

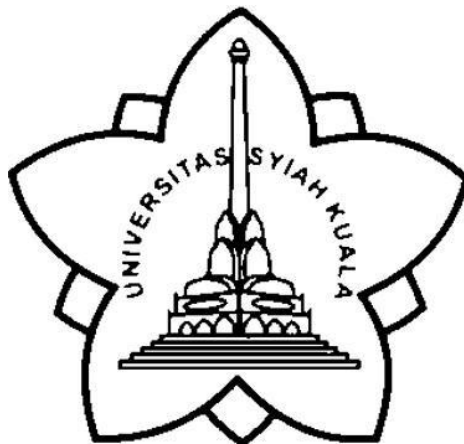
## **LAPORAN TUGAS JUNIT**

disusun untuk memenuhi  
Tugas mata kuliah Kualitas Perangkat Lunak

Oleh:

**Muhammad Nouval Rifqi**

**2208107010075**



**DEPARTEMEN INFORMATIKA  
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM  
UNIVERSITAS SYIAH KUALA  
DARUSSALAM, BANDA ACEH**

**2025**

## **1. Pendahuluan**

**JUnit** merupakan salah satu framework populer untuk pengujian unit dalam pengembangan perangkat lunak berbasis Java. Framework ini memungkinkan developer untuk melakukan pengujian kode secara otomatis, guna memastikan bahwa setiap bagian fungsional dari aplikasi bekerja sebagaimana mestinya. Penggunaan JUnit mempermudah proses identifikasi bug sejak awal dalam siklus pengembangan, meningkatkan stabilitas kode, dan mendukung proses debugging. Laporan ini membahas penerapan pengujian unit menggunakan JUnit pada aplikasi DiskonApp.

### **Latar Belakang**

Dalam proses pengembangan perangkat lunak, menjaga kualitas kode merupakan hal yang krusial. Kesalahan kecil dalam logika atau perhitungan dapat menyebabkan dampak besar terhadap keakuratan dan performa aplikasi. Oleh sebab itu, dibutuhkan metode pengujian yang terstruktur untuk memastikan bahwa setiap komponen kode telah diuji secara menyeluruh. JUnit menawarkan sistem otomatisasi pengujian yang efektif, memungkinkan pengembang untuk menguji berbagai kemungkinan skenario. Melalui penerapan JUnit dalam proyek ini, fitur perhitungan diskon dapat divalidasi agar berjalan sesuai dengan spesifikasi yang diinginkan.

### **Tujuan**

- Menerapkan pengujian unit pada aplikasi DiskonApp menggunakan framework JUnit.
- Memastikan keakuratan fungsi dalam menghitung diskon.
- Mengevaluasi sejauh mana JUnit efektif dalam menemukan kesalahan dalam kode.

### **Manfaat**

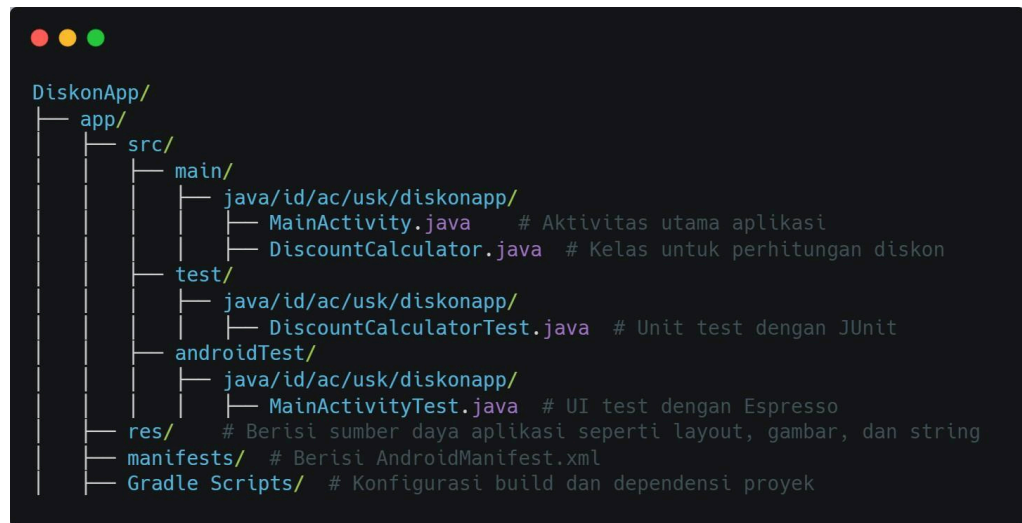
- Meningkatkan kualitas perangkat lunak dengan meminimalkan potensi kesalahan.
- Mempermudah proses pelacakan dan perbaikan bug.
- Menjamin bahwa perubahan kode tidak mengganggu fungsi yang sudah ada.
- Menghemat waktu dalam proses pengujian dibandingkan dengan metode manual.

