DESIGN PATTERN:

- Singleton:

  Ce pattern est essentiellement utilisé pour l'implémentation d'objets desquels on veut exactement une seule instance dans notre code (d'où le nom), un exemple parfait d'usage de ce pattern est pour une base de données.

  Dans notre application nous allons utiliser une base de données qui contient une liste de tous les produits qu'on peut acheter dans un magasin (par exemple Migros), du coup à n'importe quel moment dans notre application on veut qu'une seule instance de cette base de données.

  Pour en avoir une seule instance on crée des fonctions d'initialisation de base de données qui sont privés, comme ça aucune autre classe peut en créer une instance, pour donner accès à cette base de données on crée une fonction publique get base de données qui retourne la base de données si elle existe déjà, et sinon la crée puis la retourne, on peut ainsi assurer qu'elle est créée qu'une seule fois.

  On a aussi utilisé le pattern Singleton pour la classe Famille, en utilisant la même logique.

- Abstract Factory:

  Ce pattern est utilisé pour créer des familles d'objets apparentées sans préciser leurs classes concrètes.

  Dans notre projet nous utilisons ce pattern pour nos listes (favorits/ inventaire/ courses) qui du coup utiliseront les mêmes méthodes pour afficher les items ainsi que les opérations CRUD.

  L'un des avantages de l'usage de ce pattern est la facilité d'ajouter toute une nouvelle liste (par exemple une liste de recommandées), ou encore le fait qu'il nous assure que tous les items sont compatibles entre les classes.