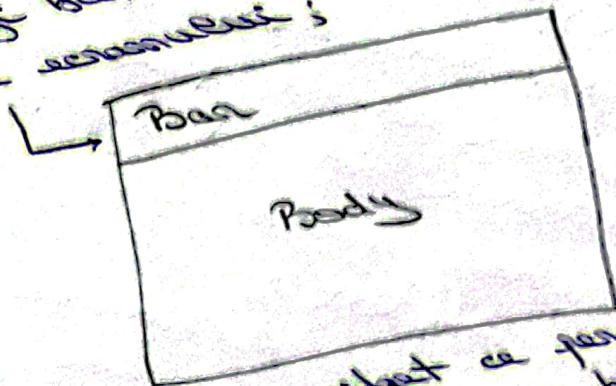


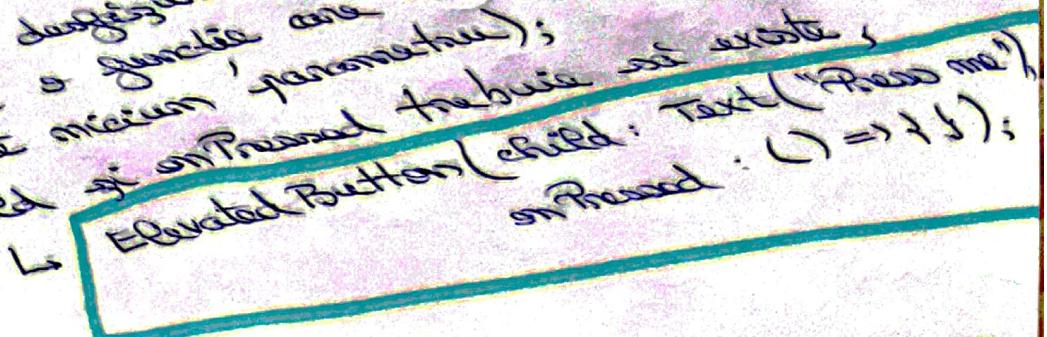
## TPPM EXAMEN

### Flutter

- widget ( $\hookrightarrow$  nice element case apres le screen);
- StatelessWidget ( $\hookrightarrow$  widget de n'importe éléments statique ou non via schéma ( $\circ$  font,  $\text{un text etc.}$ ));
- StatefulWidget ( $\hookrightarrow$  widget de certains éléments ne pas schéma (de exemple un text case se schéma le personnel programmeur));
- Widget build (BuildContext context)  $\hookrightarrow$  actualiser schéma via paramètre context (BuildContext context);
- StatelessWidget ( $\hookrightarrow$  élément statique case via apres le screen);
- (et partie StatelessWidget, si StatefulWidget);
- MaterialApp (Widget? home)  $\hookrightarrow$  home définie
- widget principal case va être object le screen;
- Scaffold ( $\hookrightarrow$  widget ce permet créer une interface view case case à base view (sous forme de buttons, text etc.) si un body son parent est de classe's



- ElevatedButton ( $\hookrightarrow$  widget ce permet appuyer sur un bouton ( $\circ$  child pour un bouton)); si une touche et un bouton pressé : ( $\Rightarrow$  if) (actionneur case et la désignation child est également bouton, on casse cette case à fonction et une autre case pour la touche);
- ElevatedButton (child: Text("Press me"), onPressed: () => {});



- Center ( $\Leftrightarrow$  widget  $\Leftarrow$  permet de spécifier une ou les positions en centre par rapport aux dimensions parentes)
- Sized Box ( $\Leftrightarrow$  widget  $\Leftarrow$  permet d'ajouter une taille fixe)
- Row ( $\Leftrightarrow$  widget  $\Leftarrow$  permet d'ajouter des éléments dans un flux horizontal)
- Column ( $\Leftrightarrow$  widget  $\Leftarrow$  permet d'ajouter des éléments dans un flux vertical)
- Container ( $\Leftrightarrow$  widget  $\Leftarrow$  permet d'ajouter des éléments génériques)
- Positioned ( $\Leftrightarrow$  widget  $\Leftarrow$  permet de spécifier une position par rapport au container)
- FractionalSized Box ( $\Leftrightarrow$  widget similaire à Sized Box mais en pourcentage du total ou des autres widgets qui peuvent prendre la dimension des dimensions parentes)
- Column Row ( $\Leftrightarrow$  widget  $\Leftarrow$  permet d'ajouter des éléments dans un flux vertical)
- TextFormFieldField ( $\Leftrightarrow$  widget  $\Leftarrow$  permet d'éditer un texte)
- InputDecoration ( $\Leftrightarrow$  widget  $\Leftarrow$  permet d'ajouter des décos au texte)

- can also specify atribute series `TextFormField`;
- `TextStyle` ↳ widget possess a reference since form de style (color, font etc.);
  - `StatefulWidget` provide application context since object `State`;
  - ↳ `State < MyApp > createState() => MyAppState();`  
`class MyAppState extends State<MyApp> {`  
`② override`  
`Widget build(BuildContext context) =>`

- `AppBar` ↳ widget possess an application bar (as proprietario title, body etc.);
- `setState()` ↳ propagate a function since it receives de instructions due's call from its own build state-up;
  - ↳ `setState(() {`  
`value = Random().nextInt(100);`  
`});`

- `setState "forget"` suplementa metodo build =>  
`redrawArea;`
- `Timer` ↳ object possess a sete for timer's duration;
- `Duration` ↳ object possess a sete & duration's value from de schimbam storage unui widget, daq nu avem un button / un widget care se permita degenera unei functie care se contine `setState`, o putem face din constructor;
- ↳ `MyAppState() {`  
`Timer.periodic(Duration(seconds: 1),`  
`(t) => setState(() {`  
`value++; }));`

- CustomPaint → widget pentru a desenă pe un  
nigă primitive basic pînă intermediar  
carnă sau poate desenă oare (arc, imagine,  
pentru etc.);
- ↳ trebuie să contine proprietate  
painter (este o clasa ce să va avea  
de dezvoltat specific casă și să spăle  
pe seama) și child (pentru un  
container să nu își specifica deosebită);  
Containerul să nu își specifică deosebită  
trebue să conțină
- clasele date la paint sau trebue să conțină  
metodele paint și shouldRepaint;

```

    ↳ class < myPainter > extends
      CustomPainter {
        @override
        void paint(Canvas canvas,
                  Size size) {
          ...
        }
        @override
        bool shouldRepaint(CustomPainter
                             oldDelegate) {
          ...
        }
      }
  
```

- shouldRepaint ↔ verifica dacă desenul să  
schimbeți și dacă trebuie redesenat;
- Sfert ↔ obiect pentru o reprezentă un flutur;
- Imagine ↔ widget pentru o imagine  
dintr-un stream/din memorie / dintr-un asset,  
(în casă asset, trebuie dat path-ul selectiv  
de la baza aplicației) și este să  
difereți subtipuri .gifs;
- ↳ Image.asset('assets/images/lion.jpg'),  
Image.asset(), Image.gif(), Image.memory(),
- Image.asset(), Image.gif(), Image.memory(),

- Image.network();
- Image functionality of we aggiungi;
- icon  $\leftrightarrow$  our glyph (as our character), a imagine;
- as a unique culture of our background transparent;
- as a unique existence of our icon;
- IconData icon  $\leftrightarrow$  Icon.\*\*\* value exists;
- Icon  $\leftrightarrow$  widget featuring a image our icon;
- (triebie si abbi un IconData? icon);
- ImageIcon  $\leftrightarrow$  widget featuring a we our icons;
- $\hookrightarrow$  are as appropriate ImageProvider (Object? image) (de example AssetImage (string image));
- asset Path);
- TextButton  $\leftrightarrow$  widget featuring a representa our button case our dear text;
- OutlinedButton  $\leftrightarrow$  widget featuring a representa our button our text of colors;

Elevated!

Text

Outlined

- differentia distinctive all 3 + widget will feature because note to our code (our want decorative);
- button disabled  $\leftrightarrow$  offpressed, null;
- ElevatedButton.icon, TextButton.icon, OutlinedButton.icon  $\leftrightarrow$  feature a specific outcome or icons;

Elevated.icon (  
offpressed: () => {},  
icon: Image.asset ("assets/images.png"),  
label: Text ("Button text"))

- label si icon promise widget will  $\leftrightarrow$  for example precedent due feature imagined to label of textul de icon, the we agree the icon immediately, going to be maintained

