Postman Pattern

Advance Observer pattern for Mobile

Jiahao Liu

Po ern

Ad obile

Index

- Design Patterns
- Postman pattern
- Demo
- Do’s/Don’ts
- Questions

Design patterns

[design pattern] is a general reusable solution to a commonly occurring problem within a given context in software design.

– *Wikipedia*

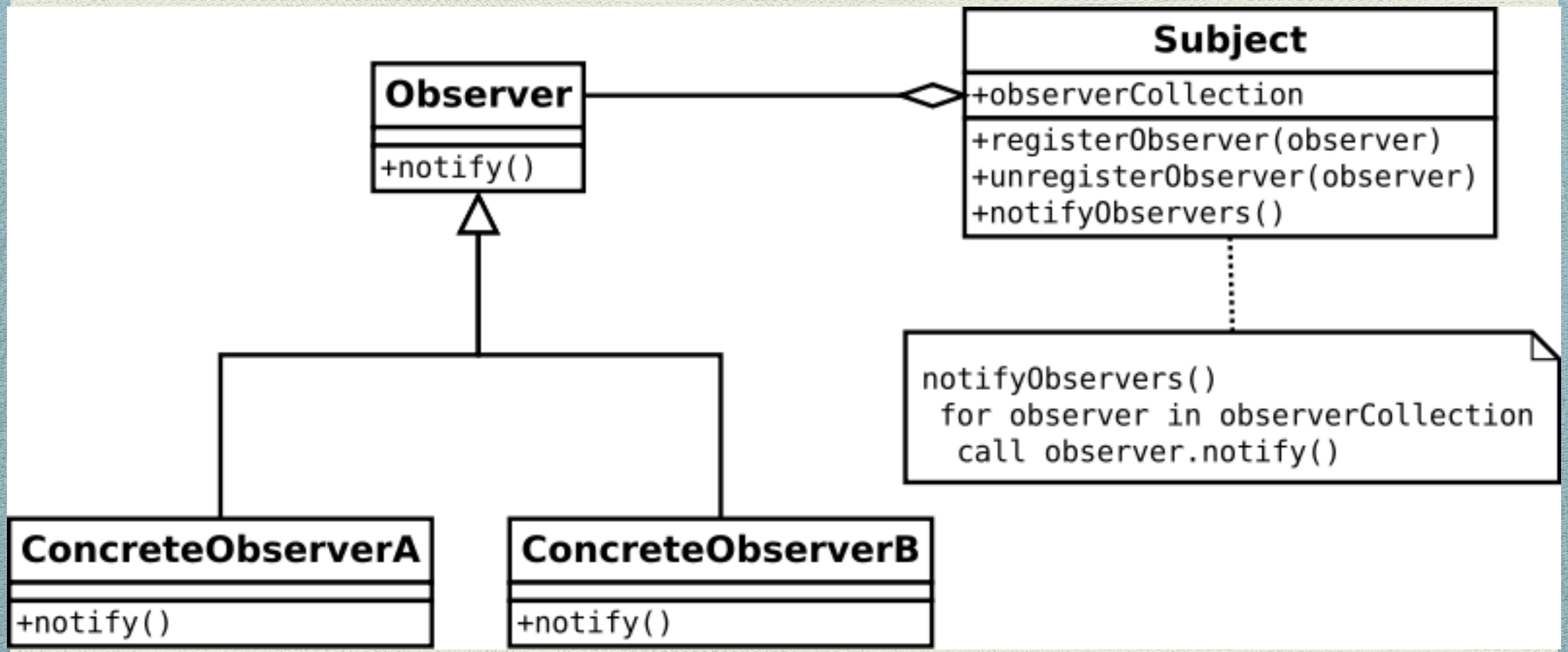
Do not reinvent the wheel

Common Design Patterns

- Creation
  - Singleton
  - Builder / Factory
- Behaviour
  - Observer
- Structural
  - Adapter

Postman pattern

Observer pattern



Problem with mobile

- UI screen as observer
- Observable could return data in any moment
  - UI screen could not be in foreground
  - The UI screen could be killed by the O.S.

