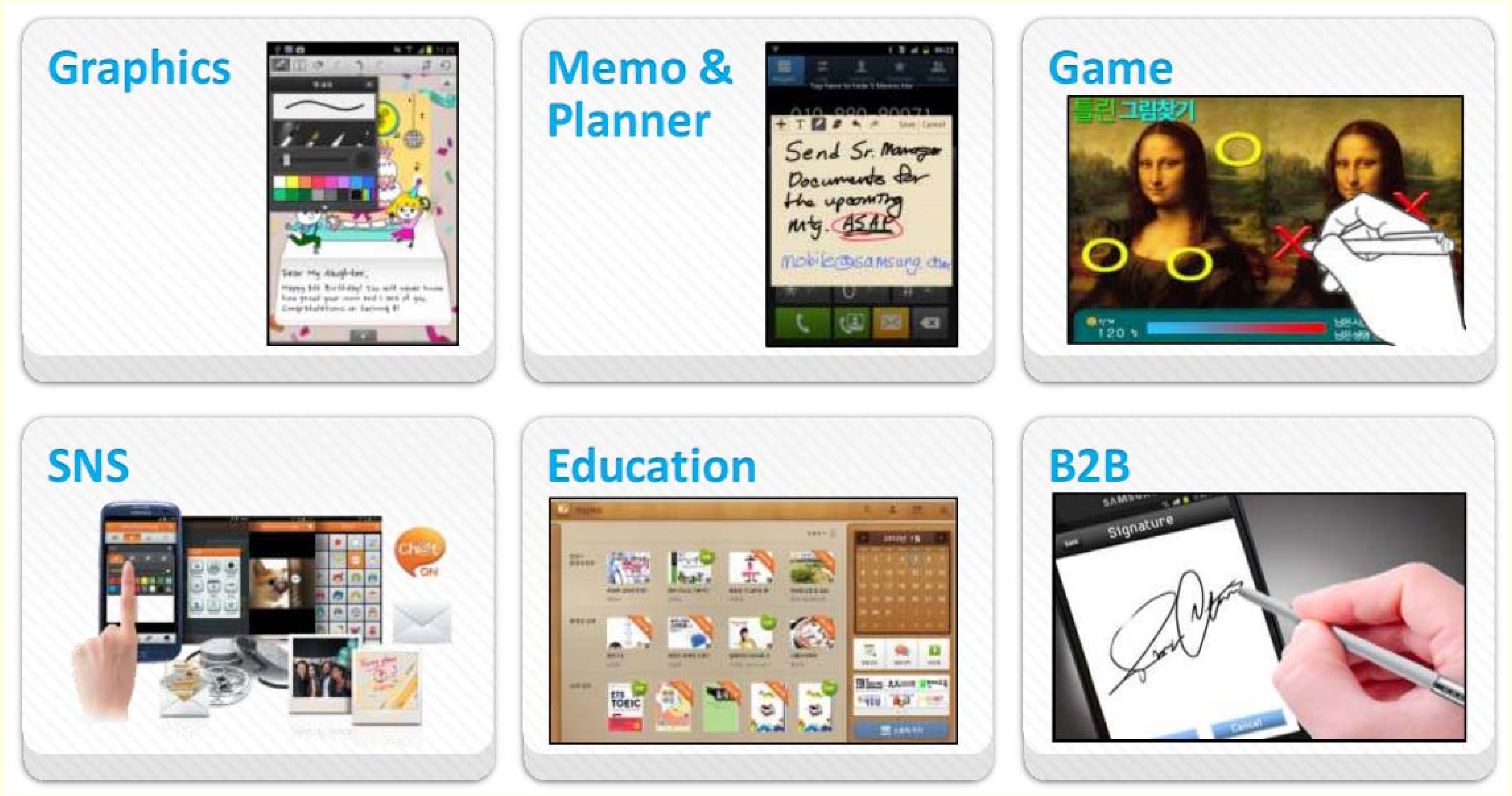
WHAT IS PEN SDK

- **What's the Benefit?**
  - Recognize the side button with or without contact.



WHAT IS PEN SDK

- **Some use cases**

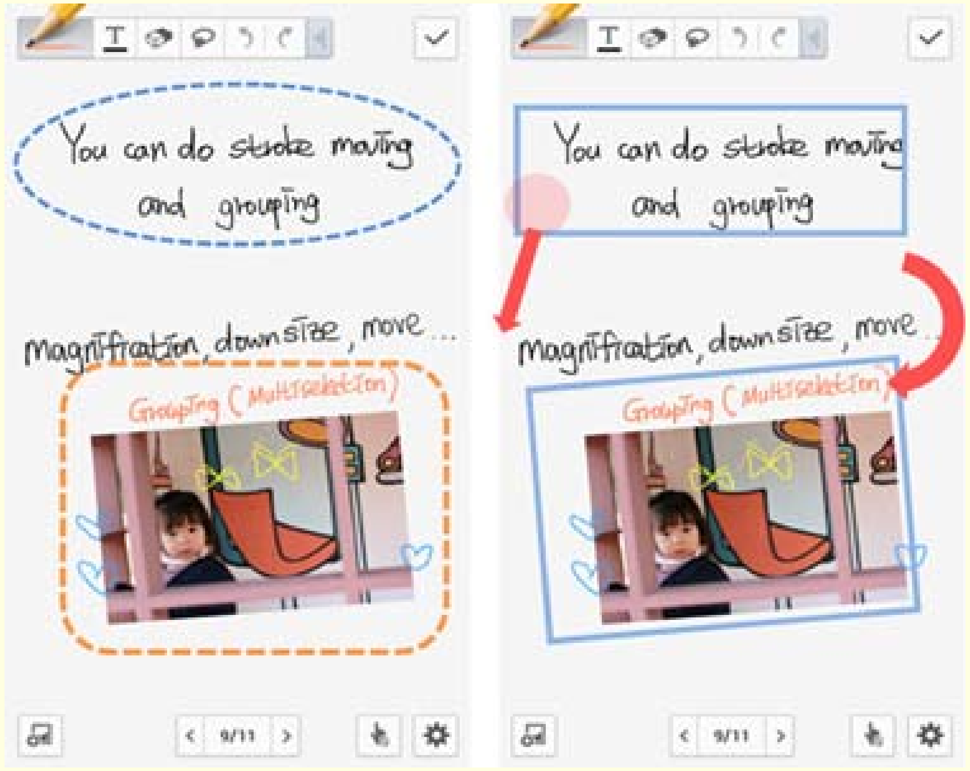


WHAT IS PEN SDK

- **Pen supports the following features:**
  - **Advanced edit**
  - **Recognition**

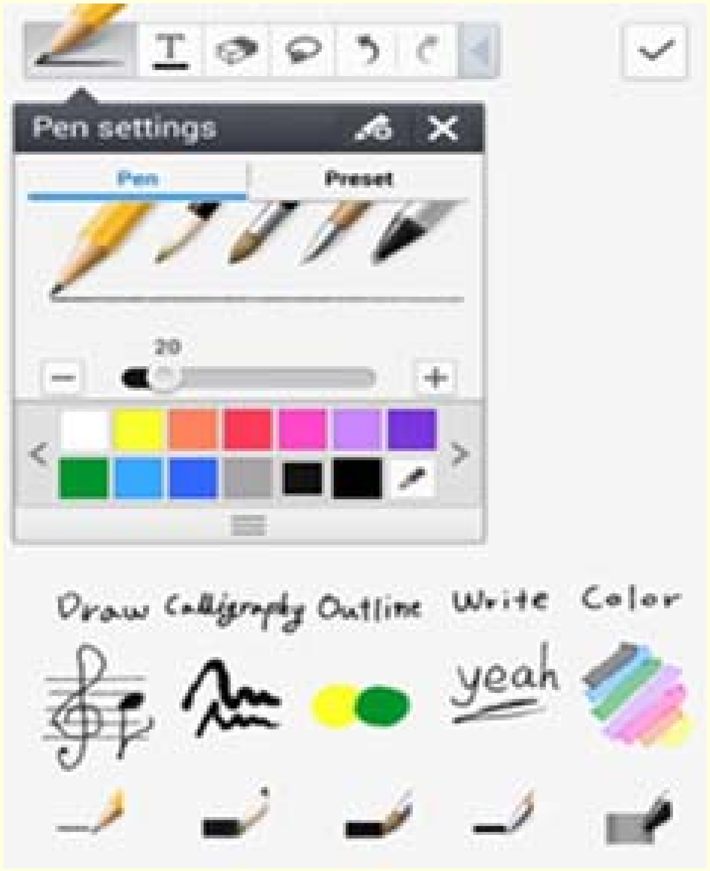
WHAT IS PEN SDK

- **Advanced edit:**
  - a drawing engine based on Stroke
  - offers advanced edit functions, such as selection, multiple selection, group/ungroup, move forward/backward, and zoom in/out


You can do stroke moving and grouping
magnification, downsize, move ...
Grouping (Multiselection)
You can do stroke moving and grouping
magnification, downsize, move ...
Grouping (Multiselection)

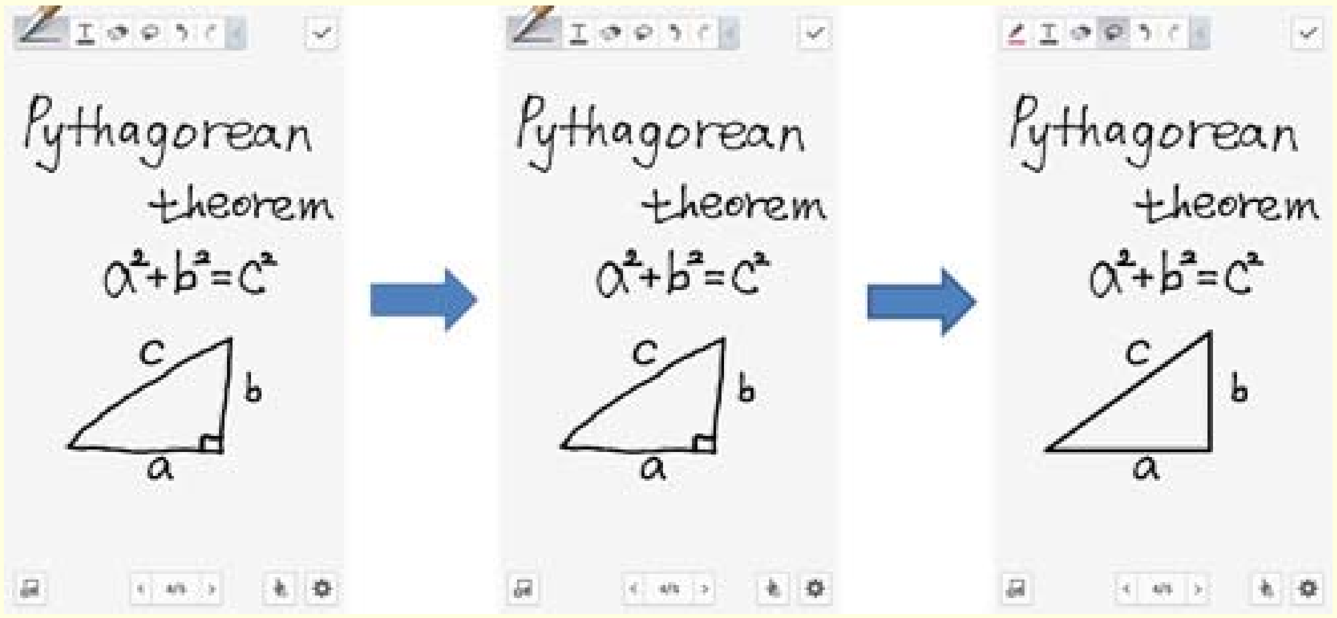
WHAT IS PEN SDK

- **Advanced edit:**
  - **Pen: Various tools such as brushes, color pens and more**