## 2. Implementasi Proyek

### a. Tools

1. Java Development Kit (JDK): Untuk menjalankan aplikasi berbasis Java.
2. Android Studio: IDE yang digunakan untuk mengembangkan aplikasi.
3. JUnit 4: Framework pengujian unit untuk Java.
4. Gradle: Untuk mengelola dependensi dan konfigurasi proyek.

### b. Struktur Proyek



*Gambar 1. struktur proyek*

Struktur proyek ini mengikuti standar pengembangan aplikasi Android. Direktori main/ berisi kode utama aplikasi, termasuk tampilan dan logika perhitungan diskon. Direktori test/ digunakan untuk pengujian unit dengan JUnit, sedangkan androidTest/ digunakan untuk pengujian UI dengan Espresso. Direktori res/ menyimpan sumber daya aplikasi, sementara manifests/ berisi konfigurasi aplikasi dalam AndroidManifest.xml. Gradle Scripts digunakan untuk mengatur build dan dependensi proyek.

## c. Implementasi Kode

### 1. Kode Aplikasi Utama

```
package id.ac.usk.diskonapp;

import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        EditText edtHarga = findViewById(R.id.edtHarga);
        EditText edtDiskon = findViewById(R.id.edtDiskon);
        Button btnHitung = findViewById(R.id.btnHitung);
        TextView txtHasil = findViewById(R.id.txtHasil);

        btnHitung.setOnClickListener(v -> {
            String hargaStr = edtHarga.getText().toString();
            String diskonStr = edtDiskon.getText().toString();

            if (!hargaStr.isEmpty() && !diskonStr.isEmpty()) {
                double harga = Double.parseDouble(hargaStr);
                double diskon = Double.parseDouble(diskonStr);

                if (harga > 0 && diskon >= 0 && diskon <= 100) {
                    double hargaSetelahDiskon =
DiscountCalculator.calculateDiscountedPrice(harga, diskon);
                    double jumlahDiskon =
DiscountCalculator.calculateSavings(harga, diskon);

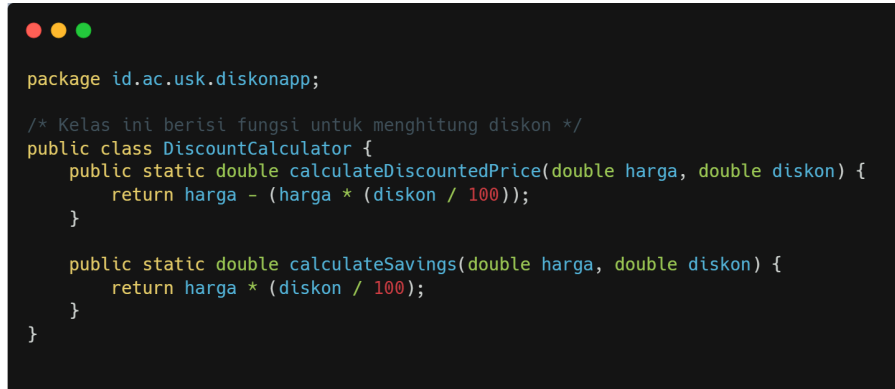
                    txtHasil.setText("Harga setelah diskon: Rp" +
hargaSetelahDiskon + "\nHemat: Rp" + jumlahDiskon);
                } else {
                    txtHasil.setText("Masukkan angka yang valid!");
                }
            } else {
                txtHasil.setText("Mohon isi semua kolom!");
            }
        });
    }
}
```

*Gambar 2. code MainActivity.java sebagai utama aplikasi*

Kode MainActivity.java bertanggung jawab untuk menangani tampilan utama aplikasi DiskonApp, yang memungkinkan pengguna menghitung harga setelah diskon berdasarkan input harga awal dan persentase diskon. Kode ini menghubungkan elemen UI seperti EditText untuk input harga dan diskon, Button untuk memproses perhitungan, serta TextView untuk menampilkan hasil. Saat tombol dihitung ditekan, sistem memeriksa apakah input tidak kosong, lalu mengonversi nilai menjadi double. Jika nilai valid, perhitungan dilakukan menggunakan metode dari DiscountCalculator, yang

memisahkan logika bisnis dari tampilan untuk meningkatkan modularitas. Hasil akhirnya ditampilkan kepada pengguna, sementara validasi tambahan memastikan input sesuai agar mencegah kesalahan.

## 2. Kode Fungsi untuk Menghitung Diskon

A screenshot of a code editor with a dark background and light-colored text. The code is in Java and defines a class named DiscountCalculator. It includes a package declaration, a comment, and two static methods: calculateDiscountedPrice and calculateSavings. The code is as follows:

```
package id.ac.usk.diskonapp;

/* Kelas ini berisi fungsi untuk menghitung diskon */
public class DiscountCalculator {
    public static double calculateDiscountedPrice(double harga, double diskon) {
        return harga - (harga * (diskon / 100));
    }

    public static double calculateSavings(double harga, double diskon) {
        return harga * (diskon / 100);
    }
}
```

*Gambar 3. code DiscountCalculator.java*

Kelas DiscountCalculator berfungsi sebagai utilitas untuk menghitung harga setelah diskon dan jumlah penghematan berdasarkan nilai harga awal dan persentase diskon. Kelas ini memiliki dua metode statis, yaitu calculateDiscountedPrice(double harga, double diskon), yang mengembalikan harga akhir setelah diskon diterapkan, dan calculateSavings(double harga, double diskon), yang menghitung jumlah diskon yang diberikan.

Metode calculateDiscountedPrice bekerja dengan mengurangi nilai diskon dari harga awal, sedangkan calculateSavings menghitung total potongan berdasarkan persentase yang diberikan. Dengan pendekatan ini, kode menjadi lebih modular dan dapat digunakan kembali di berbagai bagian aplikasi tanpa perlu membuat objek baru dari kelas DiscountCalculator.

### 3. Kode Layout Tampilan

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fillViewport="true">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="24dp"
        tools:context=".MainActivity"
        android:background="@color/white">

        <!-- Title -->
        <TextView
            android:id="@+id/title"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/app_name"
            android:textSize="30sp"
            android:textColor="@color/black"
            android:textStyle="bold"
            android:fontFamily="serif-monospace"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            android:layout_marginTop="40dp"
            android:layout_marginBottom="30dp" />

        <!-- Input Harga -->
        <com.google.android.material.textfield.TextInputLayout
            android:id="@+id/layoutHarga"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            app:boxStrokeWidthFocused="2dp"
            app:layout_constraintTop_toBottomOf="@id/title"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            android:layout_marginBottom="20dp">

            <com.google.android.material.textfield.TextInputEditText
                android:id="@+id/edtHarga"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="Masukkan Harga Barang"
                android:textSize="18sp"
                android:inputType="numberDecimal"
                android:padding="12dp" />
        </com.google.android.material.textfield.TextInputLayout>

        <!-- Input Diskon -->
        <com.google.android.material.textfield.TextInputLayout
            android:id="@+id/layoutDiskon"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            app:boxStrokeWidthFocused="2dp"
            app:layout_constraintTop_toBottomOf="@id/layoutHarga"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            android:layout_marginBottom="24dp">

            <com.google.android.material.textfield.TextInputEditText
                android:id="@+id/edtDiskon"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="Masukkan Diskon (%)"
                android:textSize="18sp"
                android:inputType="number"
                android:padding="12dp" />
        </com.google.android.material.textfield.TextInputLayout>
```

```

<!-- Button Hitung Diskon -->
<Button
    android:id="@+id/btnHitung"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Hitung Diskon"
    android:backgroundTint="@android:color/holo_purple"
    android:textColor="@android:color/white"
    android:textSize="20sp"
    android:padding="14dp"
    app:layout_constraintTop_toBottomOf="@id/layoutDiskon"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    android:layout_marginBottom="24dp" />

