**Lucruri din laboratoare folosite pentru realizarea proiectului**

***Lab1: Introducere în Java (output, tipuri de valori, funcții)***

* Programul utilizează funcții pentru gestionarea tranzacțiilor financiare, inclusiv **addTransaction, calculateTotalExpenses și calculateTotalIncomes**;
* Tipuri de date standard (**String, double, LocalDate**) sunt folosite pe scară largă pentru a reprezenta entități precum utilizatori, categorii, cheltuieli și venituri.
* Output-ul este gestionat în clasa **Main**, unde se afișează informații despre tranzacții, profit/pierdere și totaluri.

***Lab2: Introducere în Java (input, for, while, switch, if)***

* Clasa **Main** utilizează instrucțiuni **switch** și **while** pentru navigarea între opțiunile meniului principal.
* Input-ul de la utilizator este procesat prin intermediul clasei **Scanner**, asigurând o interfață interactivă pentru utilizator.
* Structuri de control precum if și bucle sunt utilizate pentru logica internă a aplicației.

***Lab3: Colecții Java (Array, List, Map)***

* Clasa **ExpenseManager** utilizează o colecție de tip **List** pentru a gestiona tranzacțiile financiare.
* Metode precum stream, filter și map sunt utilizate pentru a extrage cheltuieli, venituri și pentru a calcula totaluri.

***Lab4: Clase Java (clase cu atribute și metode)***

* Clasele **User, Category, Expense**, și **Income** sunt bine definite, fiecare având atribute (ID, nume, sumă) și metode relevante (getter, setter, toString).
* Exemple:
  + **User** include atribute pentru ID, nume și email.
  + **Category** definește o categorie financiară cu nume și descriere.

***Lab5: Moștenire în Java, clase abstracte***

* Clasa abstractă **Transactions** este utilizată ca bază pentru clasele **Expense** și **Income**, evidențiind principiul moștenirii.
* Metoda abstractă **displayTransactionDetails** este implementată diferit în fiecare subclasa.

***Lab6: Interfețe în Java***

* Interfețele **IUser, ICategory, ITransactions** și **IExpenseManager** sunt implementate de clasele corespunzătoare.
* Acest lucru asigură o structură modulară și clar definită a proiectului.

***Lab7: Teste pentru fiecare metodă***

* Testele **JUnit** acoperă toate clasele principale, asigurând funcționarea corectă a acestora.
* Exemple:
  + **UserTest** verifică metodele getter, setter și toString ale clasei **User**.
  + **ExpenseManagerTest** testează gestionarea tranzacțiilor și persistența datelor.
  + **ProfitLossCalculatorTest** validează calculul profitului și pierderii.

***Lab8: Persistența datelor***

* Clasa **DataPersistence** implementează funcționalități pentru salvarea și încărcarea datelor în/din fișiere **JSON**.
* Exemple:
  + Salvarea cheltuielilor folosind **saveExpenses**.
  + Încărcarea veniturilor folosind **loadIncomes**.

***Lab9: Diagrame UML***

* Fișierele imagine **UML1.png, UML2.png, și UML.png** prezintă structura claselor, relațiile dintre ele și utilizarea interfețelor. Aceste diagrame vizualizează moștenirile, implementările interfețelor și legăturile dintre clase.