Android Development

Patterns. Principles. Architecture

SoftUni Team

Teodor Kostadinov

Software University

SoftUni Foundation

Software University

http://softuni.bg

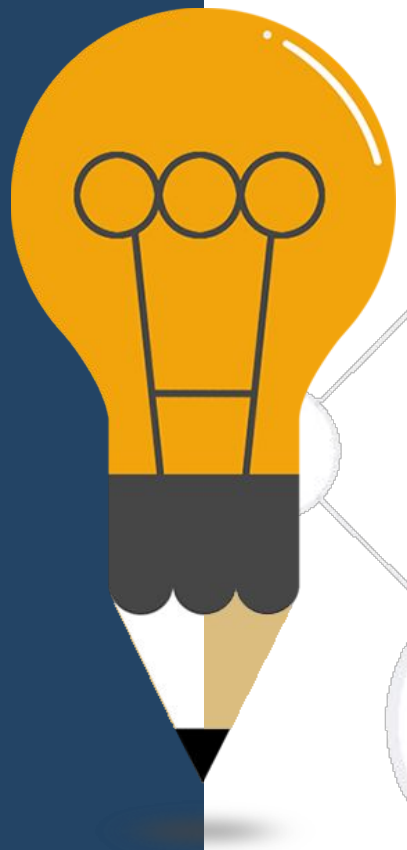
Questions

Software University

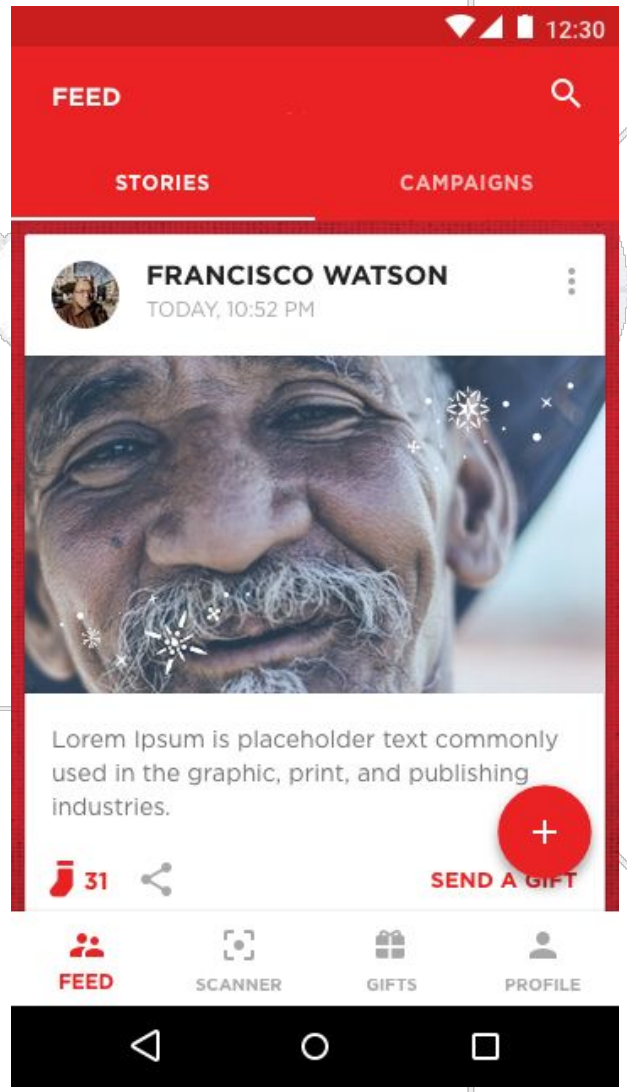
sli.do

#Android

The Road So Far



1. Views and Layouts
2. Adapters and RecyclerView
3. Activity and Fragments
4. Internet connectivity
5. Storage



Agenda for Today

Software University

- Design Patterns
  - Singleton
  - Factory
  - Decorator
- Principles
  - DRY
  - SOLID
- Dependency Injection
- Architectures
  - MVP
  - MVVM
  - Onion
- Next Level Android
- Sample Exam

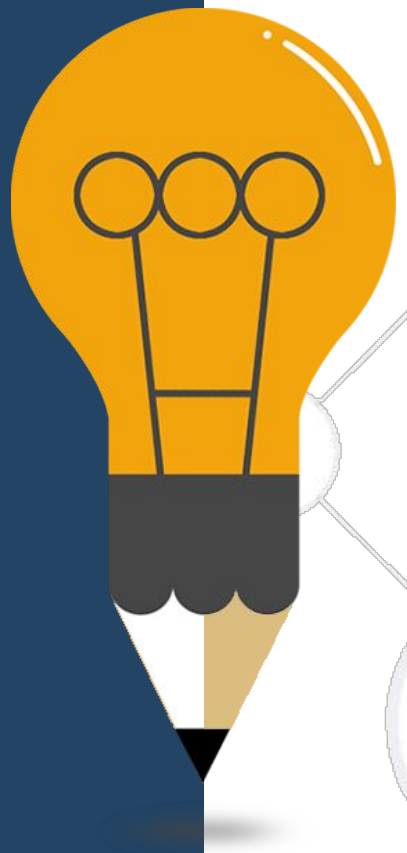
Commonly Used Design Patterns

Software University

- Singleton
  - Use it when you need only one object per class
  - It uses static field and method, and private constructor

```
class RetrofitWrapper {
    private static RetrofitWrapper instance;
    private static RetrofitWrapper getInstance() {
        if(instance == null) instance = new RetrofitWrapper();
        return instance;
    }
    private RetrofitWrapper() {}
}
```

We have a problem



We receive data from a hardware device each second onto the phone.
We have a service that reads the data and saves it locally. But we also have 5 different screens that need to show the data live.

What do we do?

One Solution is..

Software University

```
public class WeatherData {

    // instance variable declarations

    public void measurementsChanged() {

        float temp = getTemperature();
        float humidity = getHumidity();
        float pressure = getPressure();

        currentConditionsDisplay.update(temp, humidity, pressure);
        statisticsDisplay.update(temp, humidity, pressure);
        forecastDisplay.update(temp, humidity, pressure);
    }

    // other WeatherData methods here
}
```

Grab the most recent measuremets by calling the WeatherData's getter methods (already implemented).

Now update the displays...

Call each display element to update its display, passing it the most recent measurements.

One Solution is..

Software University

A.	We are coding to concrete implementations, not interfaces.
B.	For every new display element we need to alter code.
C.	We have no way to add (or remove) display elements at run time.
D.	The display elements don’t implement a common interface.
E.	We haven’t encapsulated the part that changes.
F.	We are violating encapsulation of the WeatherData class.

Commonly Used Design Patterns

Software University

- The Observer Pattern
  - defines a one-to-many dependency between objects so that when one object changes state, all of its dependents are notified and updated automatically.