<!-- Hasil Perhitungan -->
<TextView
    android:id="@+id/txtHasil"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/hasil_awal"
    android:textSize="22sp"
    android:textColor="@color/black"
    android:textAlignment="center"
    android:gravity="center"
    android:padding="16dp"
    android:background="@drawable/result_background"
    app:layout_constraintTop_toBottomOf="@id/btnHitung"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    android:layout_marginBottom="30dp" />

</androidx.constraintlayout.widget.ConstraintLayout>
</ScrollView>

```

*Gambar 4. code layout activity\_main.xml*

Kode XML di atas merupakan layout utama dari aplikasi DiskonApp yang dirancang menggunakan ScrollView agar tampilan tetap dapat di-scroll jika konten melebihi layar. ConstraintLayout digunakan sebagai wadah utama untuk mengatur tata letak elemen secara fleksibel.

Bagian utama layout terdiri dari beberapa elemen, yaitu TextView sebagai judul aplikasi, TextInputLayout yang membungkus TextInputEditText untuk input harga dan diskon, serta Button untuk memproses perhitungan diskon. Hasil perhitungan ditampilkan dalam TextView dengan latar belakang khusus untuk membedakan dari elemen lain. Elemen-elemen ini telah disusun menggunakan constraint agar tetap responsif di berbagai ukuran layar. Selain itu, desain menggunakan Material Components untuk tampilan yang lebih modern dan intuitif.

#### 4. Kode Unit Test

```
package id.ac.usk.diskonapp;

import static org.junit.Assert.assertEquals;
import org.junit.Test;

public class DiscountCalculatorTest {

    /*
     * Test Case 1: Memeriksa apakah harga setelah diskon dihitung dengan benar.
     * Input: Harga = 100000, Diskon = 20%
     * Expected Output: 80000
     */
    @Test
    public void testCalculateDiscountedPrice() {
        double result = DiscountCalculator.calculateDiscountedPrice(100000, 20);
        assertEquals(80000, result, 0.01);
    }

    /*
     * Test Case 2: Memeriksa apakah jumlah diskon yang dikurangi benar.
     * Input: Harga = 200000, Diskon = 10%
     * Expected Output: 20000
     */
    @Test
    public void testCalculateSavings() {
        double result = DiscountCalculator.calculateSavings(200000, 10);
        assertEquals(20000, result, 0.01);
    }

    /*
     * Test Case 3: Memeriksa apakah harga tetap sama jika diskon adalah 0%.
     * Input: Harga = 50000, Diskon = 0%
     * Expected Output: 50000
     */
    @Test
    public void testNoDiscount() {
        double result = DiscountCalculator.calculateDiscountedPrice(50000, 0);
        assertEquals(50000, result, 0.01);
    }
}
```

*Gambar 5. code unit test DiscountCalculatorTest.java*

Kode unit test dalam DiscountCalculatorTest digunakan untuk menguji keakuratan fungsi perhitungan diskon dalam kelas DiscountCalculator. Pengujian ini menggunakan JUnit dan terdiri dari tiga skenario utama. Pertama, testCalculateDiscountedPrice menguji apakah harga setelah diskon dihitung dengan benar dengan contoh harga Rp100.000 dan diskon 20%, yang seharusnya menghasilkan Rp80.000. Kedua, testCalculateSavings memastikan bahwa jumlah penghematan dihitung dengan benar, misalnya dengan harga Rp200.000 dan diskon 10%, maka penghematan yang diharapkan adalah Rp20.000. Ketiga, testNoDiscount menguji apakah harga tetap sama jika diskon bernilai 0%, misalnya Rp50.000 tanpa diskon harus tetap Rp50.000.



## 5. Kode Instrumental Test

```
package id.ac.usk.diskonapp;

import androidx.test.ext.junit.rules.ActivityScenarioRule;
import androidx.test.ext.junit.runners.AndroidJUnit4;
import org.junit.Rule;
import org.junit.Test;
import org.junit.runner.RunWith;
import static androidx.test.espresso.Espresso.onView;
import static androidx.test.espresso.action.ViewActions.*;
import static androidx.test.espresso.matcher.ViewMatchers.*;
import static androidx.test.espresso.assertion.ViewAssertions.matches;

