Android Design Patterns

In this exercise you will identify design patterns used in the Android source code and analyze why each pattern is used. Android is a mobile operating system based on the Linux kernel and is currently developed by Google – although many forks exist.

Due to the large size of the Android source code, we only provide a small subset of Android for today's recitation. If you want the full source distribution for Android, please visit https://source.android.com/source/downloading.html.

Hints:

- Look for base classes first when trying to identify design patterns. For example, it is very likely that `FrameFilter.java` implements `Filter.java`
- The different components of a design pattern might be in multiple packages.

Pattern Mining

For each design pattern below, complete each of the following four parts:

1. Review the class diagrams and purpose of the design pattern from the required course textbook or course slides, if necessary.
2. Find one example of the design pattern in the provided source code. We suggest a package to help you find the pattern. Using clues from each class—such as what it imports, extends, and implements—find the various components of the design pattern and record your observations. Remember: the different components of a design pattern might be in multiple packages.
3. Describe the design goals achieved with the use of the pattern.
4. Describe the design disadvantages associated with the use of the pattern.

Recording Findings

- Strategy
  - Package: `com.android.notificationstudio.editor`
  - * Strategy abstraction:
    * Concrete strategy (find at least two):
      .
      .
  - Design goals achieved:
  - Design disadvantages:

- Template Method
  - Package: `androidx.media.filterfw`
  - * Abstract Class:
    - Algorithm template method:
    - Primitive operation method:
  - * Concrete Class (at least 2):
    .
    .
  - Design goals achieved:
  - Design disadvantages:
- Builder
  - Package: `com.android.notificationstudio.generator`
  - * Builder class:
  - * Product class:
  - Design goals achieved:
  - Design disadvantages:
- Factory Method
  - Package: `androidx.media.filterfw`
  - * Creator Class:
  - * Product Classes (find at least two):
    .
    .
  - Design goals achieved:
  - Design disadvantages:
- Singleton
  - Package: `androidx.media.filterfw`
  - * Singleton Class:
  - Design goals achieved:
  - Design disadvantages:
- Observer
  - Package: `androidx.media.filterfw.decoder`
  - * Subject Class:
  - * Observer Class:
  - Design goals achieved:
  - Design disadvantages: