CADT IDT

W4 PRACTICE

Observers, ListenableBuilder

Learning objectives

- ✓ Understand **notifications principles** with the Observer Design Pattern
- ✓ No AI tools allowed to solve this practice

How to submit?

- ✓ **Create a new GitHub repository** W4-PRACTICE with the starter code
- ✓ Then commit your different exercises

How to work?

```
1 - Implement the view small components (generic or screen local)

2 - Implement the view data layout

3 - Work on the view data (initial status: initState () - getters

4 - Work on the actions
```

Resources

OBSERVER PATTERN
https://refactoring.guru/design-patterns/observer
LISTENABLE BUILDER
https://docs.flutter.dev/learn/pathway/tutorial/listenable-builder

LOCAL OR GLOBAL STATES
https://docs.flutter.dev/data-and-backend/state-mgmt/ephemeral-vs-app

TASK-001 – The Color App

💡 For this exercise, we provide the **starter code**:

`/lib/1_color_app`

Q1 - Adapt the code to remove all state drilling and use a **ChangeNotifier** and **ListenableBuilders**

- ✓ You can create a ColorService which extends or implements **ChangeNotifier**
- ✓ You can instantiate this color service **globally**

Q2 – BONUS – Extensible Approach

The color statistics app should be designed to support **easy extension** when introducing **new colors**.

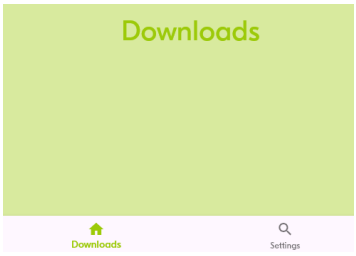
- ✓ You can – *as example* - organize the app data around a unique `map<ColorType, int>`
- ✓ Whenever a **new color type is added**, the views shall handle this color, without any code update

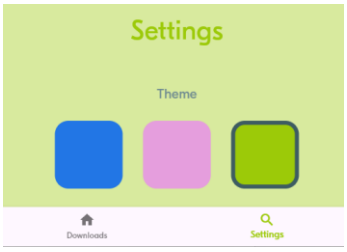
TASK-002 – Downloader – Color Theme

💡 For this exercise, we provide the **starter code**:

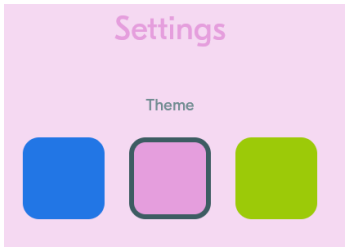
`/lib/2_download_app`

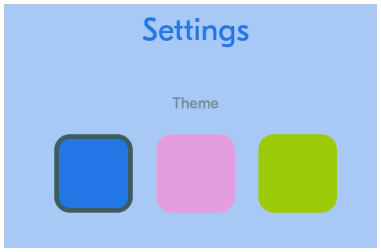
The downloader app is composed of 2 views (the downloads and settings).

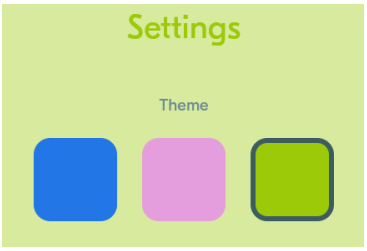




For this exercise, you will work on the settings, which allows user to change **dynamically the app color theme**:







⚠️ Theme color is composed of:

- A main color
- A background color (derived from the main color)

💡 Theme color is defined in: `ui/providers/theme_color_provider.dart`

Q1 - Adapt the code to listen to theme color changes

✓ When clicking on a color theme button, the whole app should be updated: the views colors, the menu color, etc.