@RunWith(AndroidJUnit4.class)
public class MainActivityTest {

    @Rule
    public ActivityScenarioRule<MainActivity> activityRule = new
    ActivityScenarioRule<>(MainActivity.class);

    /*
     * Test Case 1: Menguji apakah perhitungan diskon di UI berfungsi dengan
     * benar.
     * Langkah pengujian:
     * 1. Ketik harga "100000" di input harga.
     * 2. Ketik diskon "20" di input diskon.
     * 3. Tekan tombol "Hitung".
     * Expected Output: Hasil yang ditampilkan di UI harus "Harga setelah
     * diskon: Rp80000.0\nHemat: Rp20000.0".
     */
    @Test
    public void testDiscountCalculation() {
        onView(withId(R.id.edtHarga)).perform(typeText("100000"),
        closeSoftKeyboard());
        onView(withId(R.id.edtDiskon)).perform(typeText("20"),
        closeSoftKeyboard());
        onView(withId(R.id.btnHitung)).perform(click());
        onView(withId(R.id.txtHasil)).check(matches(withText("Harga setelah
        diskon: Rp80000.0\nHemat: Rp20000.0")));
    }

    /*
     * Test Case 2: Memastikan validasi input bekerja jika field kosong.
     * Langkah pengujian:
     * 1. Langsung tekan tombol "Hitung" tanpa memasukkan input.
     * Expected Output: Pesan error "Mohon isi semua kolom!" harus muncul di UI.
     */
    @Test
    public void testEmptyInputValidation() {
        onView(withId(R.id.btnHitung)).perform(click());
        onView(withId(R.id.txtHasil)).check(matches(withText("Mohon isi semua
        kolom!")));
    }
}
```

*Gambar 6. code instrumental test MainActivityTest.java*

Kode MainActivityTest merupakan pengujian otomatis berbasis UI menggunakan Espresso untuk memastikan bahwa tampilan dan fungsionalitas pada aplikasi berjalan sesuai yang diharapkan. Pengujian ini dijalankan dengan AndroidJUnit4 dan menggunakan ActivityScenarioRule untuk memulai MainActivity sebelum setiap pengujian dilakukan.

### 3. Pengujian

Terdapat dua skenario pengujian utama. Pertama, `testDiscountCalculation` menguji apakah perhitungan diskon di UI berfungsi dengan benar. Pengujian dilakukan dengan memasukkan harga "100000" dan diskon "20" ke dalam input yang tersedia, lalu menekan tombol "Hitung". Hasil yang ditampilkan di UI harus sesuai dengan yang diharapkan, yaitu "Harga setelah diskon: Rp80000.0\nHemat: Rp20000.0". Kedua, `testEmptyInputValidation` menguji validasi ketika input kosong. Dalam skenario ini, pengguna langsung menekan tombol "Hitung" tanpa mengisi input, dan aplikasi harus menampilkan pesan error "Mohon isi semua kolom!" di UI. Pengujian ini penting untuk memastikan bahwa fitur perhitungan diskon tidak hanya berfungsi dengan benar, tetapi juga menangani kesalahan input dengan baik, sehingga memberikan pengalaman pengguna yang lebih baik dan mencegah kesalahan perhitungan dalam aplikasi.