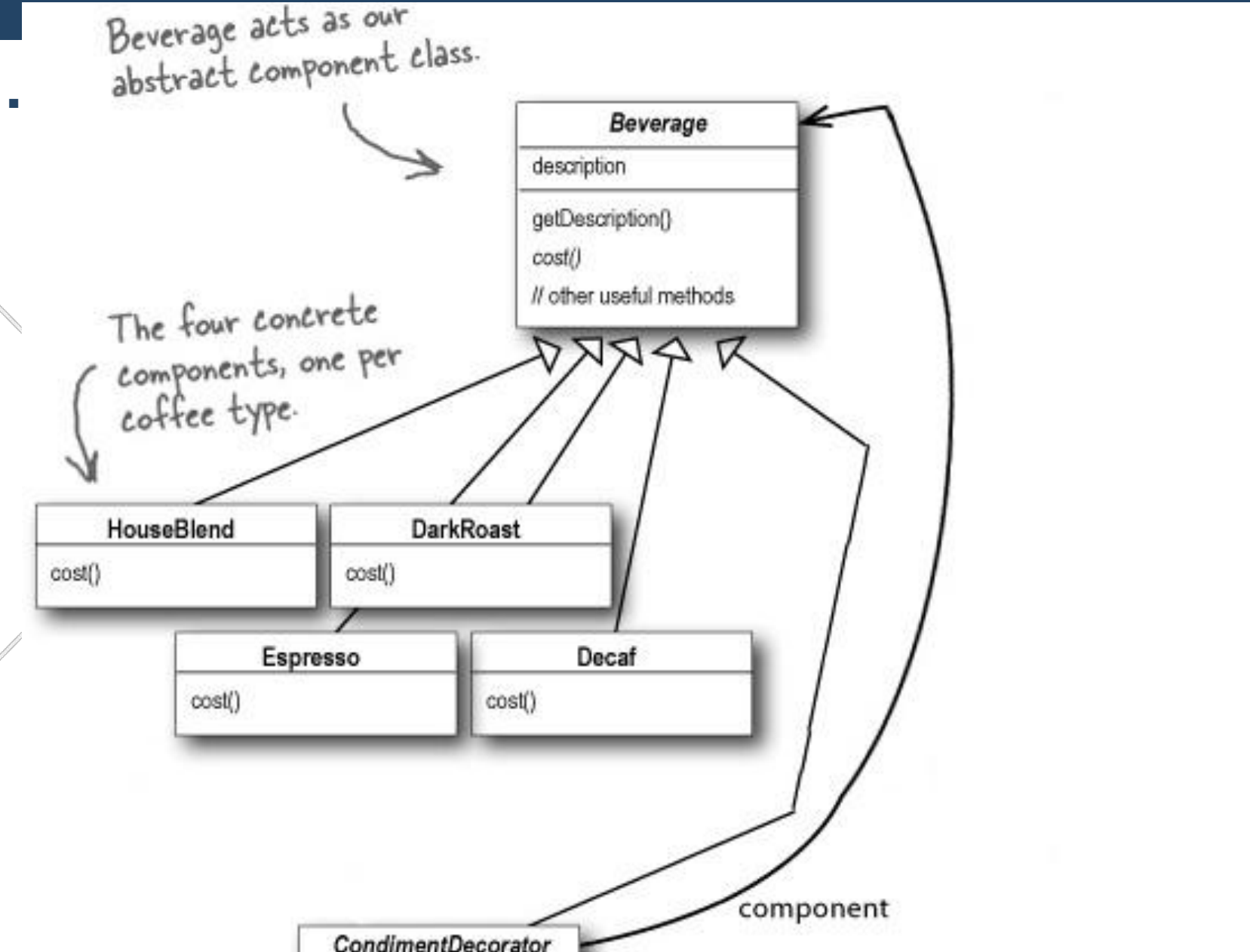
```
class WeatherData {
    interface DataListener { void onDataChanged(Data data) }

    private List<DataListener> observers;
    public void addObserver(DataListener observer) { observers.add(observer); }

    private newWeatherDataArrived(Data d) {
        for(DataListener dL : observers) dL.onDataChanged(d);
    }
}
```