WHAT IS PEN SDK

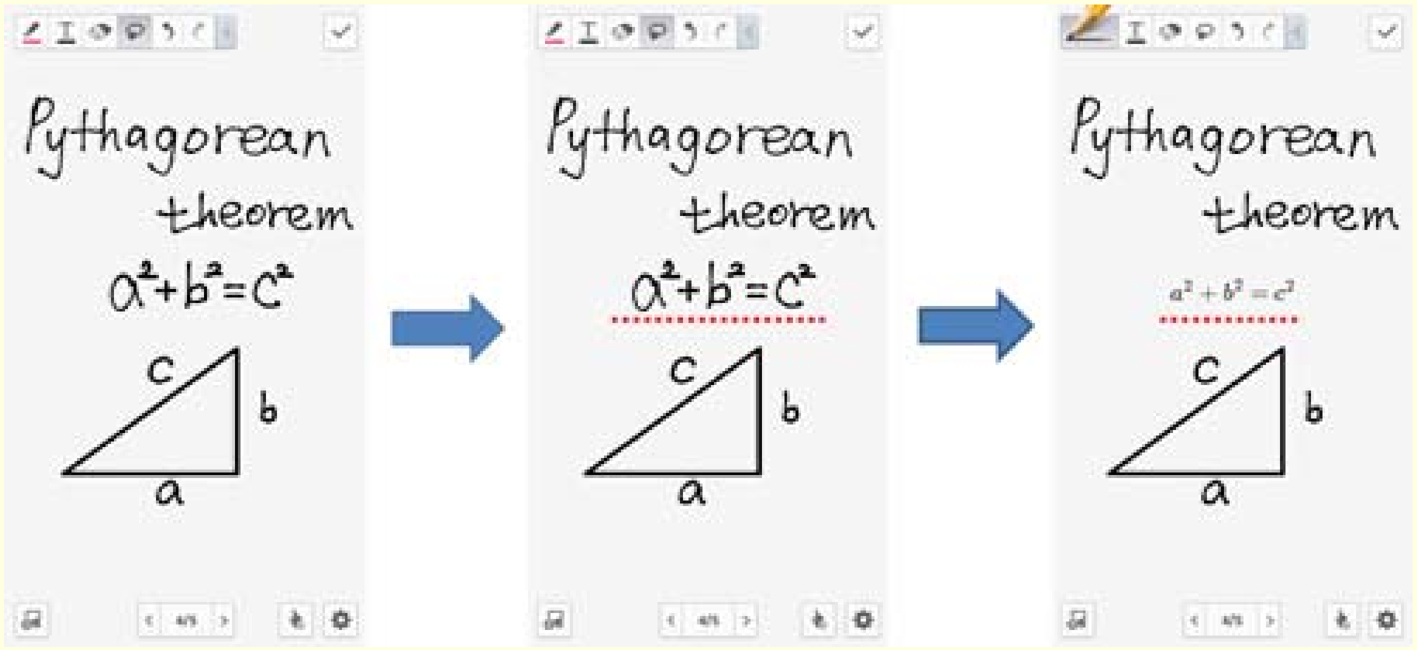
- **Recognition:**
  - to convert objects written on the canvas to drawings



Draw a triangle and select it so that the system recognizes it as an object consisting of lines

WHAT IS PEN SDK

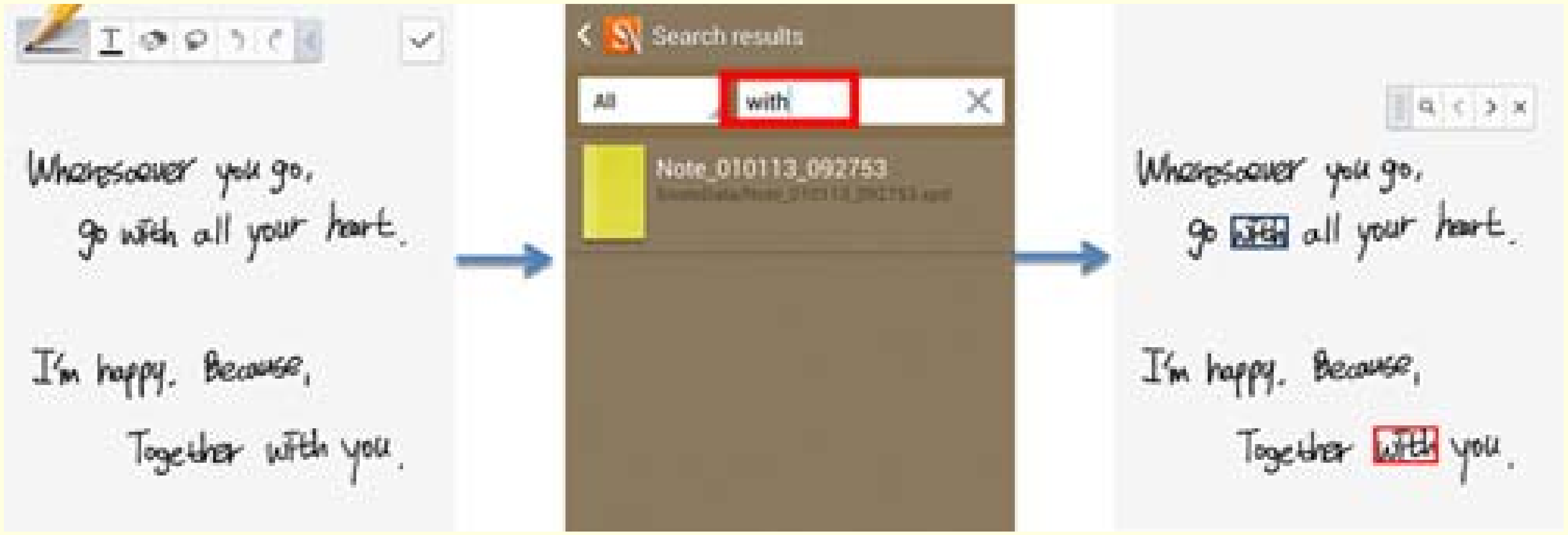
- **Recognition:**
  - to convert objects written on the canvas to drawings



Write a formula that is converted the image into a digital format.

WHAT IS PEN SDK

- **Recognition:**
  - to convert objects written on the canvas to drawings



Handwritten Text Search

USING PEN SDK

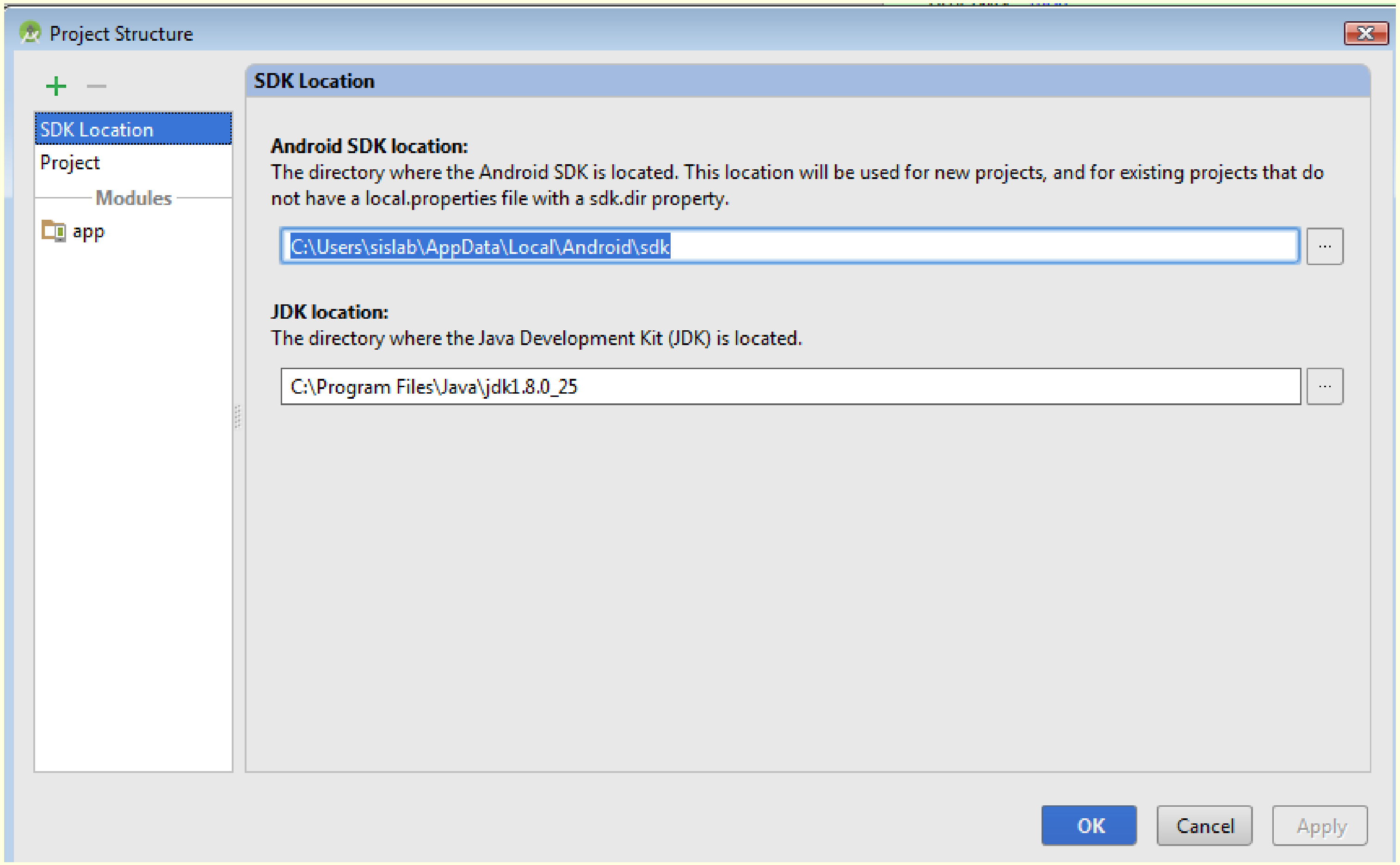
- **Samsung Device Requirements:**
  - S-Pen support
  - Android 4.0 Ice Cream Sandwich (API level 14)
  - S-Pen SDK package

USING PEN 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 PEN SDK with the Android SDK Manager**

USING PEN SDK

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



SDK Location

Android SDK location:
The directory where the Android SDK is located. This location will be used for new projects, and for existing projects that do not have a local.properties file with a sdk.dir property.