- mai suntă butoane și după imaginile;
- dacă vrem să ne găsim un buton custom mai complex pe care să-l putem adăuga așa, ne putem face o clasa care să extindă una dintr-o altă existente (de exemplu clasa MyButton extinde ElevatedButton);
- FloatingActionButton ( $\Rightarrow$  widget pentru a reprezenta un buton natural (mai ușor de personalizat));
- FloatingActionButton.extended ( $\Rightarrow$  poate fi o săcă sau buton cu scrisă și text (la fel ca la .icon, are icon și label care sunt wedgeturi);
- Spre deosebire de proprietatea floatingActionButton care este să seteze în punctul din dreapta jos;
  - $\hookrightarrow$  tipul proprietății este widget ( $\Rightarrow$  nu trebuie reprezentat ca și FloatingActionButton);
  - $\hookrightarrow$  acest widget este pus în dreapta jos;
- IconButton ( $\Rightarrow$  widget pentru a reprezenta un buton care folosește scrisă în loc de imagini);
  - $\hookrightarrow$  IconButton(  
icon: Icon(icons.warning),  
 onPressed: () => {})

- În general, IconButton este natural;
- closeButton, BackButton ( $\Rightarrow$  widgeturi pentru butoane care simulează un IconButton cu scrisă de close/back (dacă puntem scrisă);
- $\hookrightarrow$  onPressed nu mai este obligatorie;
- $\hookrightarrow$  closeButton
- Switch ( $\Rightarrow$  obiect care reprezintă starea de on/off  
(true/false));

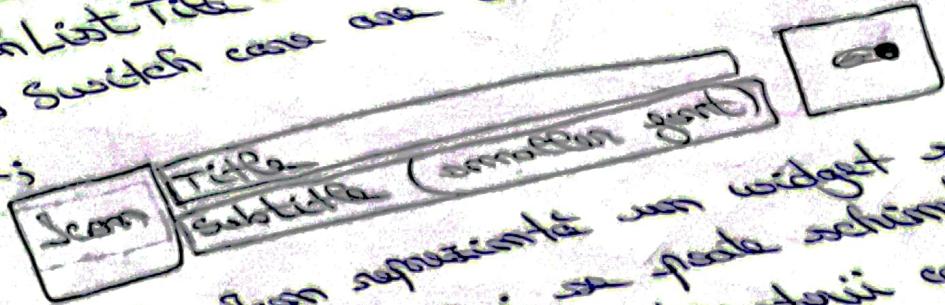
unchecked  
(`off/false`)

checked  
(`on/true`)

→ Switch  $\leftrightarrow$  widget pentru o reprezentare un obiect de tipul Switch (ace 2 sunt  $\leftrightarrow$  `on/off`/`true/false`);  
L̄ nu este un label asociat (un text care explică ce reprezintă);

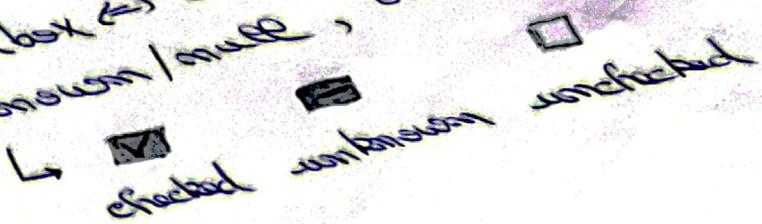
L̄ proprietatea obligatorie sunt value (`true/false`) și oncharged (privind cum se modifica valoarea în trebuință și se schimbă și se spune că parametrii noii valori);

→ Switch dezactivat  $\leftrightarrow$  oncharged: null;  
→ SwitchListTile  $\leftrightarrow$  widget pentru o reprezentare un `Switch` care are un icon, subtext și un label;  
L̄



L̄ Iată reprezentarea un widget și proprietatea Switch-ului și pe cele schimbată;  
value și oncharged sunt tot obligatorii ca să poarte subtitle și să poarte Title, subtitle și secondary pentru item (trebuie să sunt widgeturi);  
3 sunt widgeturile;

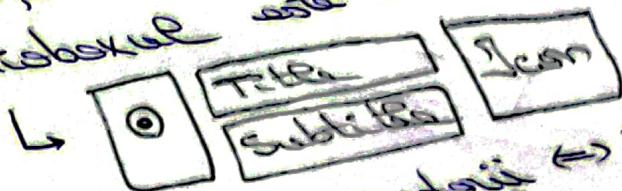
→ oncharged: null la switchListTile  $\leftrightarrow$  dezactivat  
toate elementele;  
→ checkbox  $\leftrightarrow$  obiect care poate avea 3 stări (`off/false`, `unknown/null`, `on/true`);



- checkbox  $\Leftrightarrow$  widget ce représente un objet de type checkbox (3 états possibles);
  - ↳ propriété obligatoire  $\Leftrightarrow$  value (true/false), onchanged (fonction ce la switch), triestate = false (impossible (true, false, valeur avec 3 états));
    - valeur avec 3 états avec la onchanged passe de true, false à une autre valeur de la fonction de la dernière cellule 3
- paramétrage de la value (en fonction de une autre cellule);
  - stare possibles;
  - ↳ null  $\Leftrightarrow$  absence de unknown;
  - ↳ null est true/false?
- si value est true/false;
  - checkboxListTile  $\Leftrightarrow$  widget portant à représentation checkbox case ou un icon, un titre et un subtitle;
  - ↳ Icon Title Subtitle
- checkboxListTile est exact ce ~~la~~ SwitchListTile, des valeur est bool? (si paramétrage de la onchanged);
  - ou propriété triestate (impossible false);
- radiobox  $\Leftrightarrow$  objet portant à selectionner plusieurs éléments dans un seul de valeur la valeur la moment actuel (actuel valeur va avec status checked, son état peut être unchecked);
  - ↳  (true/false) checked
    - checked (off/on)
    - checked (off/on)
  - elle ne a un label
- ce le switch qui compose n'a pas de associé;
- plusieurs groupValue portent à composer plusieurs radios;
  - ce cell des group;
- Radio  $\Leftrightarrow$  widget portant à représenter radiobox;

- ↳ proprietate obiectului  $\leftrightarrow$  T value (valoare pe care o reprezinta), T? groupValue (valoare proprietate, onChanged (functie care este apelata cand  $\Rightarrow$  un click pe un obiect); se poate scrie doar  $\Rightarrow$  un checkbox este checked si ca dim valoare, atunci acest radiobox este cu ca dim valoare).
- $\rightarrow$  dacă variabila din groupValue are valoarea respectivă ca dim valoare, atunci acest radiobox este checked;

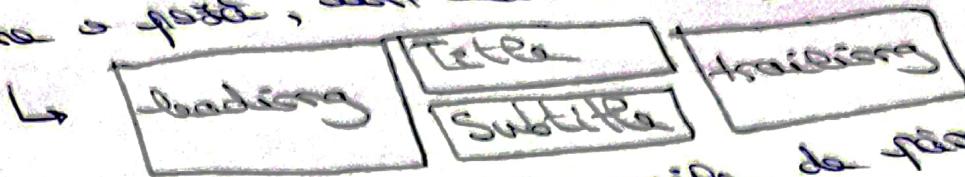
- $\rightarrow$  Radiostrip  $\leftrightarrow$  widget pentru a reprezenta un radiobox, un titlu, un subtitlu și un set de radioboxuri este în dreapta (radioboxul este în dreapta);



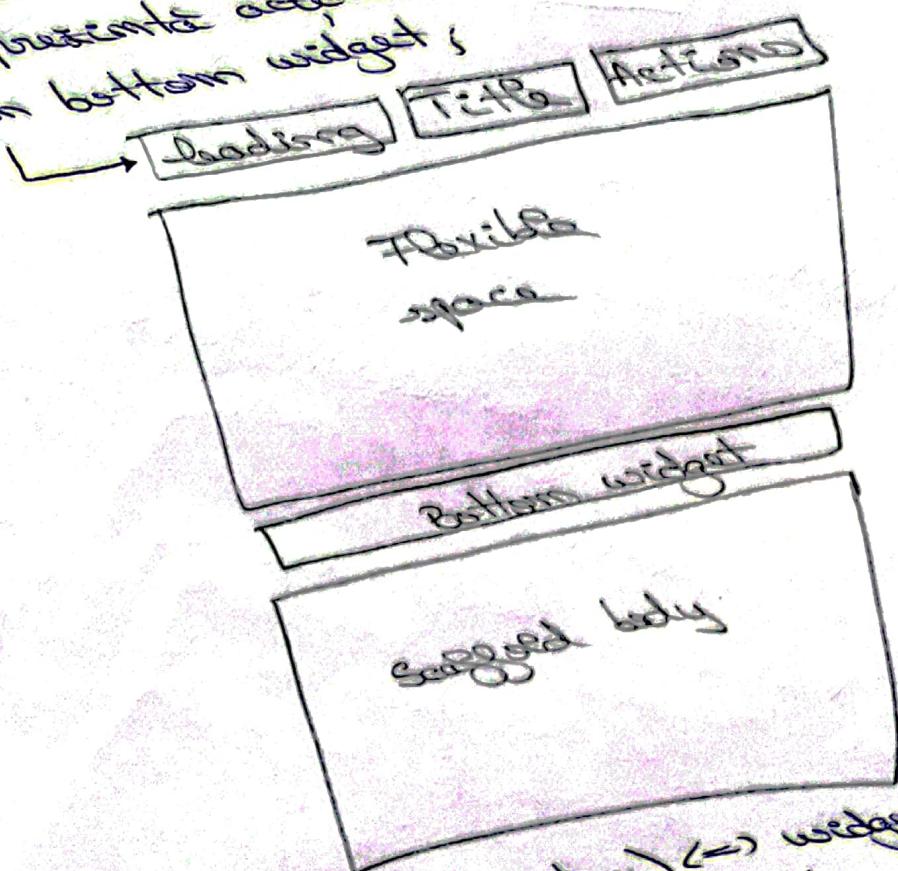
- ↳ proprietate obiectului  $\leftrightarrow$  T value, T? groupValue, onChanged (titlu pentru Title, subtitlu pentru Subtitle și secondary pentru Icon sunt widgeturi);
- $\rightarrow$  mai multe radioboxuri  $\leftrightarrow$  în Row sau Column;
- $\rightarrow$  în Radiostrip  $\leftrightarrow$  se va scrie Radiostrip  $\leftrightarrow$  avem un singur radiobox, nu o listă (ca la Switch și checkbox);