Commonly Used Design Patterns

Software University

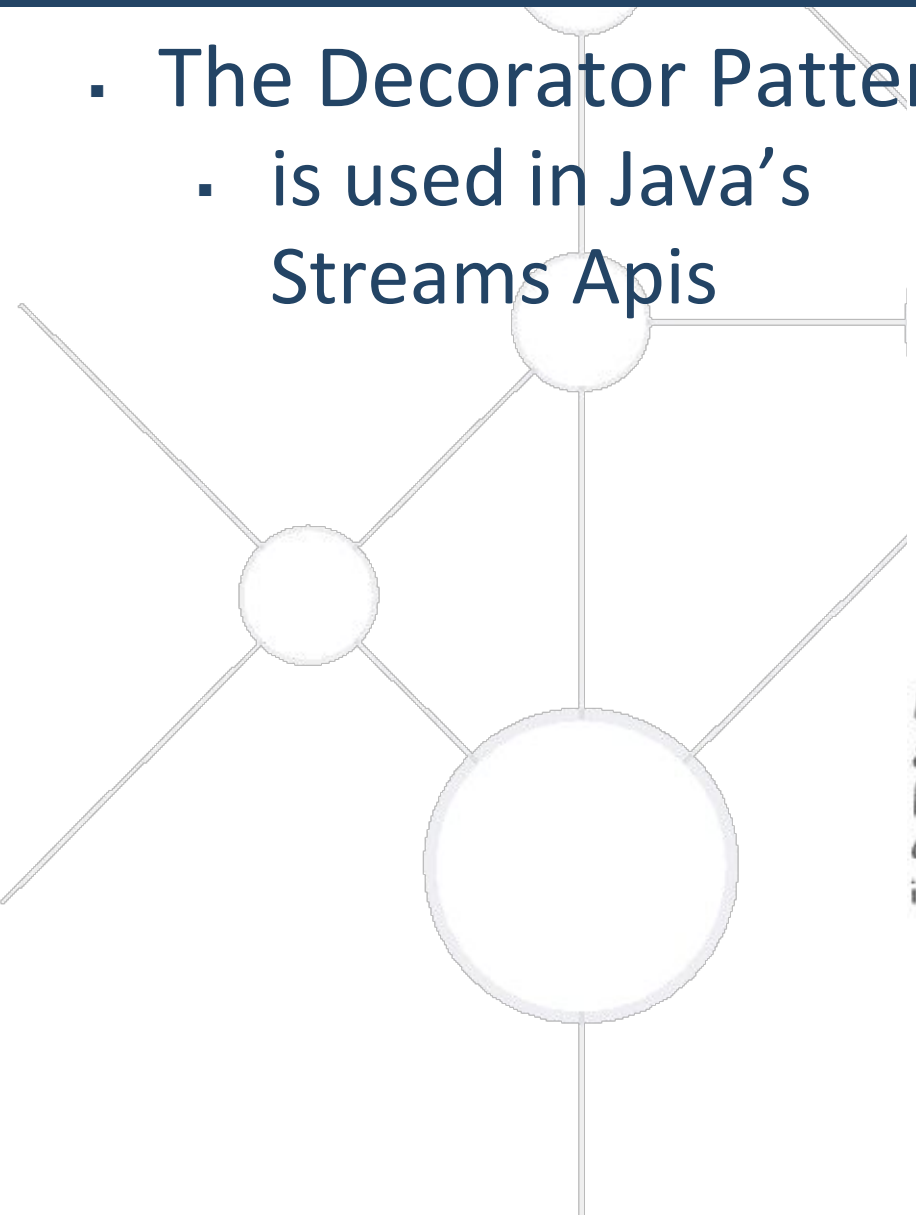


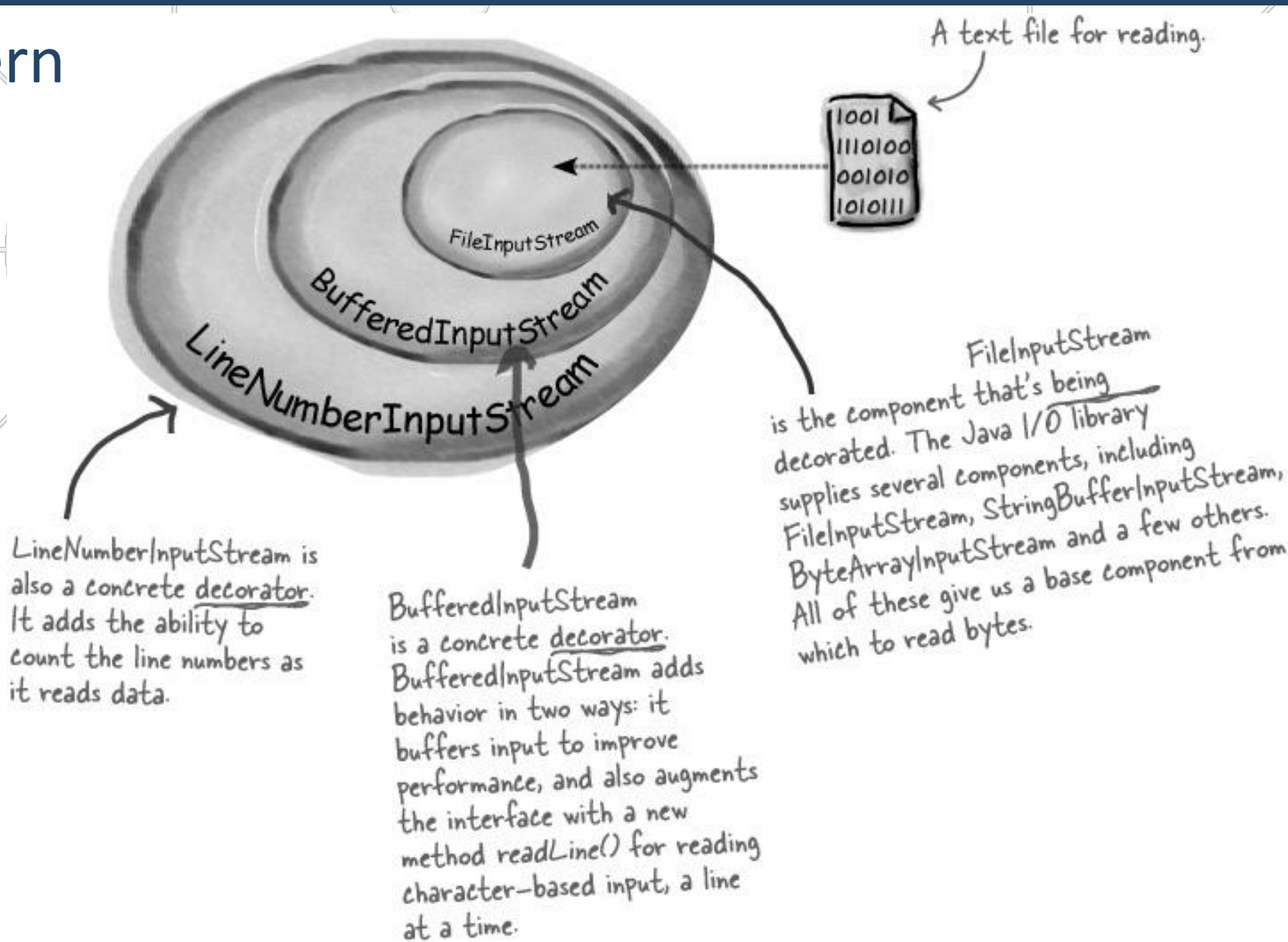
en classes

Commonly Used Design Patterns

Software University

- The Decorator Pattern
  - is used in Java's Streams Apis





Commonly Used Design Patterns

Software University

- The Factory Pattern
  - Used when creating complex objects

```
supportFragmentManager
                .beginTransaction()
                .replace(R.id.grpContainer, fragment)
                .addToBackStack("tag")
                .commit();
```

Design Principles

Software University