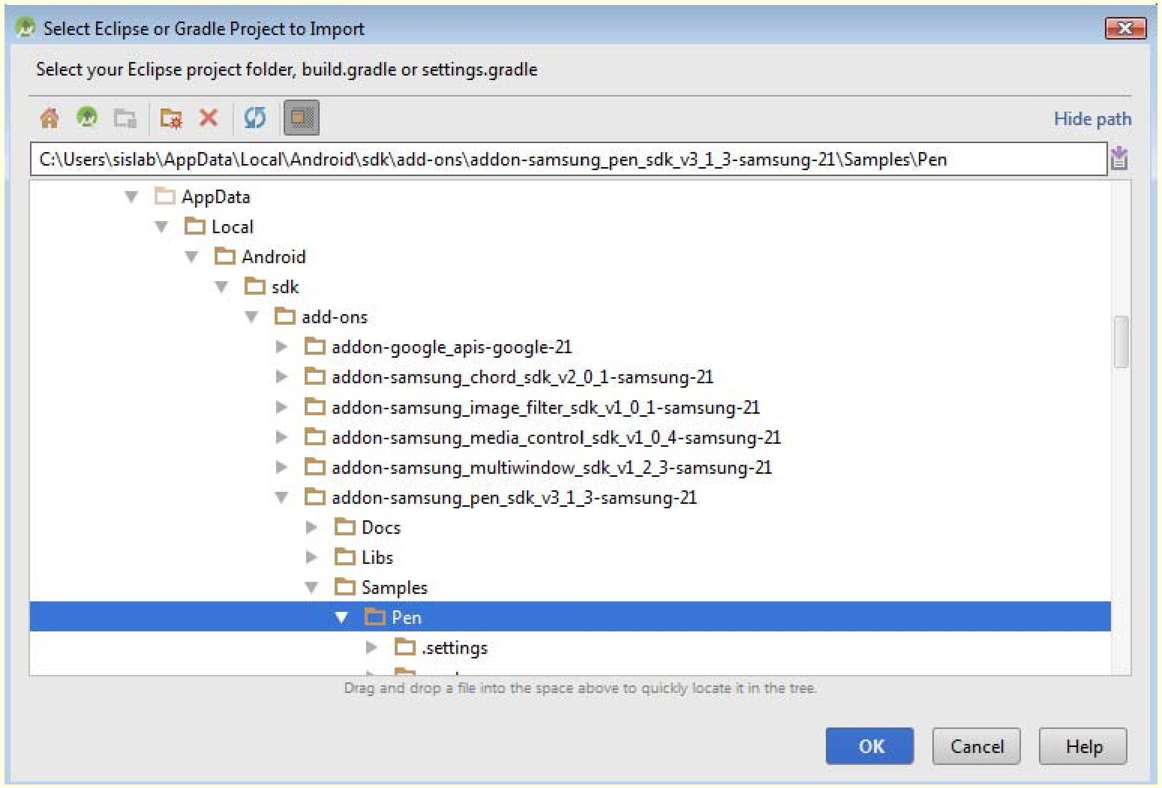
C:\Users\sislab\AppData\Local\Android\sdk

JDK location:
The directory where the Java Development Kit (JDK) is located.

C:\Program Files\Java\jdk1.8.0_25

RUNNING SAMPLE APPLICATION

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



Select Eclipse or Gradle Project to Import

Select your Eclipse project folder, build.gradle or settings.gradle

Hide path

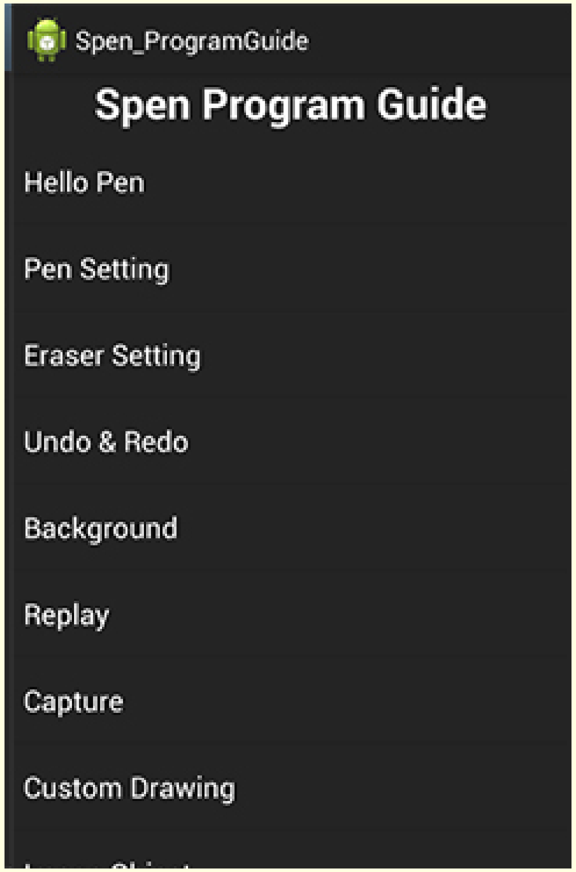
C:\Users\sislab\AppData\Local\Android\sdk\add-ons\addon-samsung_pen_sdk_v3_1_3-samsung-21\Samples\Pen

```
▼ AppData
  ▼ Local
    ▼ Android
      ▼ sdk
        ▼ add-ons
          ▶ addon-google_apis-google-21
          ▶ addon-samsung_chord_sdk_v2_0_1-samsung-21
          ▶ addon-samsung_image_filter_sdk_v1_0_1-samsung-21
          ▶ addon-samsung_media_control_sdk_v1_0_4-samsung-21
          ▶ addon-samsung_multiwindow_sdk_v1_2_3-samsung-21
          ▼ addon-samsung_pen_sdk_v3_1_3-samsung-21
            ▶ Docs
            ▶ Libs
            ▼ Samples
              ▼ Pen
                ▶ .settings
```

Drag and drop a file into the space above to quickly locate it in the tree.

RUNNING SAMPLE APPLICATION

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



PEN PROGRAMING GUIDE

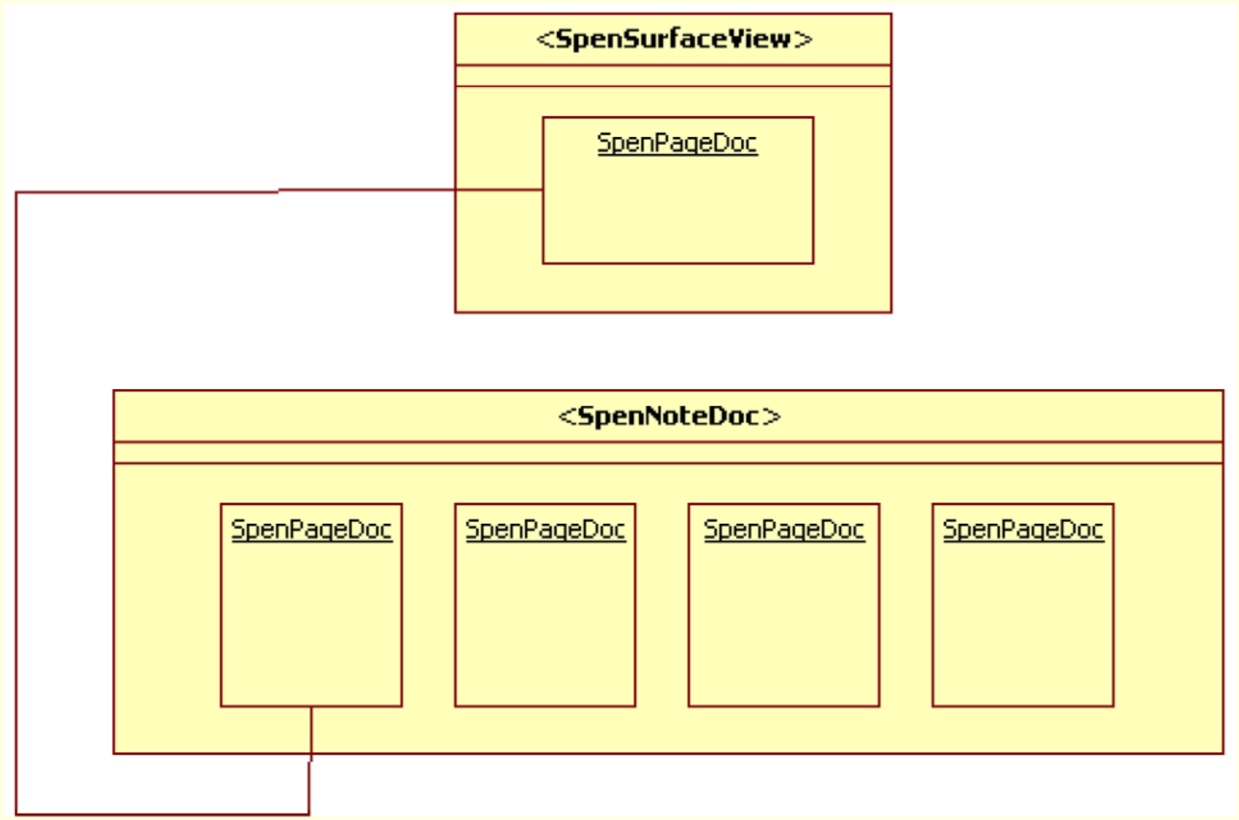
- **PEN SDK** allows you to develop applications that use handwritten inputs: S pen, finger, etc.
- **PEN SDK** provides functions for verifying if the S pen is activated, identifying event coordinates, sensing the pressure, verifying if the side button is pressed, processing hover events and more for your application.
- **You can use Pen to:**
  - draw using a **finger** and/or **S pen**
  - set **user preferences** for pens, erasers, and text
  - edit and save input objects (text, strokes, and images) as a file
  - manage history for undo and redo commands

PEN PROGRAMING GUIDE

- **The Pen motion events include:**
  - Touch events occur when a S pen touches the screen
  - Hover events occur when a S pen is within a certain range of the screen.

