اسم المشروع: SmartATM Banking System

الهدف: مشروع ATM وهمي لتدريب على استخدام لغة Java بشكل احترافي، يغطي مفاهيم البرمجة الكائنية (OOP)، استخدام Enums ،Collections، التعامل مع ملفات JSON، تصميم معمارية متعددة الطبقات (Layered Architecture)، وإدارة سجلات المعاملات (Transactions).

هيكلية المشروع:

- src
- app
  - SmartATM.java (Main class)
- controllers
  - AuthController.java
  - AccountController.java
- models
  - Account.java
  - Customer.java
  - Transaction.java
  - TransactionType.java
- services
  - AuthenticationService.java
  - AccountService.java
  - TransactionService.java
- utils
  - TransactionFileManager.java
  - LocalDateTimeAdapter.java

تفصيل الملفات:

1. BankDatabase.java
2. يمثل قاعدة بيانات وهمية باستخدام HashMap.
3. المفتاح هو رقم البطاقة، والقيمة هي العميل (Customer).
4. يتم تحميل بيانات العملاء عند بدء البرنامج باستخدام static block.
5. Customer.java
6. يمثل العميل ويحتوي على: رقم البطاقة، الاسم الكامل، PIN، والحساب المرتبط.
7. استخدام مفهوم Encapsulation لحماية البيانات.
8. Account.java
9. يمثل الحساب البنكي ويحتوي على الرصيد وسجل المعاملات.
10. يحتوي على دوال لإجراء الإيداع، السحب، والتحقق من الرصيد.
11. كل عملية يتم تسجيلها في قائمة TransactionHistory.
12. TransactionType.java

13. Enum يحدد أنواع المعاملات: DEPOSIT، WITHDRAW، CHECK_BALANCE.
14. يقلل من الأخطاء الناتجة عن استخدام النصوص.
15. Transaction.java
16. يمثل كل عملية مالية وتحتوي على النوع، المبلغ، وتاريخ/وقت العملية.
17. toString() لتنسيق عرض السجل.
18. AuthenticationService.java
19. مسؤول عن التحقق من صحة رقم البطاقة وPIN.
20. يتحقق من وجود العميل في قاعدة البيانات.
21. AuthController.java
22. وسيط بين Main و AuthenticationService.
23. يفصل منطق التحكم عن المنطق التنفيذي.
24. AccountService.java و AccountController.java
25. AccountService: تنفيذ العمليات المالية على الحساب.
26. AccountController: التعامل مع واجهة المستخدم ونقل الطلبات للخدمة.
27. تطبيق Layered Architecture لفصل المسؤوليات.
28. TransactionService.java
29. مسؤول عن استرجاع وعرض سجل العمليات لكل حساب.
30. يضمن فصل عرض البيانات عن الحساب نفسه.
31. TransactionFileManager.java
32. حفظ سجل العمليات في ملف JSON بشكل منسق.
33. استخدام مكتبة Gson مع PrettyPrinting.
34. التعامل مع الأخطاء باستخدام try/catch.
35. LocalDateTimeAdapter.java
36. حل مشكلة Gson مع LocalDateTime في Java 17+.
37. يحول LocalDateTime إلى String والعكس عند قراءة/كتابة JSON.
38. SmartATM.java (Main)
39. نقطة البداية للبرنامج.

40. قراءة مدخلات المستخدم مثل رقم البطاقة وPIN.
41. التعامل مع Controllers و Services لتنفيذ العمليات.
42. تنفيذ القوائم المختلفة: Check Balance، Deposit، Withdraw، Transaction Log، Exit.

المبادئ البرمجية المستخدمة: - OOP: Encapsulation، Abstraction، Single Responsibility. - Collections: HashMap لتخزين العملاء، ArrayList لسجل المعاملات. - Enum: لتحديد أنواع العمليات بدقة. - JSON Handling: حفظ وقراءة البيانات باستخدام Gson مع Adapter. - Layered Architecture: فصل المسؤوليات بين Controller و Service و Model و Exception Handling - Utils.: حماية البرنامج من أخطاء الملفات.

تدفق البرنامج: 1. إدخال رقم البطاقة وPIN من قبل المستخدم. 2. التحقق من صحة البيانات باستخدام AuthController و AuthenticationService. 3. عرض القائمة الرئيسية للعمليات. 4. تنفيذ العمليات المالية على الحساب من خلال AccountController و AccountService. 5. تسجيل كل عملية في TransactionHistory الخاص بالحساب. 6. حفظ السجل في ملف transactions.json باستخدام TransactionFileManager. 7. عند اختيار الخروج، يتم إنهاء البرنامج بشكل آمن.

ملخص: - المشروع يغطي مفاهيم Java OOP، Collections، Enums، JSON Handling، File Handling، Layered Architecture، و Adapter Pattern. - تصميم المشروع يركز على الفصل بين البيانات، المنطق، والتحكم لضمان قابلية الصيانة والتطوير في المستقبل.