- General guidelines to follow when writing code
- They are not as specific as the design patterns
- They are not specific to a language

Software University

Design Principle 

Program to an interface, not an implementation.

Software University

Design Principle 

Favor composition over inheritance.

Design Principles

Software University

DRY 

Don’t repeat yourself.

Design Principles

Software University

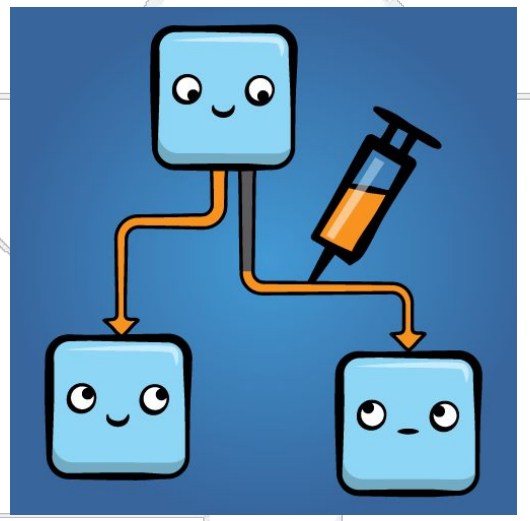
SOLID 

SRP Single Responsibility Principle
OCP Open/Closed Principle
LSP Liskov Substitution Principle
ISP Interface Segregation Principle
DIP Dependency Inversion Principle