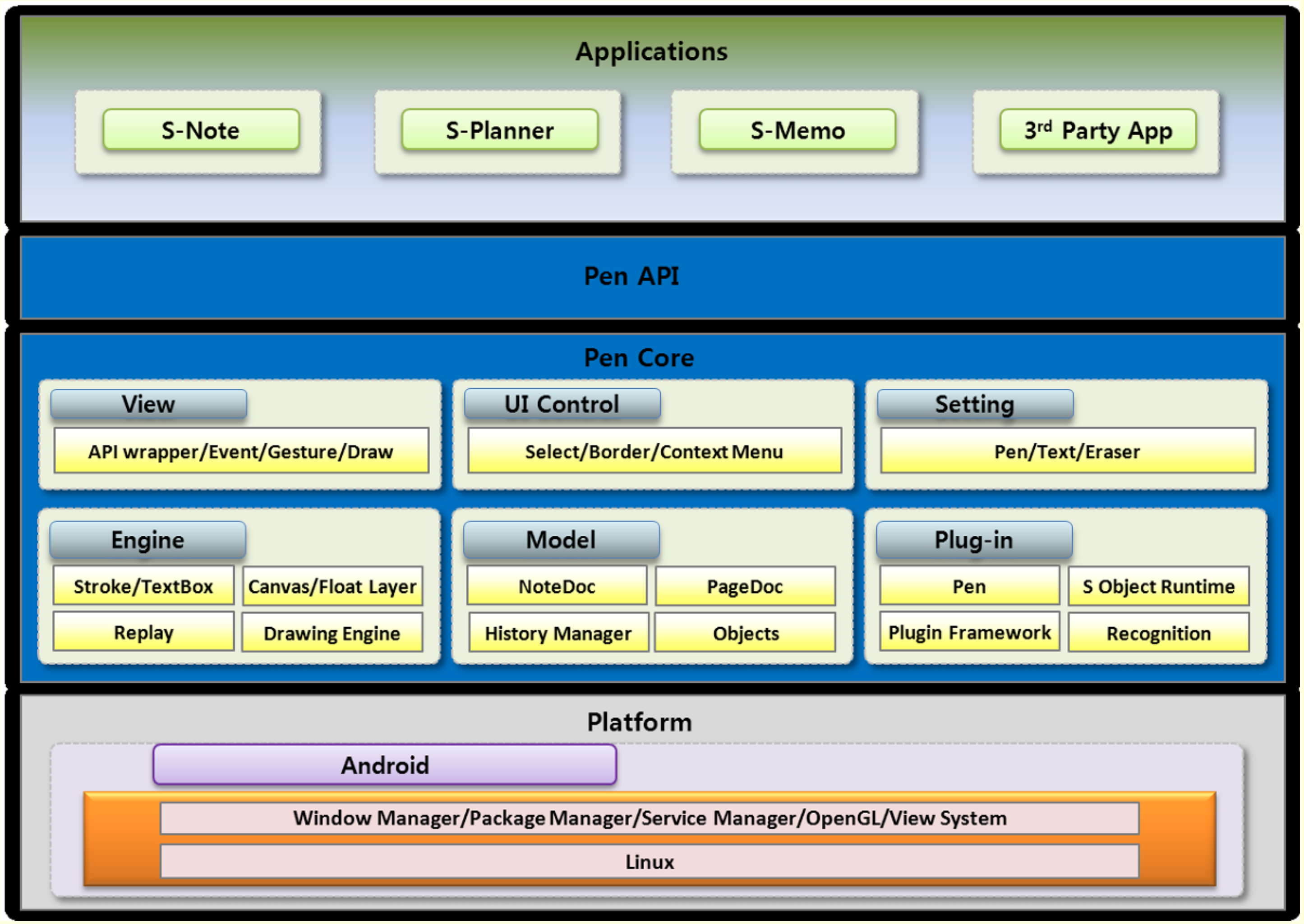
PEN PROGRAMING GUIDE

- **SpenSurfaceView** class processes finger and S pen inputs to express data on the viewport.
- Objects drawn on an **SpenSurfaceView** instance are saved in **SpenPageDoc**, with multiple SpenPageDocs making an **SpenNoteDoc** file.



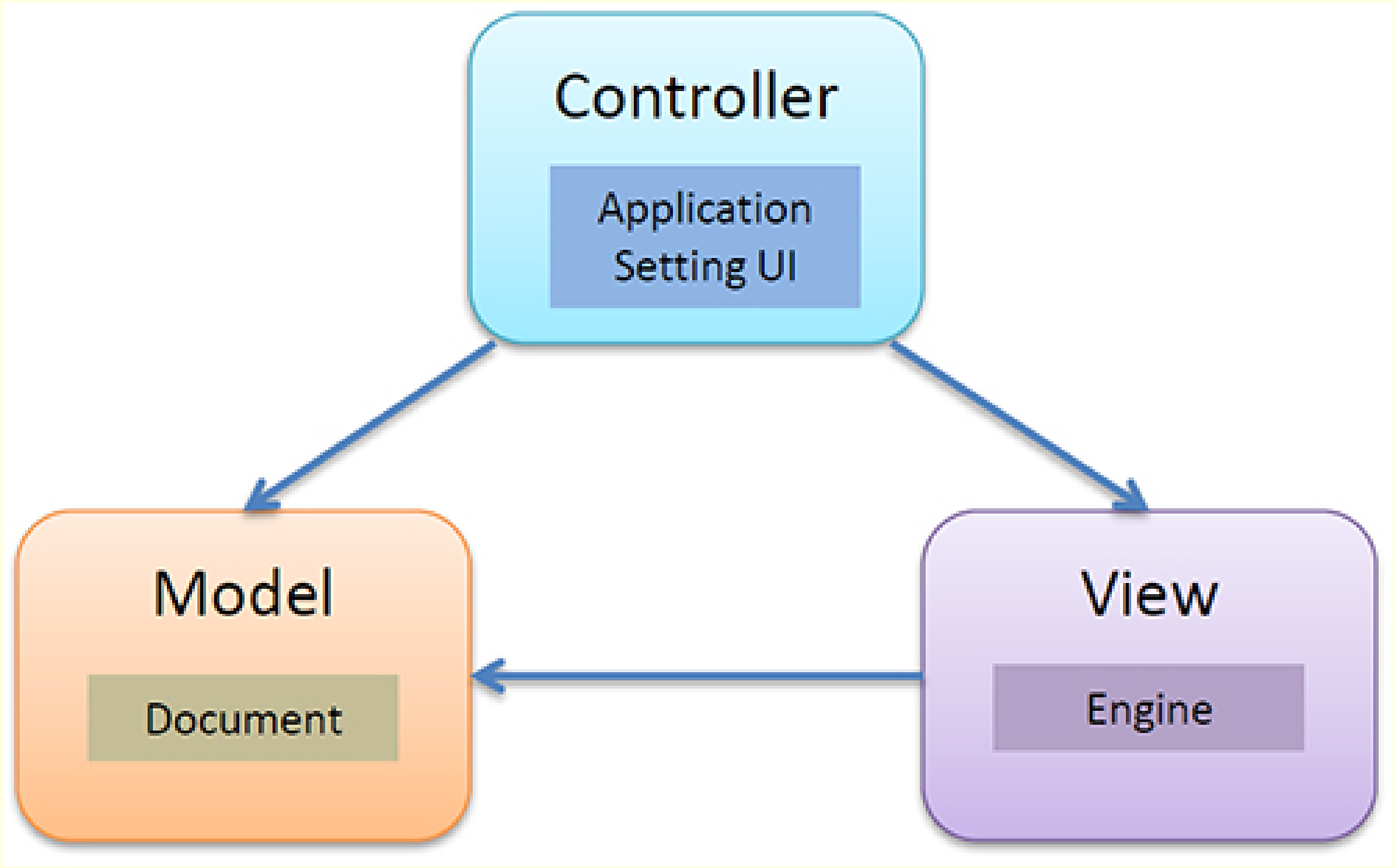
PEN PROGRAMING GUIDE

- **PEN SDK Architecture**



PEN PROGRAMING GUIDE

- Model-View-Control paradigm of S Pen 3.0 Architecture



PEN PROGRAMING GUIDE

- **Components**
  - Pen-v3.1.3.jar
  - Sdk-v1.0.0.jar

PEN PROGRAMING GUIDE

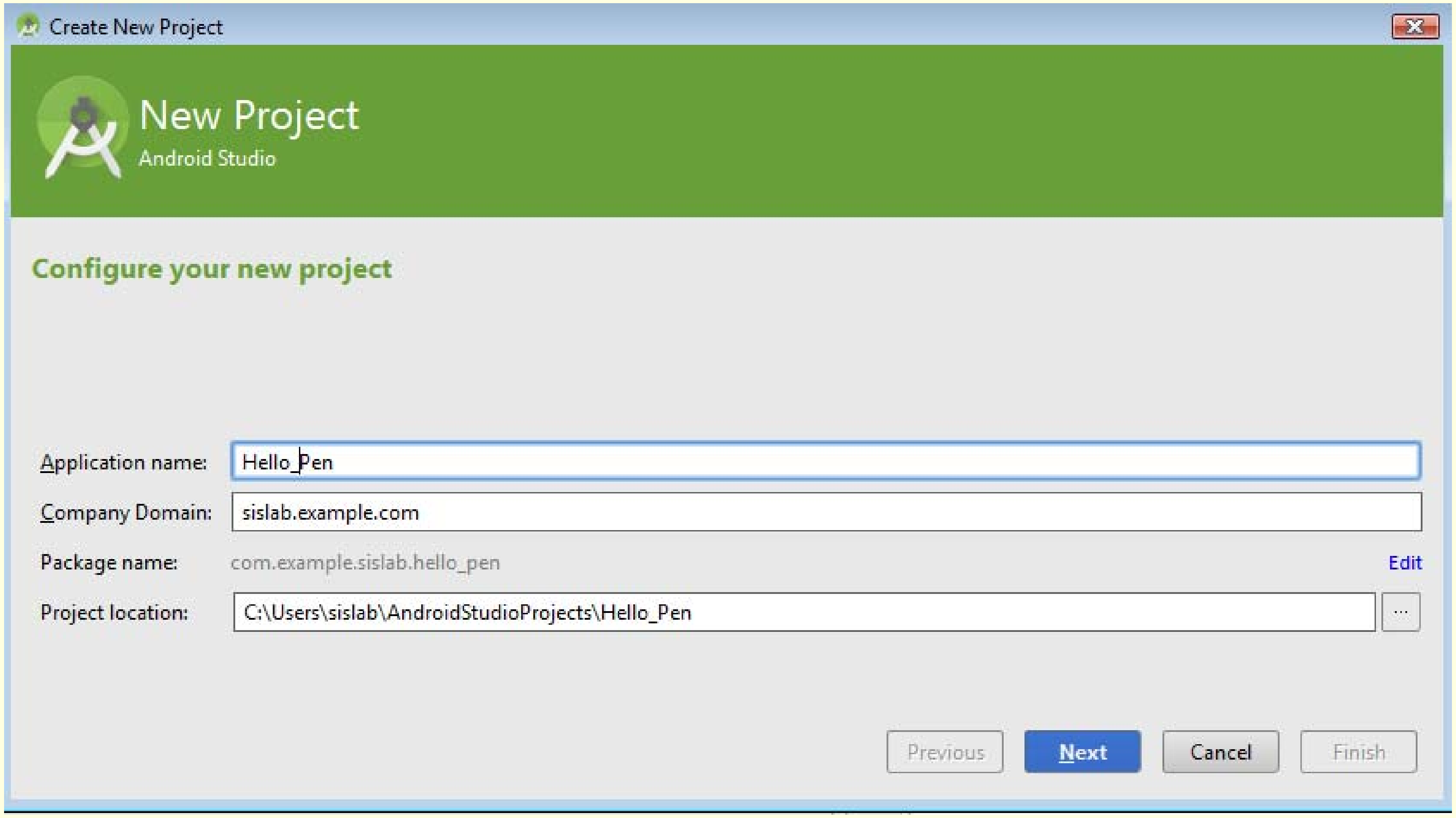
- **Implementation**
  - **Step 1: Initializing** the instance of the **Spen** class by calling the **initialize()** method requiring a context as parameter.
  - **Step 2: Creating an S Pen Surface View**
    - Simply create an object of the SpenSurfaceView class then set the object as parameter to the Relative layout addView() method.
  - **Step 3: Adding the Pen Setting View**
    - To add a Pen Setting View, add the same instance of the Relative layout used in creating the S pen surface view to the instance of the SpenSettingPenLayout through its constructor method requiring it as a parameter.
  - **Step 4: Adding the Eraser Setting View**
    - To add a Pen Setting View, add the same instance of the Relative layout used in creating the S pen surface view to the instance of the SpenSettingEraserLayout through its constructor method requiring it as a parameter.