TASK-002 – Downloader – Downloads

💡 For this exercise, we provide the **starter code**:

```
/lib/downloader_app
```

For this exercise, you will work on the **download page**, which allows the user to download some resources:

Each DownloadTile must support exactly **three states**:

1. notDownloaded

- Download has not started yet

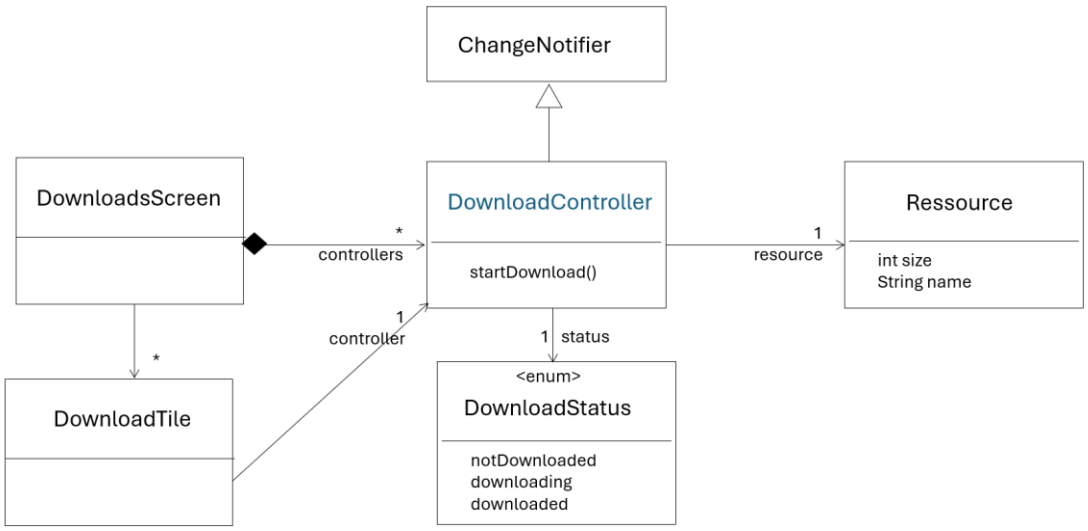
2. downloading

- Download is currently in progress
- Progress value evolves from 0% to 100%
- Progress shall also display the current downloaded size (% of the total size of the resource)

3. downloaded

- Download is complete

A – Understand the architecture



Overview of the architecture

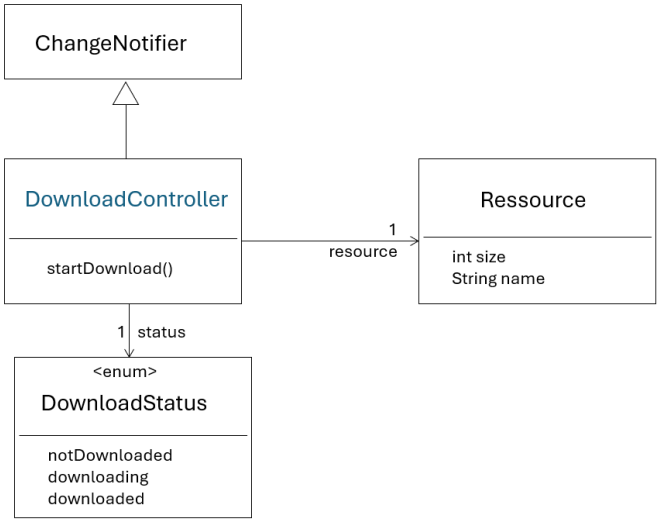
<u>We follow the controller-view architecture:</u>

- **DownloadTile** are dummy and just display the information on the related Download:

- **DownloadTile** are created with a **DownloadControler**, which contains the download data and actions:

```
class DownloadController extends ChangeNotifier {

  DownloadController(this.ressource);

  // DATA
  Ressource ressource;
  DownloadStatus _status = DownloadStatus.notDownloaded;
  double _progress = 0.0;          // 0.0 → 1.0

  // GETTERS
  DownloadStatus get status => _status;
  double get progress => _progress;

  // ACTIONS
  void startDownload() async {···
}
```

- Each **DownloadControler:**
    - is **listenable**
    - Contains the **information on the resource** to download
    - Contain the **download status** and **progress**



Q1 – Create the download tiles

- ✓ Complete the code on `download_tile.dart` to display the download information and interact with the download controller.

Q2 – Display all Download tiles

- ✓ Complete the code on `downloads_screen.dart` to display 1 download tile for each controller created

Q2 – implement the startDownload method on the controller

- ✓ Complete the code on `download_controler.dart` to start the download

1 – set status to downloading
2 – Loop 10 times and increment the download progress (0 -> 0.1 -> 0.2)
Wait 1 second :

```
await Future.delayed(const Duration(milliseconds: 1000));
```

3 – set status to downloaded