Dependency Injection

Software University

- Начин за създаване на обекти
- Външно тяло създава обектите и ги дава на класовете, които искат да ги използват
- Позволява по-голямо ниво на абстракция и имплементиране на различни архитектури
- В андроид се използва библиотеката Dagger



```
class WeatherData {

    @Inject WeatherService weatherService;
}
```

Demo

Dagger

Android Architecture

Software University

- Архитектурата на едно приложение е зависимостта между отделните му компоненти и как те са разпределени в пакети и модули
- Всяко приложение има архитектура, дори да не сте помислили за нея.
- Има различни готови шаблони, които може да използвате, нагласяйки ги спрямо вашите нужди

Android Architecture

Software University

- MVP
  - Състои се от три основни части - Model, View, Presenter
  - Разделя логиката от презентационния слой
- Model - слоя с данните (от интернет, от локалната база, от блутуут у-во)
- View - обикновено е Activity, Fragment или View
- Presenter - клас, който взима данните от различни модели и контролира едно View
  - Във View-то не трябва да има никакъв код, който да променя състоянието му, всичко това се решава и прави от Presenter-а

Android Architecture

Software University

MVVM

- Състои се от три основни части - Model View ViewModel
  - Основната разлика с MVP е че докато при MVP презентъра директно казва на View-то какво да прави, при MVVM View-то само се абонира да слуша ViewModel-а. Т.е. ViewModel-а не знае за View-то. Това намалява нуждата от използване на интерфейси за комуникация между слоевете.
- При MVVM потребителя на данните знае за създателя им, но обратното не е вярно
- ViewModel-а позволява View-та да се абонират не само за евентите му, но и за състояния

Android Architecture

Software University

Onion / Clean code architecture

- Не толкова архитектура, колкото поредица правила, които ако се спазват водят до чист и подреден код
- Основната черта на архитектурата е съществуването на Usecases - малки класове с бизнес логика, правещи само 1 нещо
- Логиката за един екран се държи от т.нар. Интерактори, които пък вътрешно използват множество usecases за да постигнат желания резултат

Next Level Android

Software University

- View Binding -> Data Binding
- Kotlin
  - Safety
  - Co-routines
  - Lambdas
- Android JetPack
  - LiveData
  - ViewModel
  - Navigation Manager

Demo

Sample Exam

Summary and Resources

Software University

Thinking before coding is good. Do it.

Resources:

Design Patterns: https://www.youtube.com/watch?v=Av8HCXwsswg

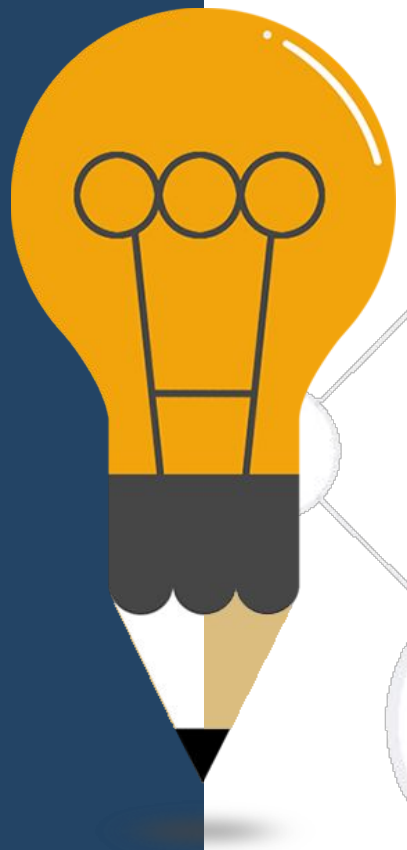
Clean code: https://www.youtube.com/watch?v=3vONlcmG8Uc