Real world example

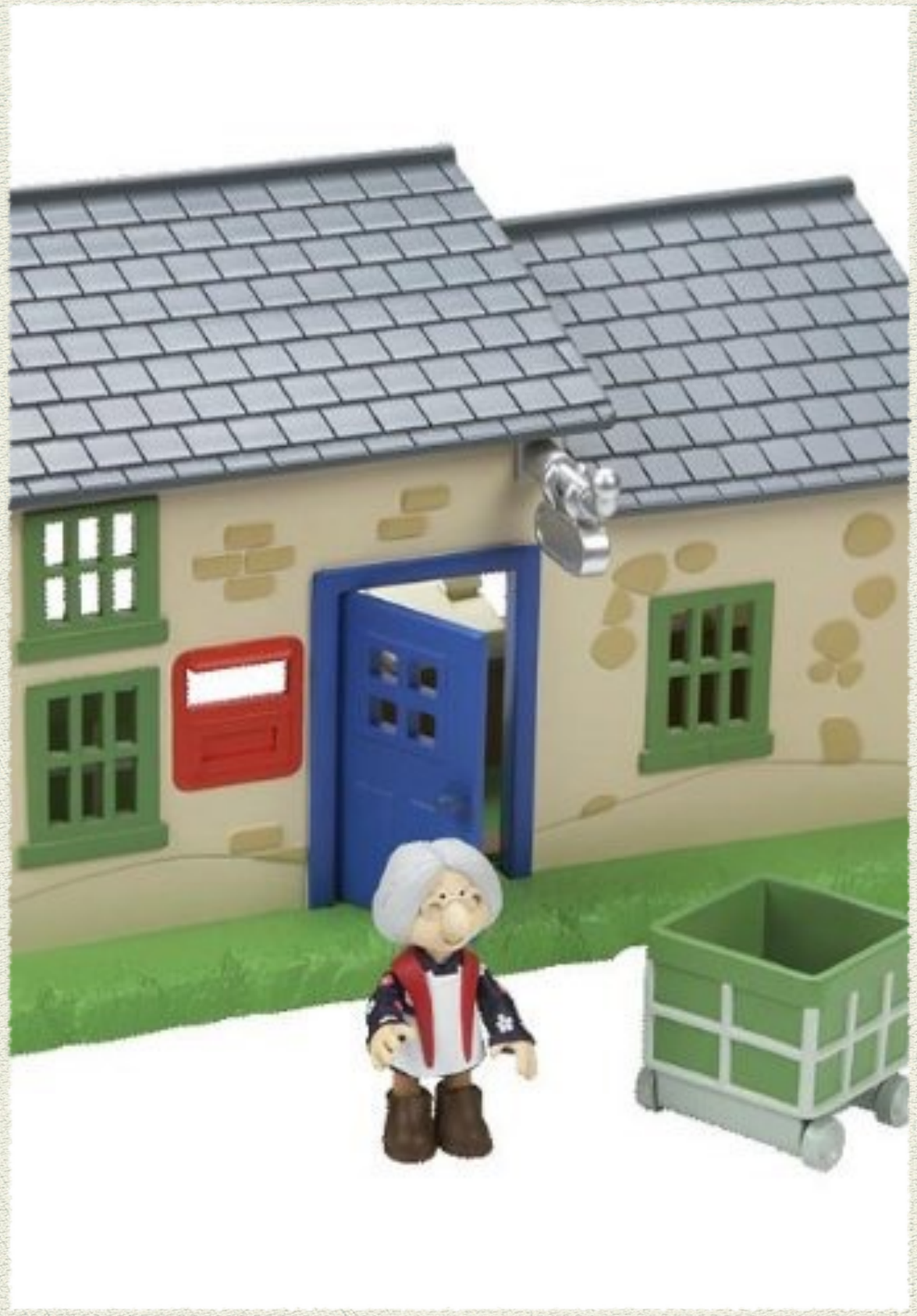
Postman

Request

Imagine you have a shop and you request a package

Preparation

The sender takes some time to prepare the package and send it to the post office



Reception

When the postman delivers the package to your shop, the shop could be open or close

Shop open

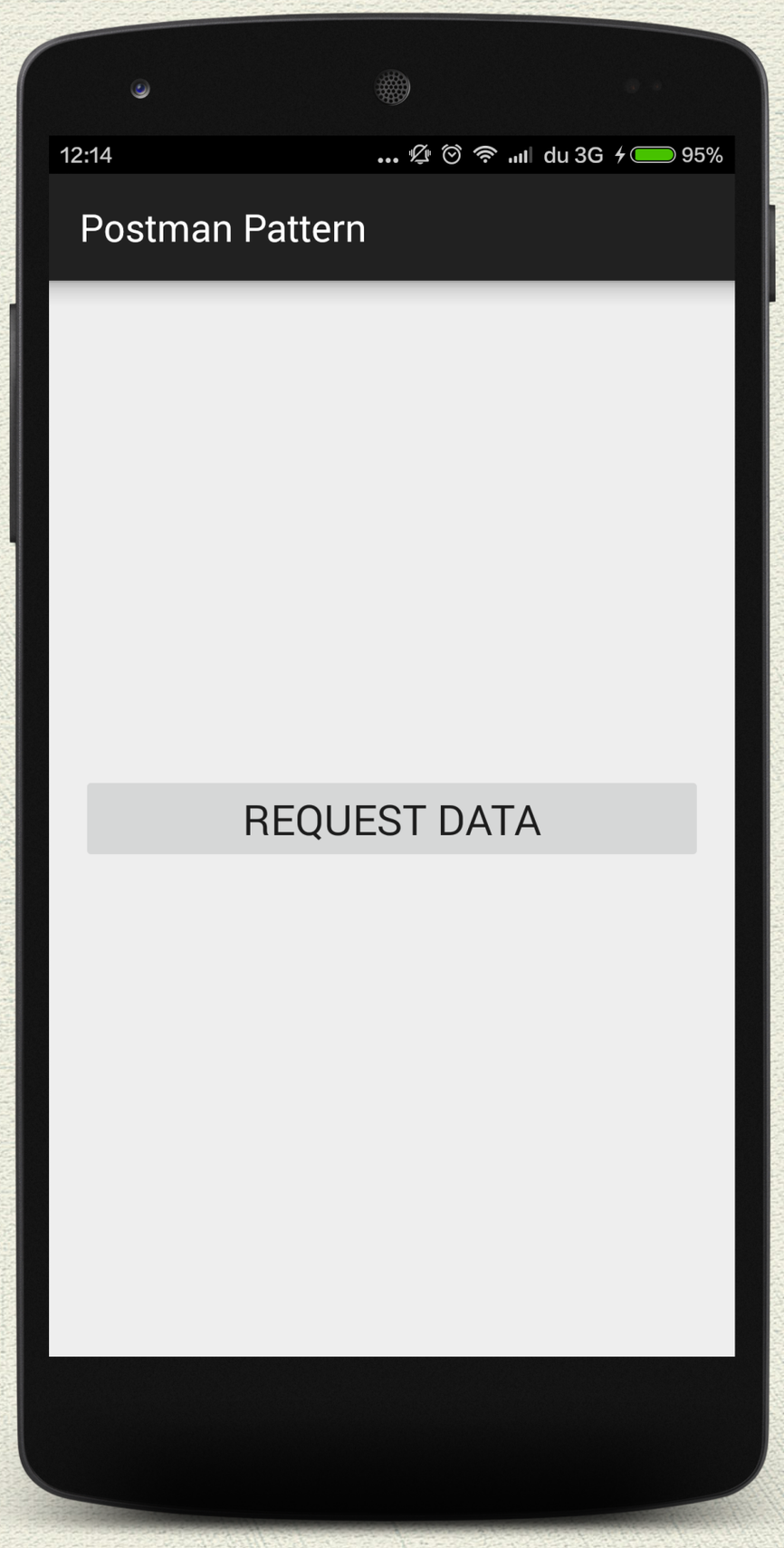
Direct deliver

Shop closed

Just leave the package in the mail box.

When the shop opens again, the package will be checked

Demo



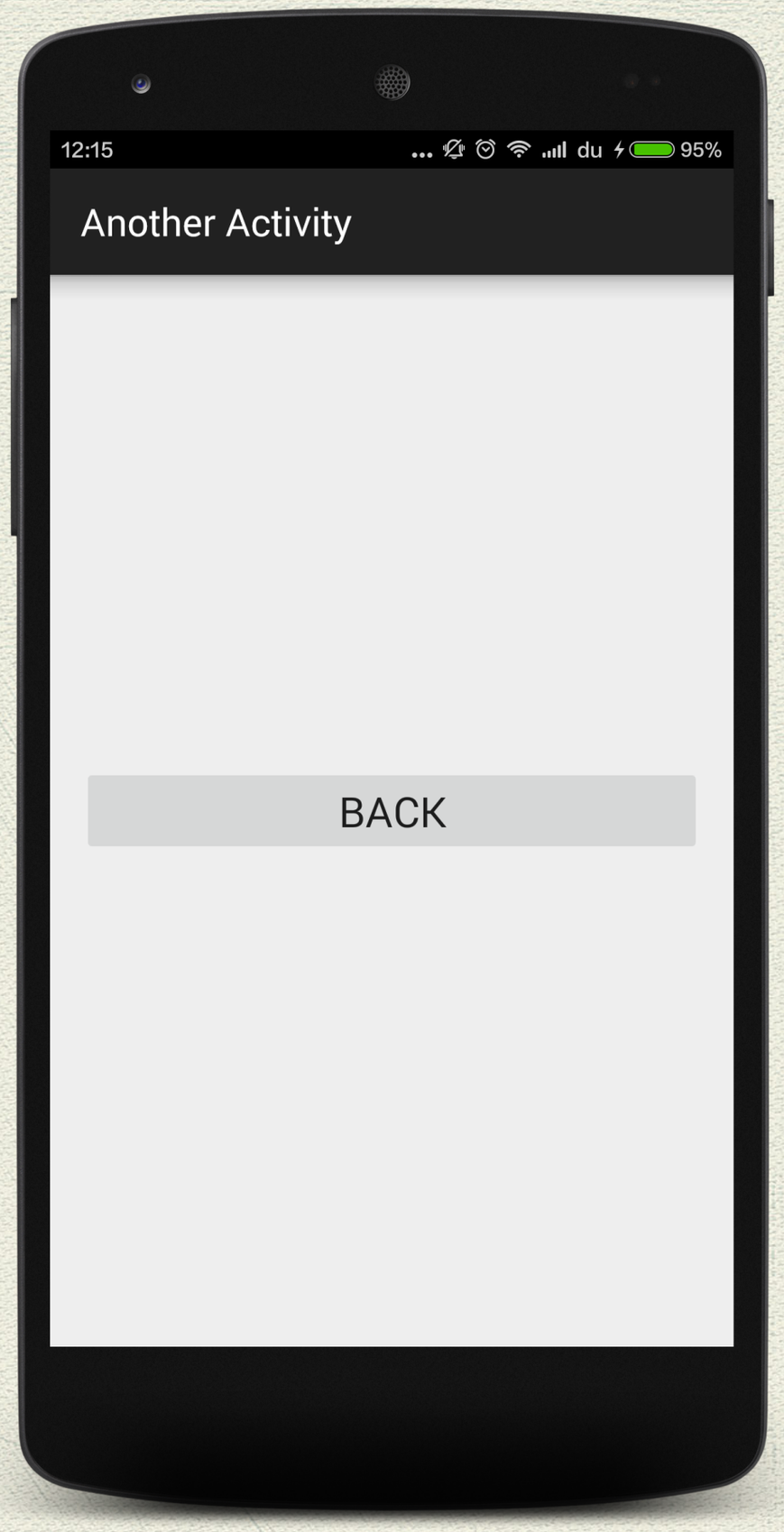
Main screen

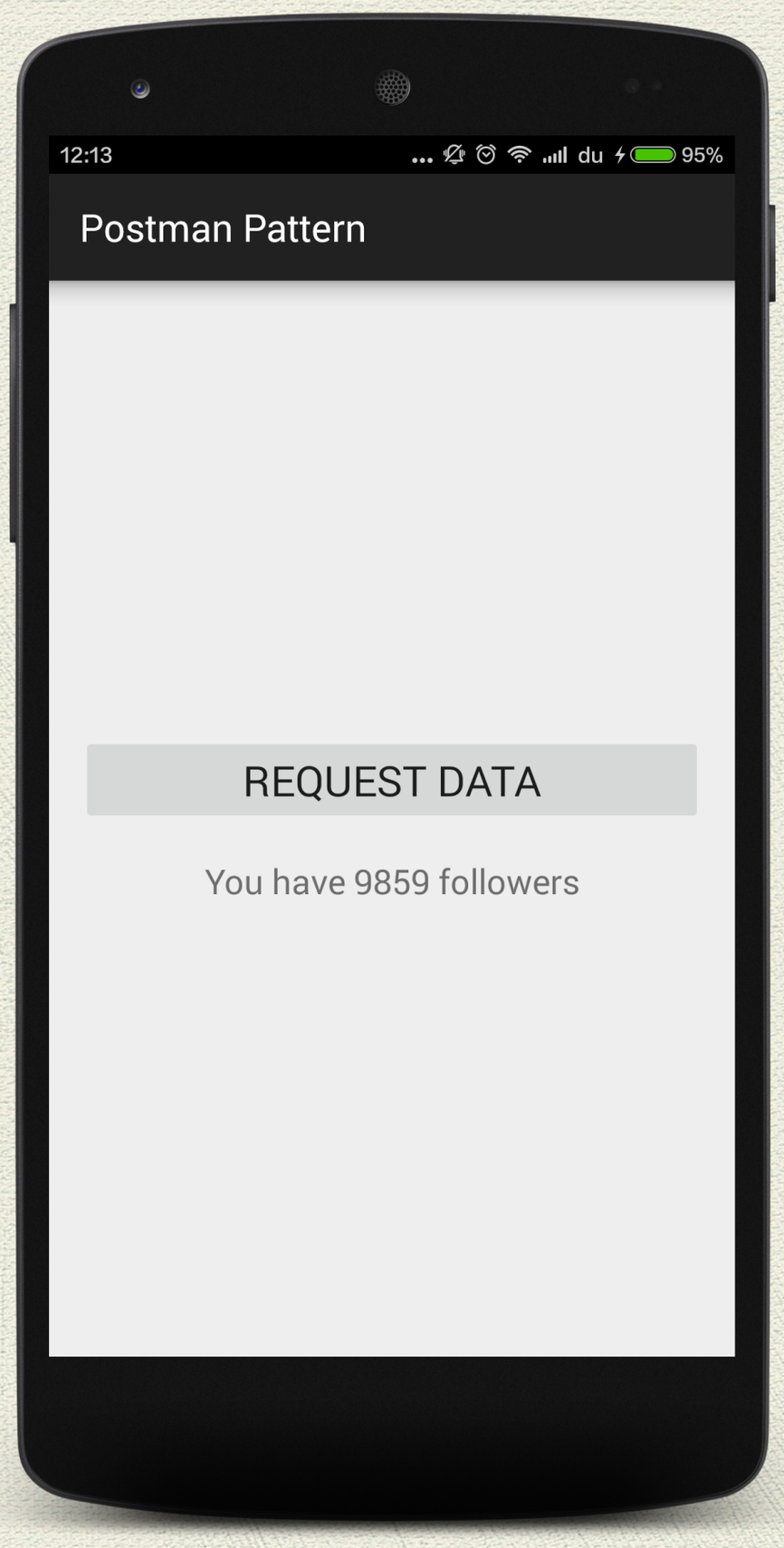
- Request data to PostmanObservable

- Launch another screen

Second screen

Click on back to finish





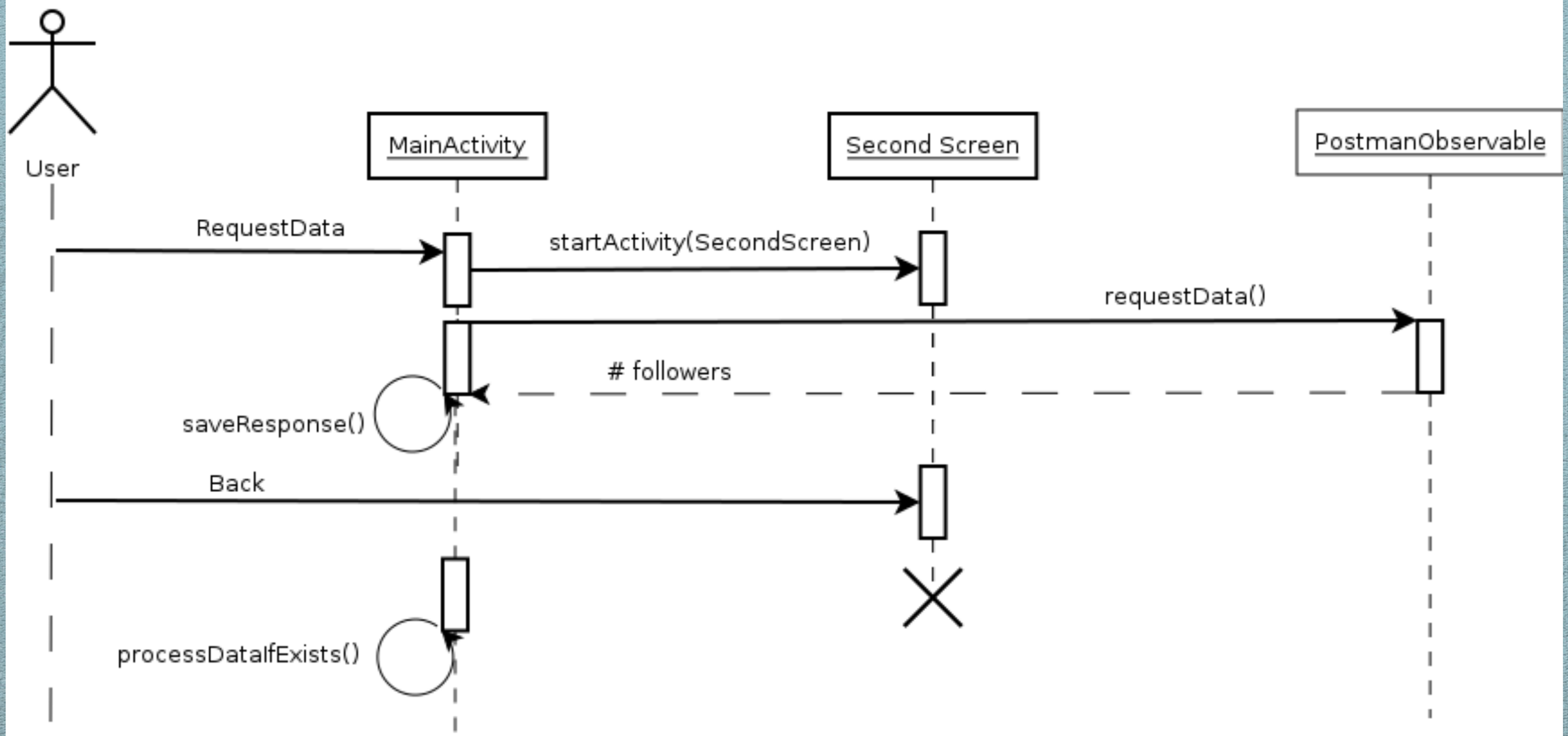
Main screen

- Data received but not in foreground

- Save data

- Refresh UI when it is in foreground

Sequence diagram



Sample code

https://github.com/jiahaoliuliu/PostmanPattern

Do's/Don'ts

Do's

- UI need async data
- Continues updates of data

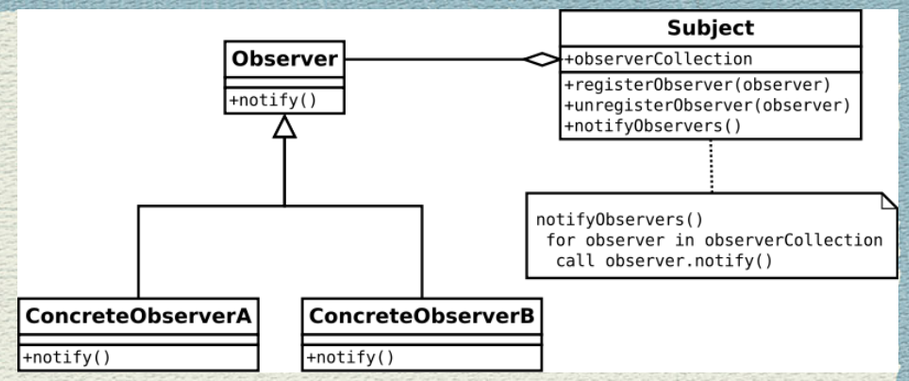
Don’ts

- Not UI implicated
  - Callback
  - Observer pattern
- Synchronous data request

Questions

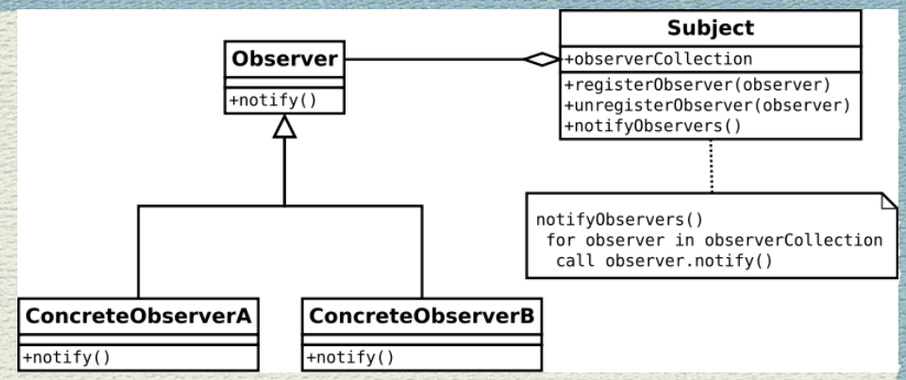
(This slide is left blank intentionally.
All the follow slides are for answer questions)

Observer



- Ask observable about data
- Notified when the data is available
- Can unregister itself

Observable



- Register observer
- Contains a list of observer
- Notify observers when data is ready

Backstage

- *Postman Activity*
- *Postman Observable*

Postman Activity

- Abstract class
- Implements Observer
- *protected boolean isInForeground*
  - Updated by *onResume()* and *onPause()*

Postman Activity

- *protected abstract void processDataIfExists()*
  - Called *onResume()*

1. If the data does not exists, finish
2. Otherwise, process the data
3. And remove the data

Activities

- Extends from PostmanActivity
- Implements *processDataIfExists()*
- Implements *update(Observable, Object)*
  - *If (isInForeground) processDataIfExists()*
  - *observable.deleteObserver(this)*

Postman Observable

- Extends from Observable

1. Add Observer
2. Retrieve data
3. Notify observer
4. ~~delete observer*~~