FIRT PROGRAM: Hello Pen

- **Hello Pen is a simple program that:**
  - gets input from a S pen
  - expresses drawing on the viewport

FIRT PROGRAM: Hello Pen

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



Configure your new project

Application name: Hello_Pen

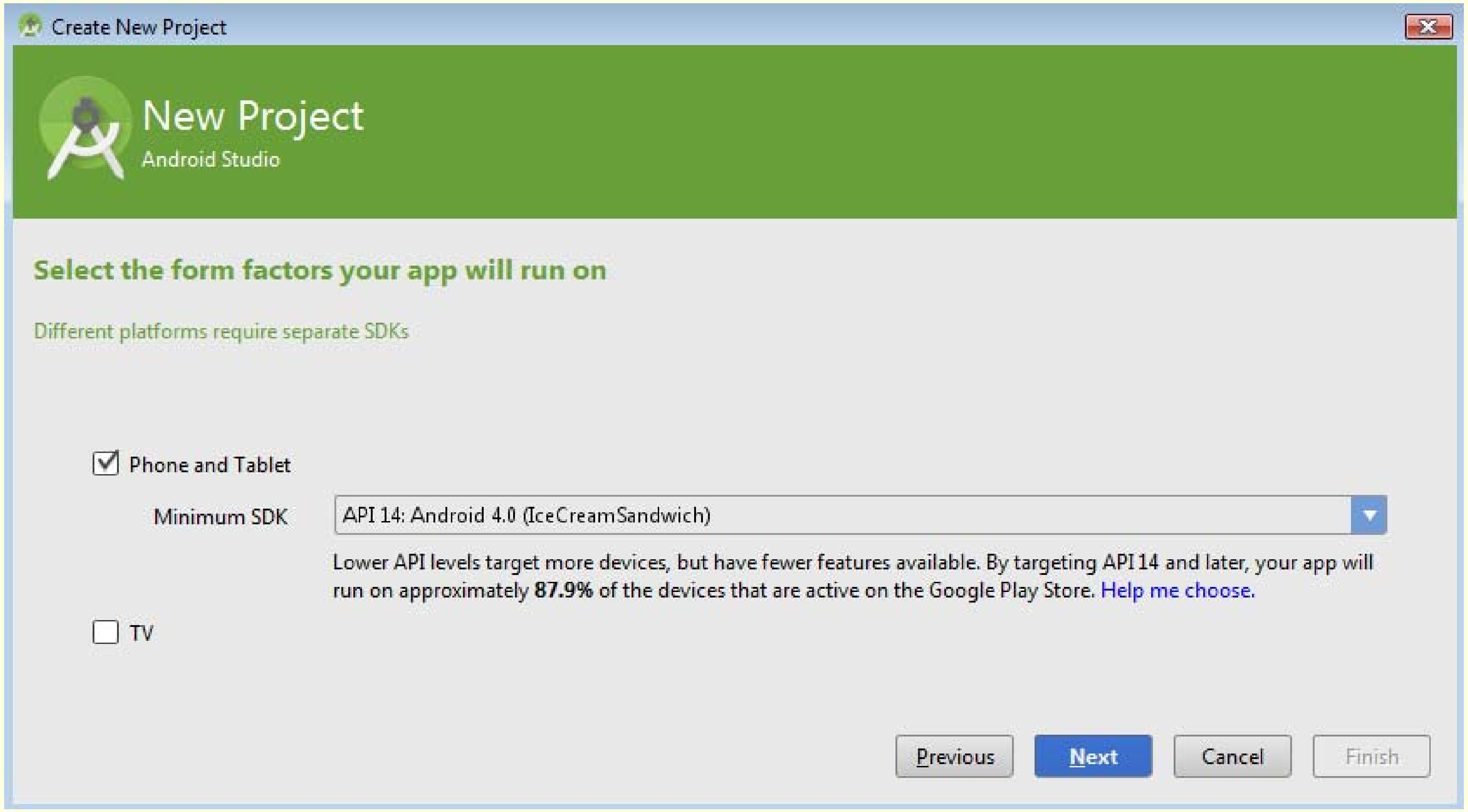
Company Domain: sislab.example.com

Package name: com.example.sislab.hello_pen

Project location: C:\Users\sislab\AndroidStudioProjects\Hello_Pen

FIRT PROGRAM: Hello Pen

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



Select the form factors your app will run on

Different platforms require separate SDKs

☑ Phone and Tablet

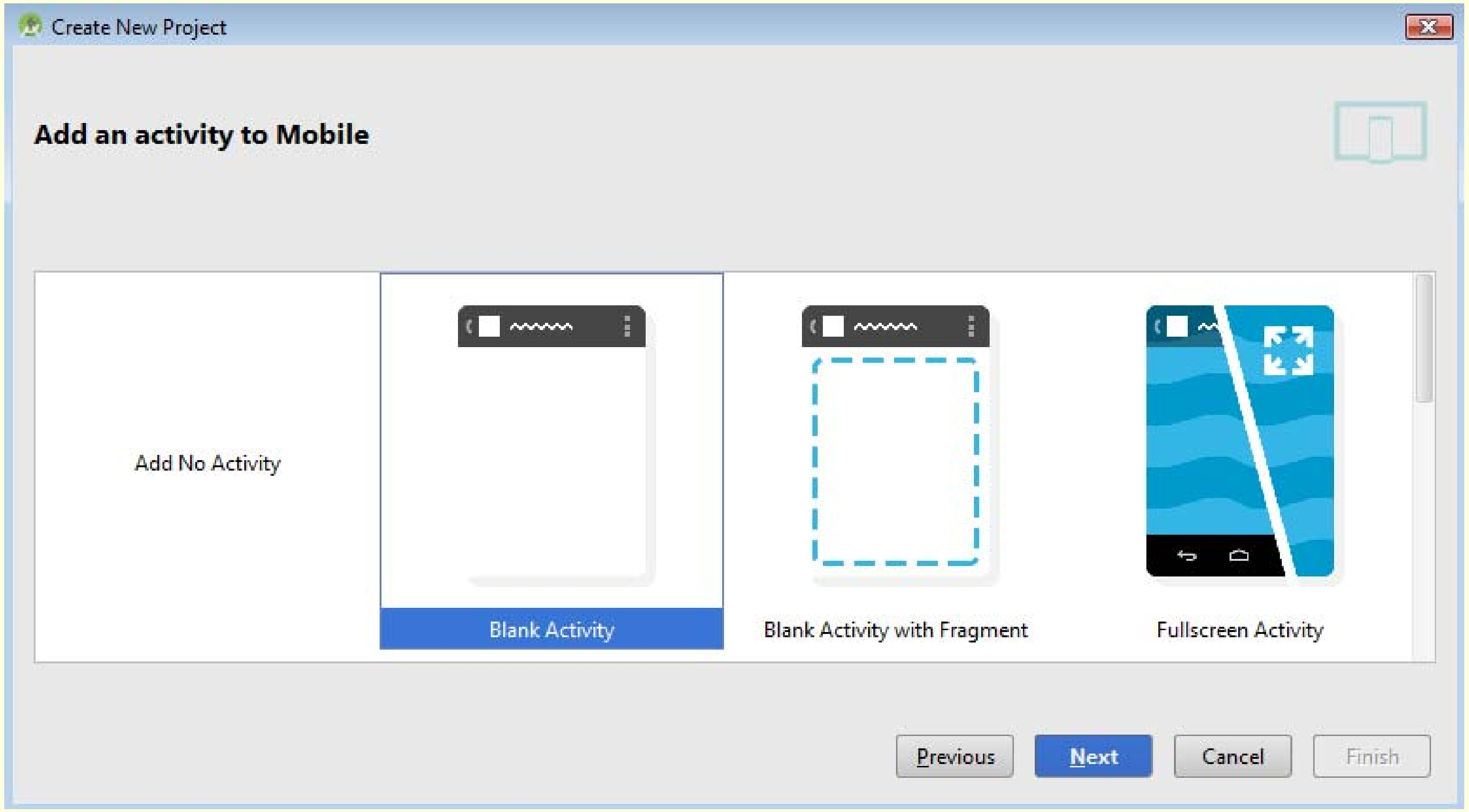
Minimum SDK: API 14: Android 4.0 (IceCreamSandwich)

Lower API levels target more devices, but have fewer features available. By targeting API 14 and later, your app will run on approximately **87.9%** of the devices that are active on the Google Play Store. Help me choose.

