Bank Management System

Goal:

Students will build a fully functional bank system with loan functionality using:

1. Arrays
2. Control structures
3. Methods
4. Classes & objects
5. Inheritance and interfaces
6. Exception handling
7. File I/O

WEEK 1: FOUNDATIONS WITH ARRAYS

Students Know:

- Variables, control flow (if, switch, loops), arrays, methods

Tasks:

- Build a **menu-driven CLI skeleton** using switch:

```
1. Register User
2. Login User
3. Exit
```

- Store user data in **parallel arrays** (String[] usernames, String[] passwords) ▪ Create foundational methods: registerUser(), loginUser()
- Add balance tracking with double[] balances

Concepts: Arrays, switch-case, loops, basic validation

WEEK 2: EXPANDING CORE FEATURES

Tasks:

- Expand the CLI menu:

```
4. Deposit Money
5. Withdraw Money
6. Show Balance
7. View Account Details
```

- Build deposit/withdraw functionality with proper validation
- Introduce **2D arrays** for user data: String[][] userDetails
- Replace parallel arrays with structured data storage

Concepts: Multi-dimensional arrays, data organization

WEEK 3: STRING HANDLING & VALIDATION

Tasks:

- Apply **String methods** for:
  - Password validation (length, complexity requirements)
  - Email validation (must contain @)
- Input trimming and formatting
- Enhance user experience with robust input handling

Concepts: String manipulation, validation patterns

WEEK 4: INTRODUCTION TO CLASSES & OBJECTS

Tasks:

- Create the Userclass:

```
class User {
String username; String password;
  String email; double balance;
}
```

- Replace arrays with User[] objects
- Refactor registration/login logic using object-oriented principles

Concepts: Classes, objects, encapsulation basics

WEEK 5: BANK ACCOUNTS & ENCAPSULATION

Tasks:

- Add the BankAccountclass:

```
class BankAccount {
  private String accountNumber;
  private double balance;
  private User owner;
  // getters/setters
}
```

- Implement proper encapsulation with privatefields ▪

Link each user to one bank account

Concepts: Encapsulation, getters/setters, object relationships

WEEK 6: INHERITANCE & POLYMORPHISM

Tasks:

- Create base Personclass:

```
class Person {
  protected String name;
  protected String email;
}
```

- Build Userclass that extends Person
- Create abstract Accountclass with SavingAccountand CurrentAccount

subclasses ▪ Implement different withdrawal rules for each account

type

Concepts: Inheritance, super, abstract classes, polymorphism

WEEK 7: INTERFACES & LOAN SYSTEM FOUNDATION

Tasks:

- Create interfaces:

```
interface ILogin {
  boolean login(String username, String password);
}

interface ITransaction {
  void deposit(double amount); void
  withdraw(double amount);
}
```

- Introduce Loanclass:

```
class Loan {
  private double principal; private double
  interestRate; private int termMonths;
  private double monthlyPayment;
}
```

Concepts: Interfaces, loan calculations

WEEK 8: LOAN FUNCTIONALITY & CALCULATIONS

Tasks:

- Implement loan approval logic (based on account balance and income)
- Calculate monthly payments using standard loan formulas
- Track loan balances and payment history

Concepts: Financial calculations, business logic

WEEK 9: EXCEPTION HANDLING

Tasks:

Handle built-in exceptions:

- InputMismatchException for invalid menu choices
- NumberFormatException for invalid numeric inputs
- Create custom exceptions:

```
class InsufficientFundsException extends Exception { }
class LoanNotApprovedException extends Exception { }
class InvalidLoanPaymentException extends Exception { }
```

WEEK 10: FILE I/O & PERSISTENCE

Tasks:

Implement file operations:

- Save and load user data, accounts, and loans
- Use CSV format for simplicity
- Track program execution count
- Add data persistence across program runs

Concepts: File I/O, data serialization, persistence

WEEK 11: INTEGRATION & FINAL FEATURES

Tasks:

Complete loan system integration:

- Loan approval workflow
- Payment scheduling
- Interest calculations

Add reporting features:

- Account statements
- Loan payment history
- System-wide statistics

Conduct final testing and refinement

Concepts: System integration, reporting, testing

Final Project Features

Complete Banking System:

- User registration and login with validation
- Multiple account types (Saving/Current)
- Deposit and withdrawal operations
- **Loan system** with streamlined approval, payment processing, and comprehensive tracking

  Robust exception handling across all operations
- Reliable file-based data persistence
- Clean, professional CLI interface

Technical Skills Demonstrated:

- Object-oriented programming principles
- Inheritance and polymorphism implementation Interface design and

  implementation
- Comprehensive exception handling
- File I/O operations and data management
- Accurate financial calculations
- Data validation and security protocols