Pada tahapan ini, saya melakukan pengujian terhadap aplikasi penghitung diskon sederhana ini untuk memastikan bahwa semua fitur yang telah dibuat berjalan dengan benar. Pengujian saya lakukan dalam dua tahap, yaitu Unit Test dan Instrumental Test.

#### a. Unit Test

Unit test dilakukan untuk menguji fungsi `calculateDiscountedPrice()` dan `calculateSavings()` yang terdapat dalam `DiscountCalculator.java`. Pengujian ini menggunakan JUnit untuk memastikan bahwa perhitungan diskon berjalan dengan benar dalam berbagai skenario, termasuk perhitungan harga setelah diskon, jumlah penghematan, serta validasi jika diskon bernilai nol.

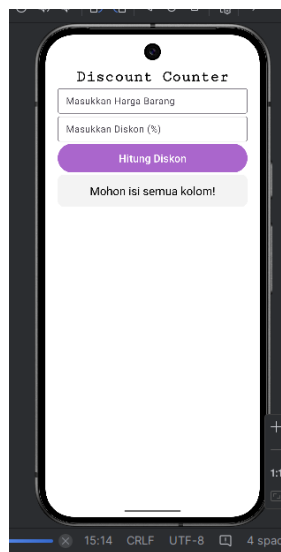


*Gambar 7. Hasil pengujian Unit Test*

Hasil: Semua unit test berjalan dengan sukses tanpa error, menunjukkan bahwa perhitungan diskon telah berfungsi dengan benar.

b. Instrumental Test

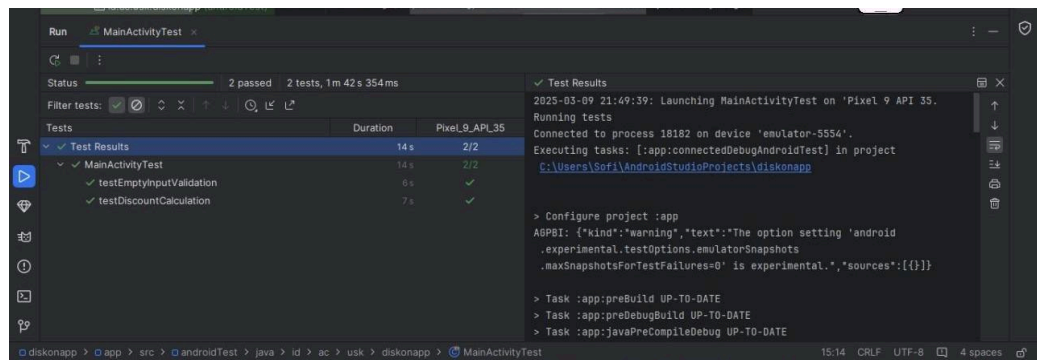
Instrumental test dilakukan untuk menguji interaksi pengguna dengan UI pada aplikasi menggunakan emulator. Pengujian ini menggunakan Espresso untuk memastikan bahwa input harga dan diskon dapat dimasukkan dengan benar, serta tombol "Hitung" menghasilkan output sesuai dengan perhitungan yang diharapkan. Selain itu, dilakukan validasi jika pengguna tidak mengisi input sebelum menekan tombol.



*Gambar 8. tampilan hasil instrumental test case 1*



*Gambar 9. tampilan hasil instrumental test case 2*



*gambar 10. Hasil pengujian Intrumental Test*

Hasil: Instrumental test berhasil dijalankan tanpa error, dan hasil yang ditampilkan di layar sesuai dengan ekspektasi, termasuk validasi input yang bekerja dengan baik.

#### 4. Kesimpulan

Dari hasil pengujian yang telah dilakukan, penggunaan JUnit sebagai kerangka kerja untuk pengujian unit pada aplikasi DiskonApp terbukti mampu memastikan ketepatan dalam perhitungan diskon. Pengujian dengan JUnit menunjukkan bahwa fungsi *calculateDiscountedPrice()* dan *calculateSavings()* bekerja sesuai harapan dalam berbagai situasi, baik saat diskon diterapkan, tidak diterapkan, maupun dalam proses validasi input.

Selain itu, pengujian instrumental dengan Espresso berhasil memastikan bahwa interaksi pengguna dengan antarmuka aplikasi berjalan dengan lancar. Tampilan hasil perhitungan di layar sesuai dengan yang diharapkan, dan aplikasi mampu menangani berbagai jenis input, termasuk saat input kosong diberikan.

Dengan dukungan pengujian otomatis ini, proses pelacakan dan perbaikan bug menjadi lebih cepat, serta potensi kesalahan perhitungan dapat ditekan. Integrasi JUnit dan Espresso dalam proyek ini tidak hanya meningkatkan kualitas perangkat lunak secara keseluruhan, tetapi juga menjamin stabilitas fitur utama aplikasi meskipun ada perubahan kode di masa mendatang.

#### 5. Lampiran

Link	Github	Repository	Project:
<a href="https://github.com/nouval2004/TugasJUnitApp">https://github.com/nouval2004/TugasJUnitApp</a>			