- $\rightarrow$  dacă avem 2 Radiostripuri Radio și RadioListTile ca acestea să împart valoarea respective să fie apăsește ca dim valoare, iar groupValue, să fi corespondentă cu dim valoare care trebuie să fie la valoare în ca dim valoare radio și radioListTile să fie la valoare; și deoarece radioListTile conține o componentă groupValue, nu va fi bine să scriu;
- $\rightarrow$  ListTile  $\leftrightarrow$  widget pentru a reprezenta o componentă a componentei a unei liste de itemi și care are title, subtitle, lista de itemi și care are title, subtitle,

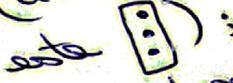
leading & trailing (wedge/tail for ease see system  
pure & fast, von icon etc.),



- généraliser pour les title-wile de films avec  
(la titre, Switch / Checkbox / Radio sont les  
leading/trailing);
  - AppBar => composante principale à unui scaffold  
qui contient un leading widget (icon, imagine etc)  
un title, actions (liste de widgets qui  
représentent actions), un espacement flexible et  
un bottom widget;



- dropdown button (combobox) ( $\leftrightarrow$ ) widget pentru a selecta o optiune din mai multe opțiuni existente (este bazat pe un template);  
↳ 2 case ( $\Rightarrow$ ) Dropdown Button (buton),  
DropdownMenuItem (sternează din lista de elemente din casă și poate fi legături);

- DropdownButton <T>  $\leftrightarrow$  obligatorii sunt schimbat (functie care ne notifică că s-a schimbat selecția), T? value (valoarea de selectat) și List<DropdownMenuItem<T>> items (lista de itemi);
- DropdownMenuItem <T>  $\leftrightarrow$  obligatorii sunt child (widget pentru forma de reprezentare, de tip Text), T? value și similar cu dropdown button, doar nu button (similar cu dropdown button, dar nu există o selecție, când este ales un item din acesta, nu operează selectat, e ca o comandă mai mult);
- Un principiu este reprezentat de o iconă sau un child widget, micădă se amânează dacă nu este dat niciunul, defaulțul este 
- PopupMenuItem <T>  $\leftrightarrow$  Widget? child, widget? icon, T? initialValue, itemBuilder (functie care returnează o listă de PopupMenuItem<T>) și Selected (functie care crează și adaugă item com
- PopupMenuItem <T>  $\leftrightarrow$  classe de bază din care se pot extinde PopupMenuItem (este implementată); derive astă widgeturi (item din meniu), checkedPopupMenuItem (item checked/nu checked) și checkmark din meniu), Aggregation PopupMenuProvider (o listă verticală care reprezintă și itemi din meniu);
- Slider  $\leftrightarrow$  widget pentru a selecta o valoare numerică dintr-un interval;

- 
- Slider  $\leftrightarrow$  required double value (valence selectable)  
required void Function (double)? unchanged (function  
call primitive valence move), double min = 0.0,  
double max = 1.0 etc.;  
double default  $\leftrightarrow$  unchanged: null;
- Slider deaktivat  $\leftrightarrow$  unchanged