Introduction:

Project - ColorChips

Required Environment - The Project directory is ready for use in **Android Studio**

Run Samples: The below screen shots illustrate the ColorChips Application work flow
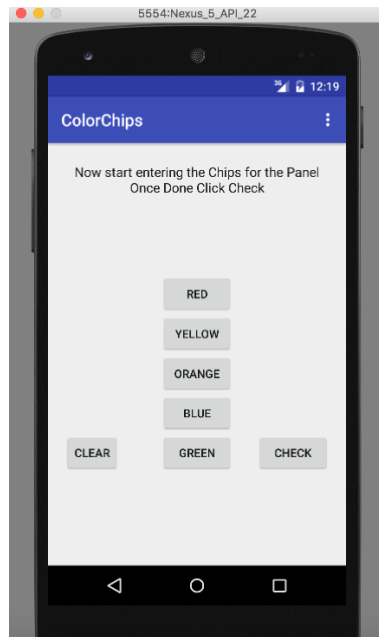
i.   Click on the front color marker

ii.   Click on end color marker

iii.   Now start clicking the Chips colors, once done click Check

iv.   If you click only one color of the chip and click 'Check', program will ask for other color

v.   If all the Chips placed in the access panel are placed end to end such that the adjacent colors match and the starting and ending chips are color matched to the corresponding markers, panel Unlocks
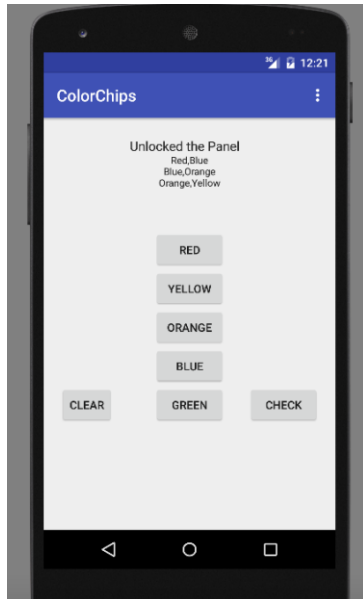
vi.   If the Chips placed in the access panel do not have an appropriate color match, access panel cannot be unlocked

vii.   Click the "Clear " button to start again

Testing:

In order to run the test, go to "Run/Debug Configurations" add new Android Test
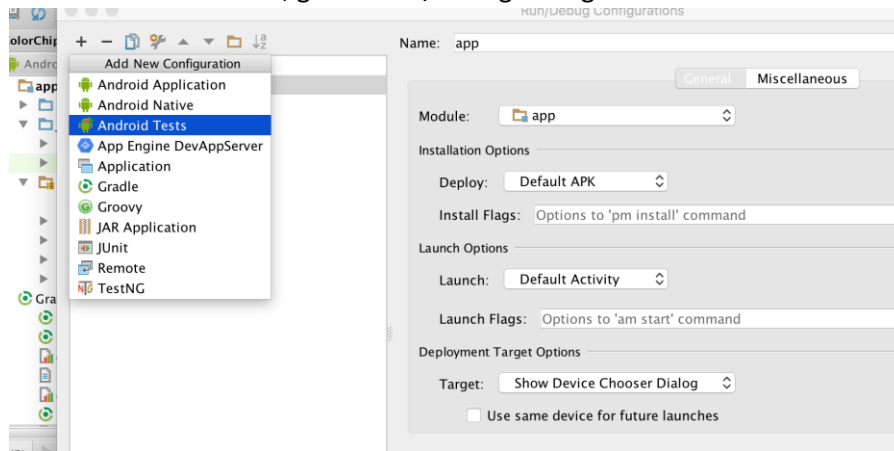
Now select newly created Test Configuration and run the test.
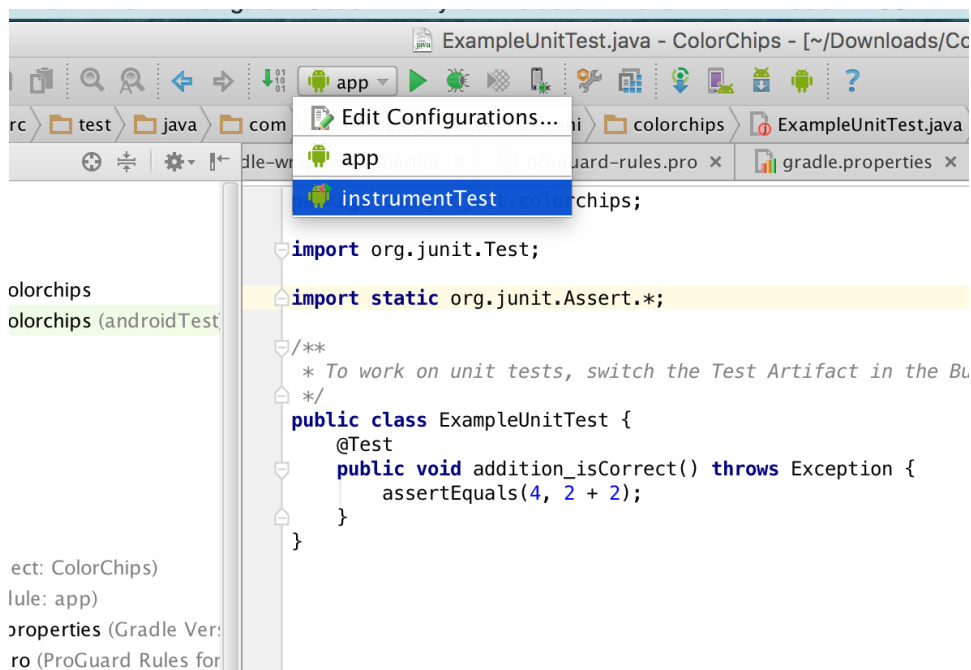
In this Project we have two kinds of testing