☐ TV

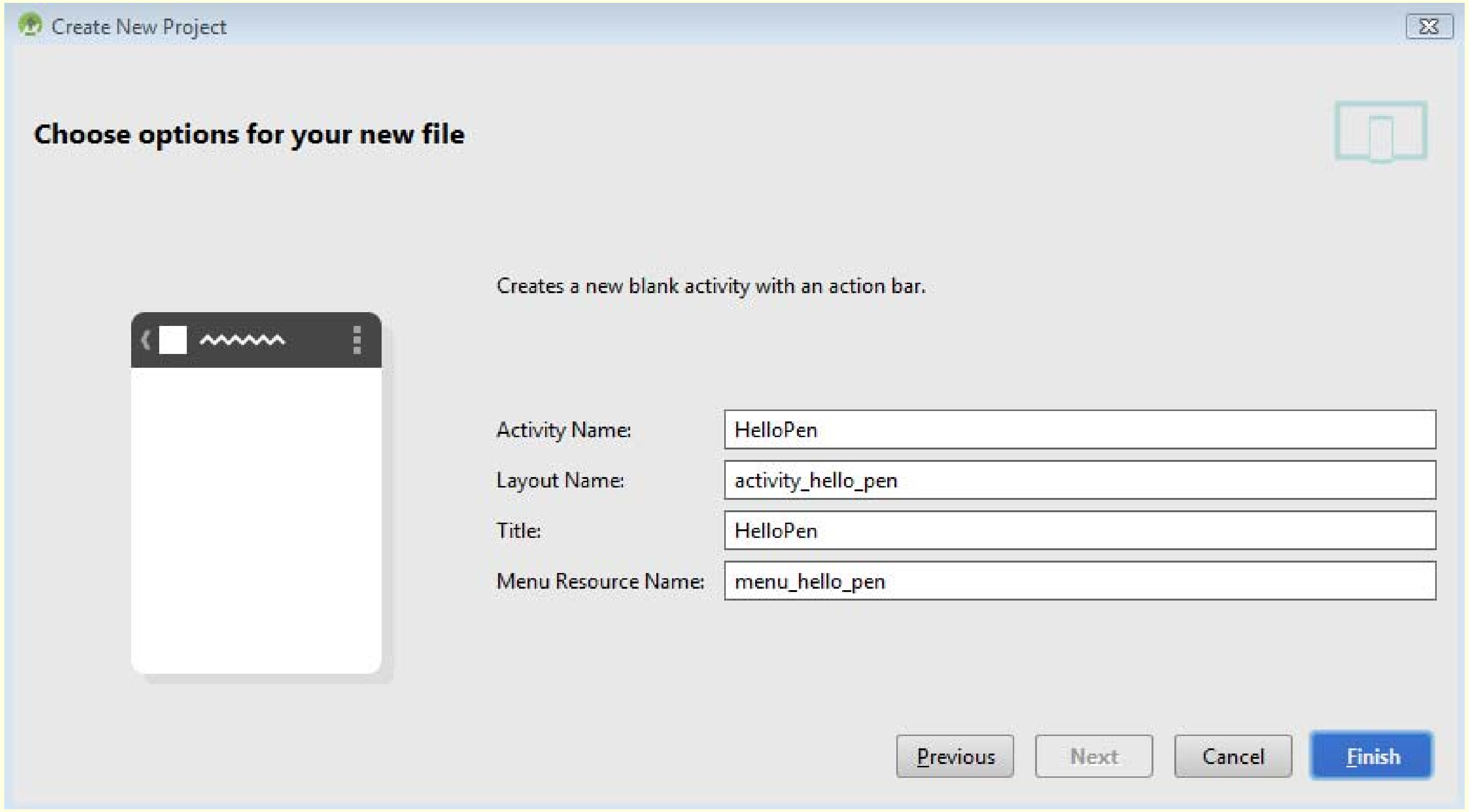
FIRT PROGRAM: Hello Pen

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



FIRT PROGRAM: Hello Pen

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



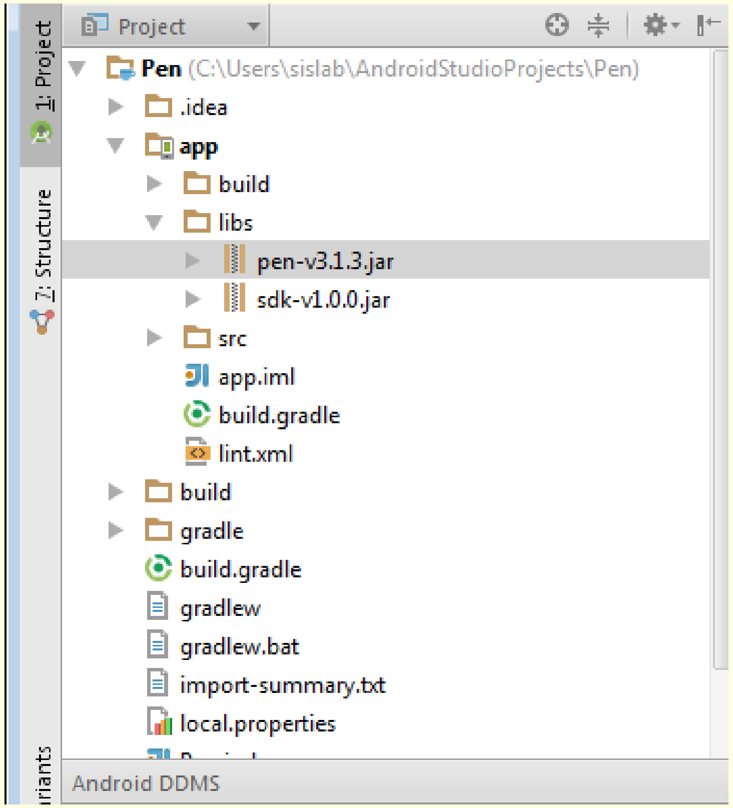
Choose options for your new file

Creates a new blank activity with an action bar.

Activity Name:	HelloPen
Layout Name:	activity_hello_pen
Title:	HelloPen
Menu Resource Name:	menu_hello_pen

FIRT PROGRAM: Hello Pen

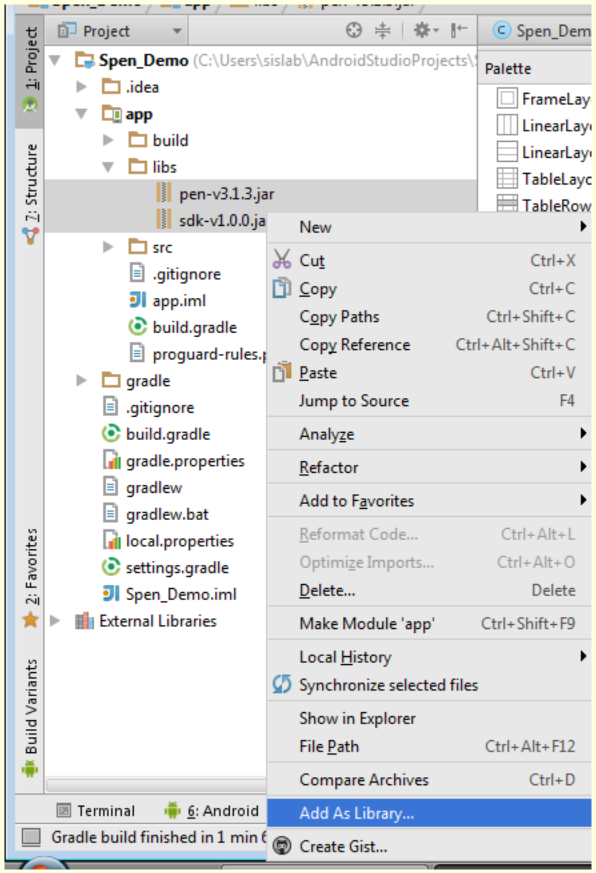
- **Adding a Library** of the downloaded PEN SDK to "**libs**" folder of the newly created project.
  - Copy the **SDK .jar files** to the **'libs'** folder in your new project to use the SDK you need for your application.



```
▼ Pen (C:\Users\sislab\AndroidStudioProjects\Pen)
  ▶ .idea
  ▼ app
    ▶ build
    ▼ libs
      ▶ pen-v3.1.3.jar
      ▶ sdk-v1.0.0.jar
    ▶ src
      app.iml
      build.gradle
      lint.xml
  ▶ build
  ▶ gradle
    build.gradle
    gradlew
    gradlew.bat
    import-summary.txt
    local.properties
```

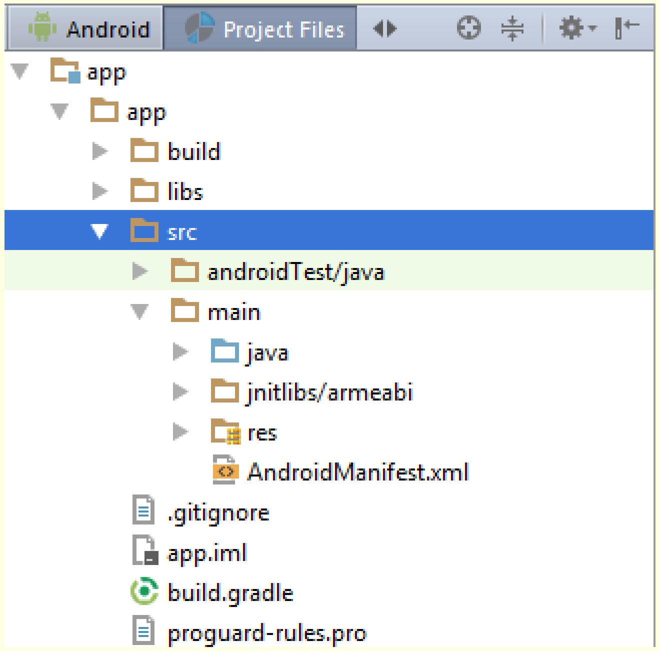
FIRT PROGRAM: Hello Pen

- **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**".



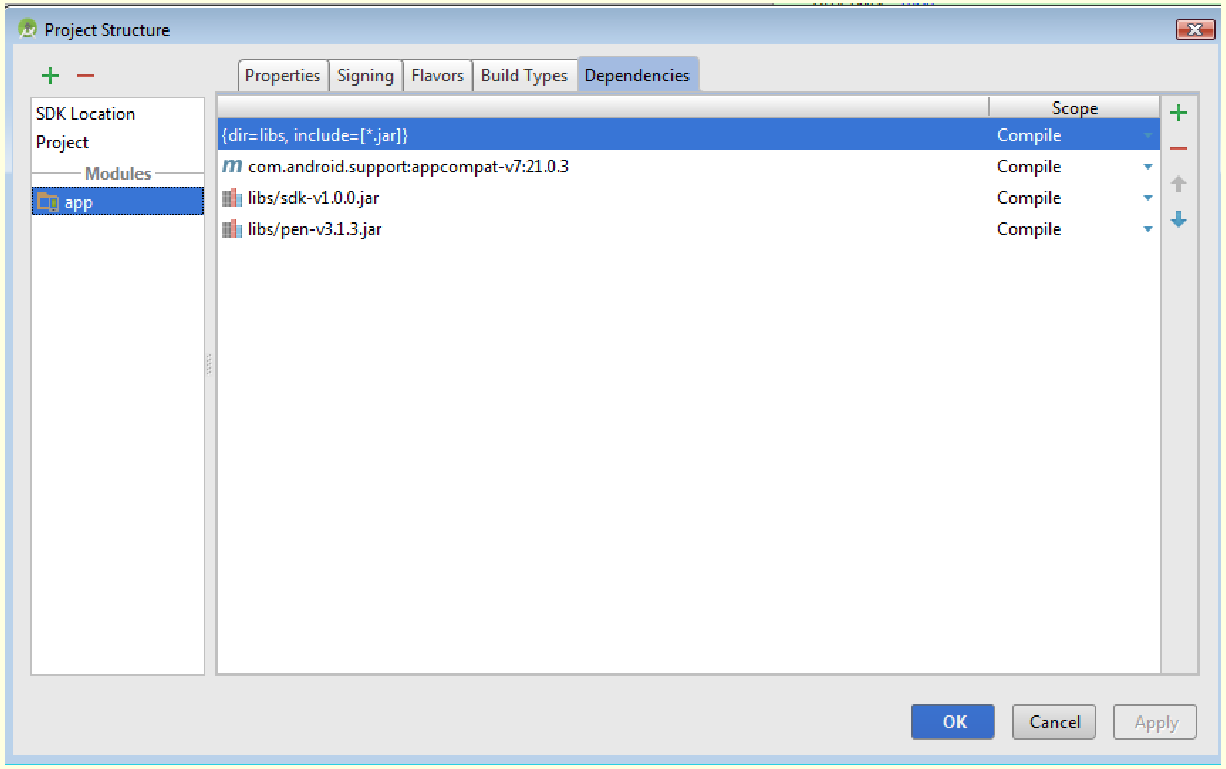
FIRT PROGRAM: Hello Pen

- **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**" folder in "libs" of the downloaded Spen SDK to the 'jnitlibs' folder.



FIRT PROGRAM: Hello Pen

- **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.