Clean code 2: https://www.youtube.com/watch?v=9jopYq8FDvo

SOLID: https://www.youtube.com/watch?v=5gli477L5w0

Clean Code book: https://www.investigatii.md/uploads/resurse/Clean_Code.pdf

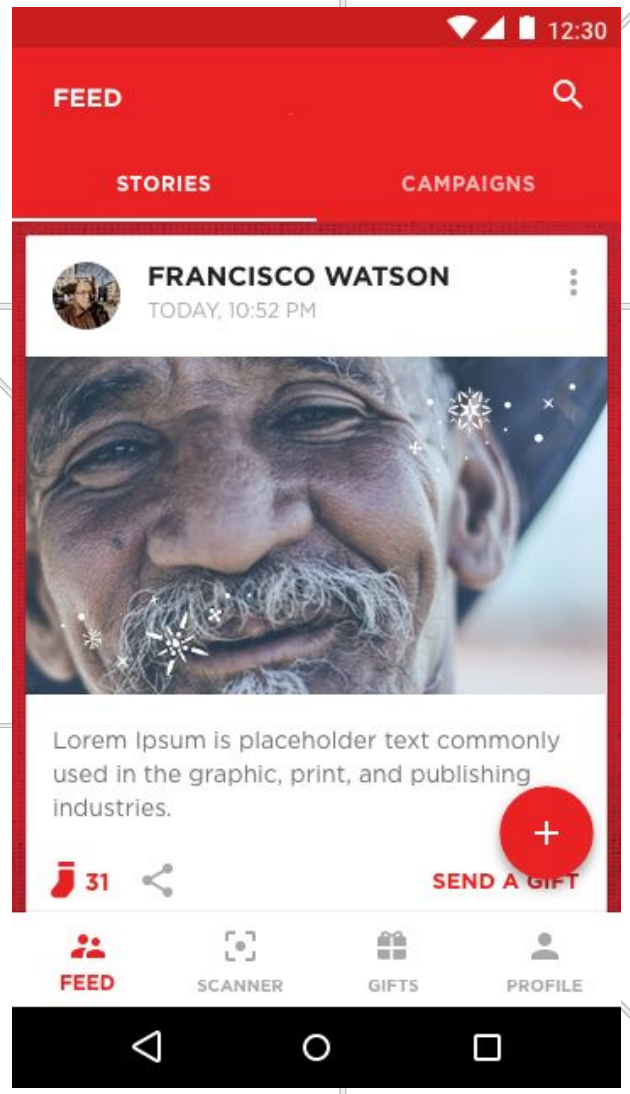
Homework (1)



Create an app with the following design.

It should have:

- Toolbar with action. Pressing the action should open a context menu.
- Bottom Navigation with Fragments
- The Feed page should have tabs
- The Stories tab should have a recycler view and a FAB
- Each element of the recycler view should be a card view
- Pressing the + FAB will open a new page where inforamtion can be added:
  - User image and name
  - Story image and text
  - Add Story button, upon pressing it the story is added to the local database and it's shown on Feed > Stories
- The scanner, gifts and profile screens can be empty
- The campaign screen can be empty



You will find resources for the design (icons) in the github of the course. (https://github.com/android-soft-uni/03-Views-Layouts)

Questions?

SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

СофтУни диамантени партньори

Software University

INDEAVR
Serving the high achievers

INFRAGISTICS®

SoftwareGroup
doing it right

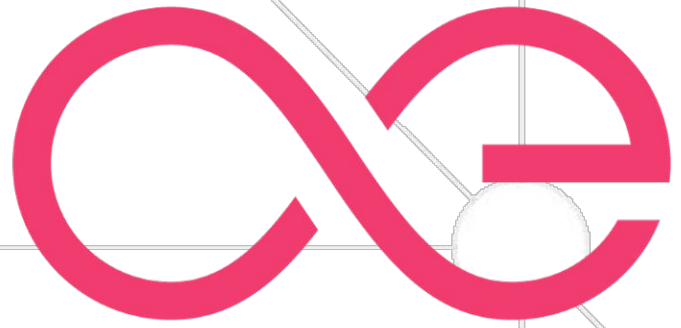
xssoftware

NETPEAK

SUPER HOSTING .BG®

СофтУни диамантени партньори

Software University



Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
- Software University Foundation
  - http://softuni.foundation/
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

SoftUni Foundation