1. Local Unit Tests:
   These tests run on your local machine only. These tests are compiled to run on your local machine only. These tests do not have dependencies on Android Framework

```
                                    }
 ▶ com.project.colorchips            @Test public void example3(){
 ▼ com.project.colorchips (androidTest)   ChipsInput input = buildInput(
    ▶ suite                                   "blue", "red",
    ⓒ ApplicationTest                         "red", "blue",
    ⓒ HintMatcher                             "orange", "red",
    ⓒ UnlockChipJunitTest                     "green", "green",
    ⓒ unlockChipsInstrumentationTest          "orange", "yellow",
 ▼ res                                        "red", "yellow");
    drawable                              unlocker.unlock(input);
 ▼ layout
    activity_main.xml                      assertThat(Unlockchips.UNLOCK_FAILED_MSG, is(equalTo(unlocker.getResult())));
    content_main.xml                   }
 ▼ menu
    menu_main.xml                     private ChipsInput buildInput(String... colors) {
 ▼ mipmap                                  ChipsInput input = new ChipsInput();
    ▼ ic_launcher.png (5)                  input.parse(buildPairs(colors));
       ic_launcher.png (hdpi)             return input;
       ic_launcher.png (mdpi)         }
       ic_launcher.png (xhdpi)
                                      private String buildPairs(String... colors) {
                                          StringBuilder sb = new StringBuilder();
```

2. Instrumented Unit Tests:
    These tests run on an Android Device or emulator. These tests have access to Instrumentation information. These tests have Android dependencies which mock objects cannot easily satisfy.

```
    ⓒ HintMatcher                     (mActivityRule.getActivity()).setInputString(input);
    ⓒ UnlockChipJunitTest             // Type the two operands in the EditText fields
    ⓒ unlockChipsInstrumentationTest  // Click on a given operation button
 ▼ res                                onView(withId(btnOperationResId)).perform(click());
    drawable                          // Check the expected test is displayed in the Ui
 ▼ layout                             onView(withId(R.id.resultText)).check(matches(withText(expectedResult)));
    activity_main.xml
    content_main.xml              }
 ▼ menu
    menu_main.xml                 private String buildPairs(String... colors) {
 ▼ mipmap                             StringBuilder sb = new StringBuilder();
    ▼ ic_launcher.png (5)             for(int i = 0; i < colors.length; i++){
       ic_launcher.png (hdpi)             sb.append(colors[i]);
       ic_launcher.png (mdpi)             //figure out separator by position
       ic_launcher.png (xhdpi)            if(i % 2 == 0){
       ic launcher png (xxhdpi)              sb.append(",");
                                         } else {
                                             sb.append("\n");
                                         }
                                      }
                                      return sb.toString().trim();
```