	Scope
{dir=libs, include=[*.jar]}	Compile
com.android.support:appcompat-v7:21.0.3	Compile
libs/sdk-v1.0.0.jar	Compile
libs/pen-v3.1.3.jar	Compile

FIRT PROGRAM: Hello Pen

- **Adding source code.**
  - Hello_pen.pdf

FIRT PROGRAM: Hello Pen

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

```
// add permission
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-permission
android:name="com.samsung.android.providers.context.permission.WRI
TE_USE_APP_FEATURE_SURVEY" />
```

FIRT PROGRAM: Hello Pen

- **Open “activity_hello_pen.xml” file:**
  - Delete: **<TextView android:text …>**
  - Add: **android:id="@+id/spenViewLayout">**

```
<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=".HelloPen"
    android:id="@+id/spenViewLayout">


</RelativeLayout>
```

FIRT PROGRAM: Hello Pen

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

```
public class HelloPen extends ActionBarActivity {

    private Context mContext;
    private SpenNoteDoc mSpenNoteDoc;
    private SpenPageDoc mSpenPageDoc;
    private SpenSurfaceView mSpenSurfaceView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_hello_pen);
        mContext = this;

        // Initialize Pen.
 boolean isSpenFeatureEnabled = false;
     Spen spenPackage = new Spen();
     try {
         spenPackage.initialize(this);
         isSpenFeatureEnabled =
spenPackage.isFeatureEnabled(Spen.DEVICE_PEN);
        } catch (SsdkUnsupportedException e) {
            Toast.makeText(mContext, "This device does not support S pen.",
                Toast.LENGTH_SHORT).show();
            e.printStackTrace();
            finish();
```

FIRT PROGRAM: Hello Pen

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

```
} catch (Exception e1) {
            Toast.makeText(mContext, "Cannot initialize Pen.",
                Toast.LENGTH_SHORT).show();
            e1.printStackTrace();
            finish();
        }

        // Create Pen View.
        RelativeLayout spenViewLayout =
            (RelativeLayout) findViewById(R.id.spenViewLayout);
        mSpenSurfaceView = new SpenSurfaceView(mContext);
        if (mSpenSurfaceView == null) {
            Toast.makeText(mContext, "Cannot create new SpenSurfaceView.",
                Toast.LENGTH_SHORT).show();
            finish();
        }
        spenViewLayout.addView(mSpenSurfaceView);

        // Get the dimensions of the screen.
        Display display = getWindowManager().getDefaultDisplay();
        Rect rect = new Rect();
        display.getRectSize(rect);
        // Create SpenNoteDoc.
        try {
```

FIRT PROGRAM: Hello Pen

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

```
mSpenNoteDoc =
            new SpenNoteDoc(mContext, rect.width(), rect.height());
      } catch (IOException e) {
         Toast.makeText(mContext, "Cannot create new NoteDoc.",
            Toast.LENGTH_SHORT).show();
         e.printStackTrace();
         finish();
      } catch (Exception e) {
         e.printStackTrace();
         finish();
      }
      // After adding a page to NoteDoc, get an instance and set it
      // as a member variable.
      mSpenPageDoc = mSpenNoteDoc.appendPage();
      mSpenPageDoc.setBackgroundColor(0xFFD6E6F5);
      mSpenPageDoc.clearHistory();
      // Set PageDoc to View.
      mSpenSurfaceView.setPageDoc(mSpenPageDoc, true);

if(isSpenFeatureEnabled == false) {
       mSpenSurfaceView.setToolTypeAction(SpenSurfaceView.TOOL_FINGER,
       SpenSurfaceView.ACTION_STROKE);
       Toast.makeText(mContext,

            "Device does not support S pen. \n You can draw strokes with your finger",
```

FIRT PROGRAM: Hello Pen

- Open “HelloPen.java” file and replace *class Hellopen* with the following codes:

```
     Toast.LENGTH_SHORT).show();
   }
  }

  @Override
  protected void onDestroy() {
      super.onDestroy();

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

      if(mSpenNoteDoc != null) {
          try {
              mSpenNoteDoc.close();
          } catch (Exception e) {
              e.printStackTrace();
          }
          mSpenNoteDoc = null;
      }
  };
}
```

FIRT PROGRAM: Hello Pen

- **Import classes to “HelloPen.java” file:**

```
import android.content.Context;
import android.widget.Toast;

import com.samsung.android.sdk.SsdkUnsupportedException;
import com.samsung.android.sdk.pen.Spen;
import com.samsung.android.sdk.pen.document.SpenNoteDoc;
import com.samsung.android.sdk.pen.document.SpenPageDoc;
import com.samsung.android.sdk.pen.engine.SpenSurfaceView;
```

Using the S-Pen Class

- **The Spen class provides the following methods:**
  - **initialize()** initializes Pen.
    - You need to initialize Pen before you can use it:

void initialize (Context context) throws SsdkUnsupportedException

    - If the device does not support S Pen, SsdkUnsupportedException is thrown.
  - **getVersionCode()** returns the Pen SDK version number as an integer.
  - **getVersionName()** returns the Pen SDK version name as a string.
  - **isFeatureEnabled()** checks if a Pen feature is available on the device.

Using the S-Pen Class

- If an **SsdkUnsupportedException** exception is thrown, check the exception message type using SsdkUnsupportedException.getType():
  - **VENDOR_NOT_SUPPORTED**: The device is not a Samsung device.
  - **DEVICE_NOT_SUPPORTED**: The device does not support Pen.
  - LIBRARY_NOT_INSTALLED: SpenSdk3.apk is not installed on the device.
  - **LIBRARY_UPDATE_IS_REQUIRED**: A necessary update for the Pen library or SpenSdk3.apk. If the
  - library or apk is not updated, the user cannot use the application.
  - **LIBRARY_UPDATE_IS_RECOMMENDED**: A recommendation to update the Pen library or SpenSdk3.apk, but it is not mandatory. The user can use the application without updating them.

Using the S-Pen Class

- **Checking the Availability of Pen Features:**
  - Using isFeatureEnabled() method:

boolean isFeatureEnabled(int type)

  - If the method returns false (which means S pen are not supported), **you can enable users to use their fingers:**

```
if(spenPackage.isFeatureEnabled(Spen.DEVICE_PEN) == false) {
    mSpenSurfaceView.setToolTypeAction(SpenSurfaceView.TOOL_FINGER,
                SpenSurfaceView.ACTION_STROKE);
    Toast.makeText(mContext,
                "Device does not support S pen. \n You can draw strokes with your finger",
    Toast.LENGTH_SHORT).show();
}
```

QUIZ

- **Modify your code so that you can use your finger to input information into Samsung devices**

Using Pen Views

- ***"SpenSurfaceView"* class:**
  - processes finger gestures or S pen input to express drawings on the viewport.
  - converts handwritten input to text or replays user input.
  - provides controls for scaling, rotating, moving, and selecting objects.
  - Combines with "SpenSettingPenLayout" to provide methods for managing user preferences (e.g. font, font size, and font color; the size, color, or type of the pen tool; the size of the eraser tool; and options for objects.

Using Pen Views

- **To create a drawing container in your application:**
  - Create an **SpenSurfaceView** instance by passing your application **Context** to the **SpenSurfaceView** constructor.
  - Add the **SpenSurfaceView** instance to the main layout.

```
RelativeLayout spenViewLayout = (RelativeLayout)
findViewById(R.id.spenViewLayout);
mSpenSurfaceView = new SpenSurfaceView(mContext);
if (mSpenSurfaceView == null) {
        finish();
}
spenViewLayout.addView(mSpenSurfaceView);
```

Using Pen Views

- **Creating a container to save input data from the SpenSurfaceView instance:**
  - Create an SpenNoteDoc instance by passing your application Context and the width and height of the SpenSurfaceView instance to the SpenNoteDoc constructor.
  - Call SpenPageDoc.setBackgroundColor() to specify the background color.
  - Use the SpenSurfaceView.setPageDoc() method to connect the SpenPageDoc instance to your SpenSurfaceView instance.

Using Pen Views

- **Creating a container to save input data from the SpenSurfaceView instance:**

```
try {
	mSpenNoteDoc = new SpenNoteDoc(mContext, rect.width(), rect.height());
} catch (IOException e) {
	e.printStackTrace();
	finish();
} catch (Exception e) {
	e.printStackTrace();
	finish();
}
// After adding a page to NoteDoc, get an instance
// and set it as a member variable.
mSpenPageDoc = mSpenNoteDoc.appendPage();
mSpenPageDoc.setBackgroundColor(0xFFD6E6F5);
mSpenPageDoc.clearHistory();
// Set PageDoc to View.
mSpenSurfaceView.setPageDoc(mSpenPageDoc, true);
```

Using Pen Views

- **Preventing Memory Leaks:**
  - Call **SpenNoteDoc.close()** and **SpenSurfaceView.close()** to close the **SpenNoteDoc** and **SpenSurfaceView** instances to prevent memory leaks when your application closes.

```
protected void onDestroy() {
        super.onDestroy();

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

        if (mSpenNoteDoc != null) {
            try {
                mSpenNoteDoc.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
            mSpenNoteDoc = null;
       }}
```

Using Pen Views

- **Preventing Memory Leaks:**
  - Call **SpenNoteDoc.close()** and **SpenSurfaceView.close()** to close the **SpenNoteDoc** and **SpenSurfaceView** instances to prevent memory leaks when your application closes.

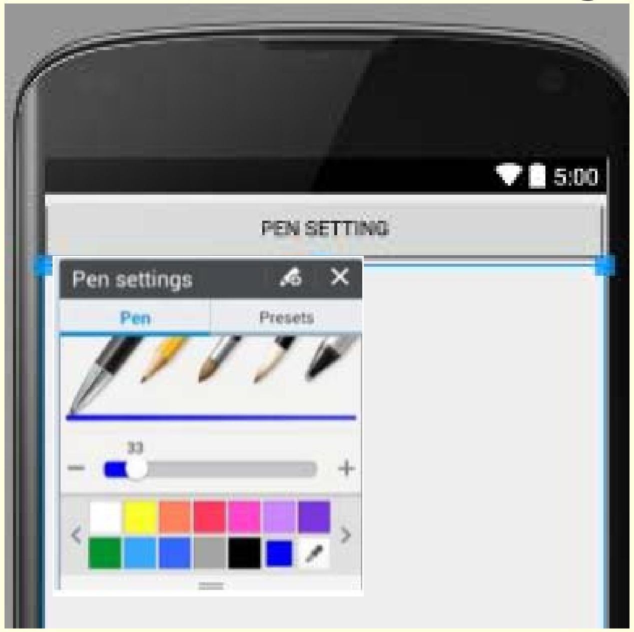
```
protected void onDestroy() {
        super.onDestroy();

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

        if (mSpenNoteDoc != null) {
            try {
                mSpenNoteDoc.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
            mSpenNoteDoc = null;
       }}
```

Using Pen Setting

- **Objective:**
  - add a pen settings button to your application for setting user preferences for the pen size, color and type.
- **Description:**
  - When the button is clicked, the SpenSettingPenLayout view appears to allow the user to configure the settings for the pen. If the button is clicked again, the window closes.



Using Pen Setting

- **Creating a New Project in Android Studio**
  - Application Name: **Demo_Pen**
  - Activity Name: **DemoPen**
- **Modify "AndroidManifest.xml"**
- **Add libraries**
- **Add source code:**
  - **Demo_pen.pdf**

Using Pen Setting

- **Edit Code of "Activity_demo_pen.xml":**

```
<LinearLayout 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:orientation="vertical"
    tools:context=".DemoPen" >

    <LinearLayout
        android:id="@+id/tool_menu"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <Button
            android:id="@+id/penBtn"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Pen Setting"/>

    </LinearLayout>

    <FrameLayout
        android:id="@+id/spenViewContainer"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >

        <RelativeLayout
            android:id="@+id/spenViewLayout"
            android:layout_width="match_parent"
            android:layout_height="match_parent" >
        </RelativeLayout>
    </FrameLayout>

</LinearLayout>
```

Using Pen Setting

- **Creating SpenSurfaceView and penSettingPenLayout**
  - **Call addView() and add your SpenSettingPenLayout instance to the SpenSurfaceView container defined in FrameLayout.**
  - To connect the settings view to your SpenSurfaceView instance, call SpenSettingPenLayout.setCanvasView() and pass your SpenSurfaceView instance.

Using Pen Setting

- **Creating SpenSurfaceView and penSettingPenLayout**

```
FrameLayout spenViewContainer =
            (FrameLayout) findViewById(R.id.spenViewContainer);
        RelativeLayout spenViewLayout =
            (RelativeLayout) findViewById(R.id.spenViewLayout);

        // Create PenSettingView.
        mPenSettingView =
            new SpenSettingPenLayout(mContext, new String(),
                spenViewLayout);
        if (mPenSettingView == null) {
            Toast.makeText(mContext, "Cannot create new PenSettingView.",
                Toast.LENGTH_SHORT).show();
            finish();
        }
        spenViewContainer.addView(mPenSettingView);

        // Create Pen View.
        mSpenSurfaceView = new SpenSurfaceView(mContext);
        if (mSpenSurfaceView == null) {
            Toast.makeText(mContext, "Cannot create new SpenSurfaceView.",
                Toast.LENGTH_SHORT).show();
            finish();
        }
        spenViewLayout.addView(mSpenSurfaceView);
        mPenSettingView.setCanvasView(mSpenSurfaceView);
```

Using Pen Setting

- **Initializing Pen Settings**
  - Create an SpenSettingPenInfo instance with your default settings for the pen tool.
  - Call SpenSurfaceView.setPenSettingInfo() to save the settings modified on your SpenSurfaceView instance.
  - Call setInfo() to save the settings to your SpenSettingPenLayout instance.

Using Pen Setting

- **Initializing Pen Settings**

```
private void initPenSettingInfo() {
      // Initialize pen settings.
      SpenSettingPenInfo penInfo = new SpenSettingPenInfo();
      penInfo.color = Color.BLUE;
      penInfo.size = 10;
      mSpenSurfaceView.setPenSettingInfo(penInfo);
      mPenSettingView.setInfo(penInfo);
  }
```

Using Pen Setting

- **Registering a Listener for the Pen Settings Button**
  - Create a pen Settings button.

```
mPenBtn = (Button) findViewById(R.id.penBtn);
mPenBtn.setOnClickListener(mPenBtnClickListener);
```

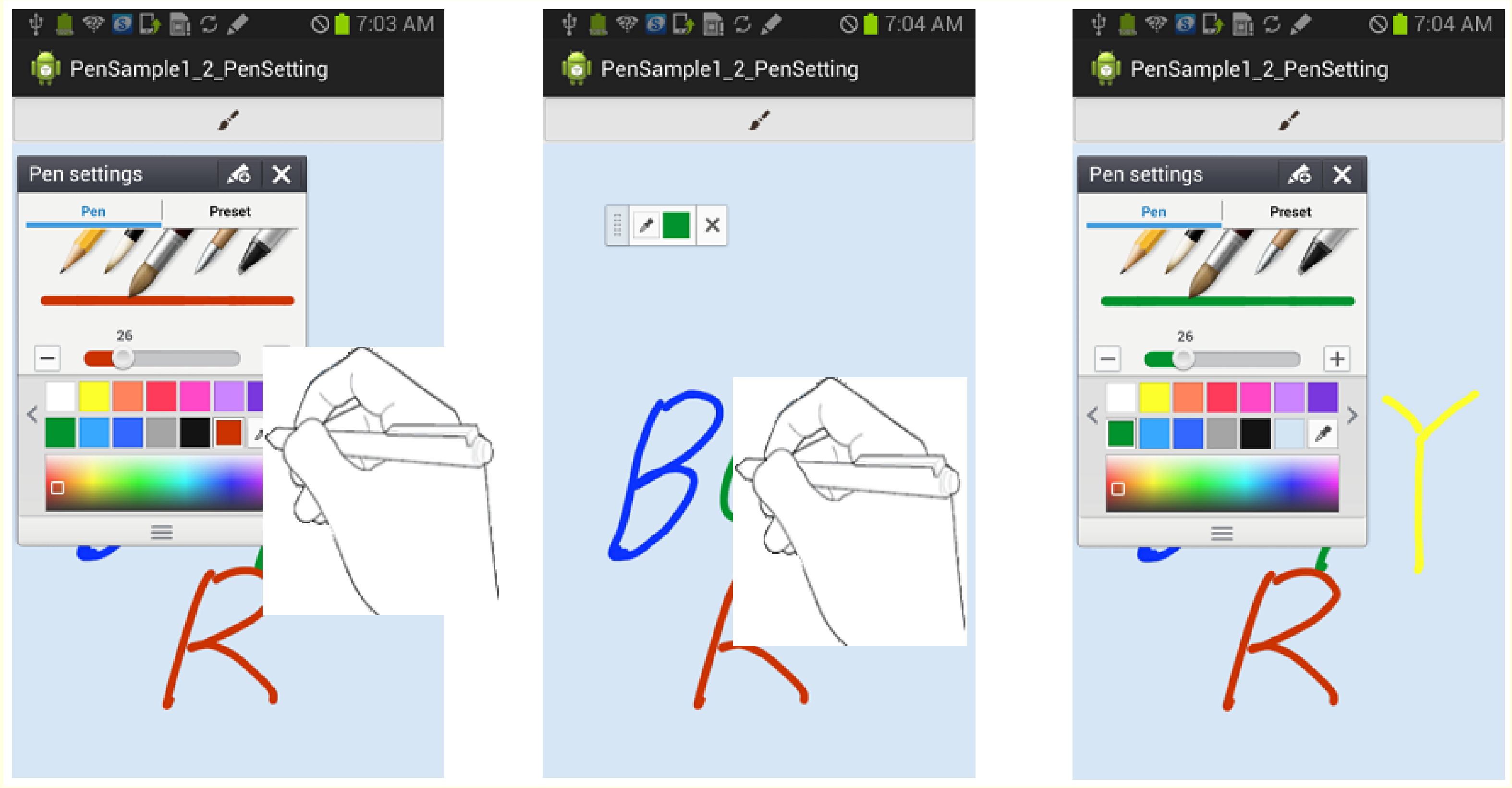
Using Pen Setting

- **Registering a Listener for the Pen Settings Button**
  - calling setOnClickListener() to handle the button click events.

```
private final View.OnClickListener mPenBtnClickListener = new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // If PenSettingView is open, close it.
            if (mPenSettingView.isShown()) {
                mPenSettingView.setVisibility(View.GONE);
                // If PenSettingView is not open, open it.
            } else {
                mPenSettingView.setViewMode(SpenSettingPenLayout.VIEW_MODE_EXTENSION);
                mPenSettingView.setVisibility(View.VISIBLE);
            }
        }
    };
```

Using Pen Setting

- **Registering a Listener for Color Picking**
  - Create an SpenColorPickerListener listener instance for the color picker in the SpenSettingPenLayout view and handle the event it returns.



Using Pen Setting

- **Registering a Listener for Color Picking**
  - Create an SpenColorPickerListener listener instance for the color picker in the SpenSettingPenLayout view and handle the event it returns.

```
private final SpenColorPickerListener mColorPickerListener = new
SpenColorPickerListener() {
        @Override
        public void onChanged(int color, int x, int y) {
            // Set the color from the Color Picker to the setting view.
            if (mPenSettingView != null) {
                SpenSettingPenInfo penInfo = mPenSettingView.getInfo();
                penInfo.color = color;
                mPenSettingView.setInfo(penInfo);
            }
        }
    };
```

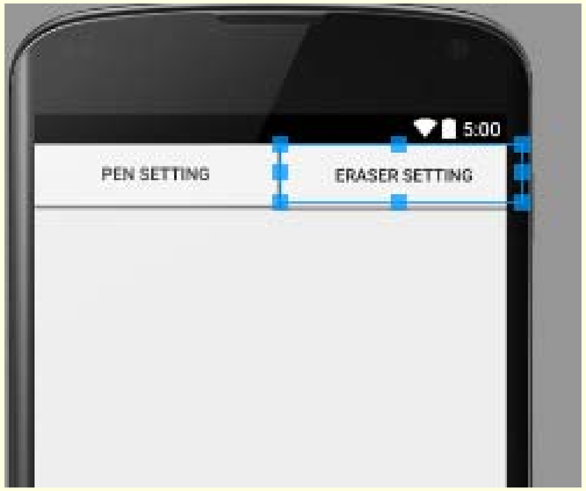
Using Pen Setting

- **Preventing Memory Leaks**
  - Call SpenSettingPenLayout.close() (in the onDestroy() method) to close your SpenSettingPenLayout instance when your application closes.

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

Using Eraser Setting

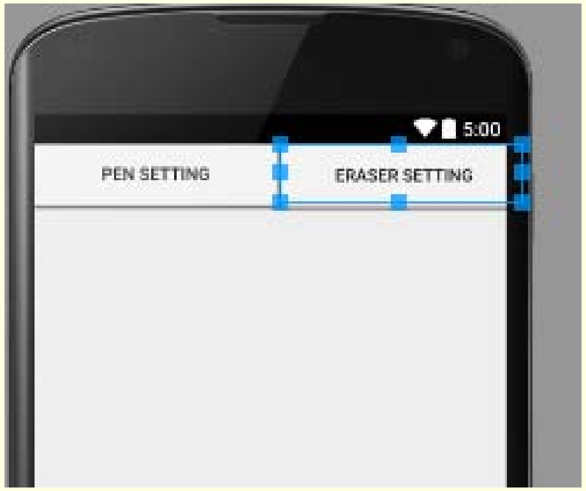
- **Project**
  - **Objective:** add a "Eraser setting" button to your application for setting preferences for the Eraser.
  - **Description:** When the "Eraser setting" button is clicked, the SpenSettingEraserLayout View appears to allow the user to configure the settings for the Eraser. If the button is clicked again, the window closes.



  - Reference: 4.1.3 section of ProgrammingGuide_Pen.pdf

Using S Pen

- References:
  - ProgrammingGuide_Pen.pdf


5:00
PEN SETTING
ERASER SETTING