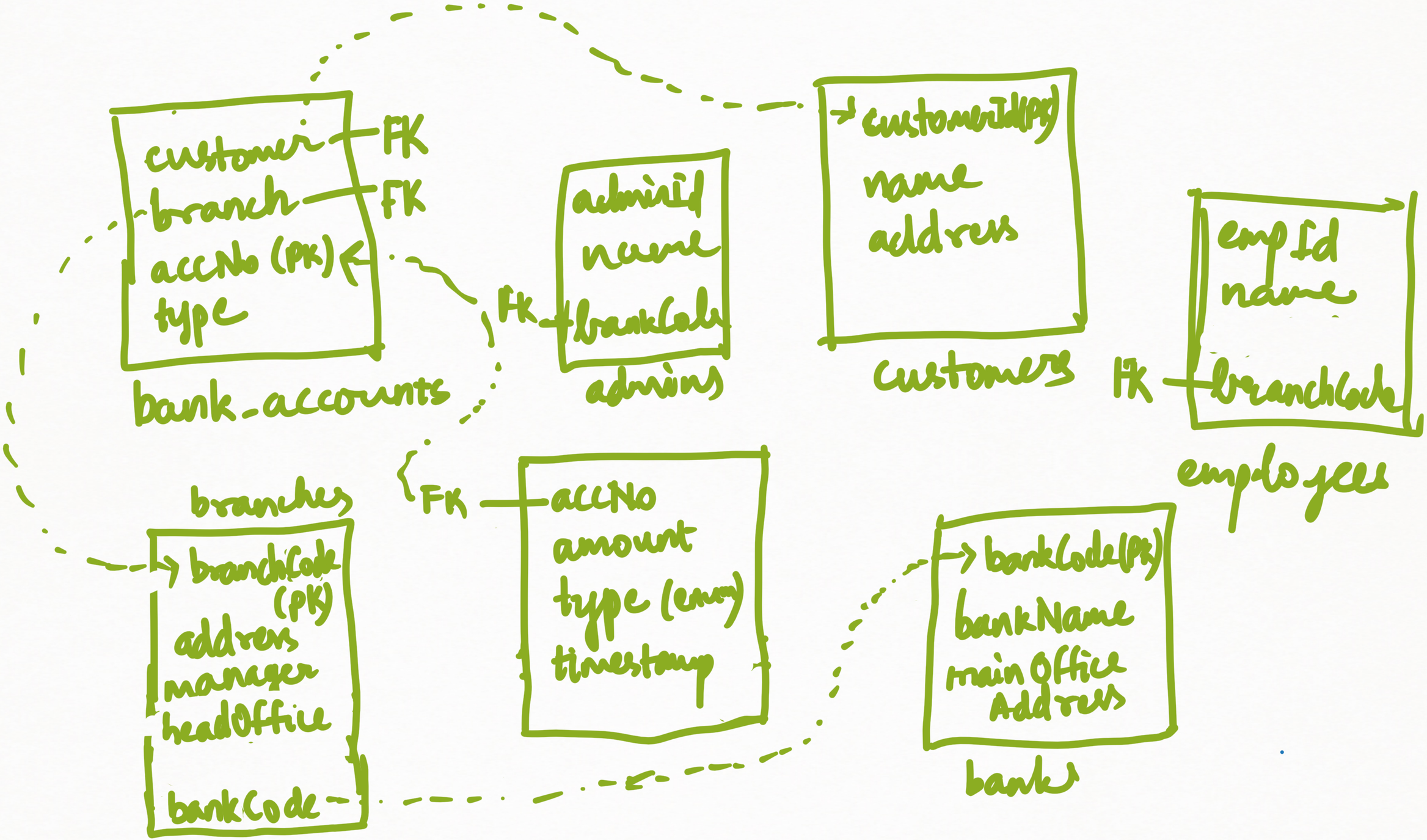

customer
branch
accNo (PK)
type
FK
FK
bank_accounts
FK
adminId
name
bankCode
admins
customerId(PK)
name
address
customers
empId
name
FK
branchCode
employees
branches
branchCode (PK)
address
manager
headOffice
bankCode
FK
accNo
amount
type (enum)
timestamp
bankCode(PK)
bankName
mainOffice Address
banks

Classes and interfaces :-

1) Bank
2) Branch
3) BankAccount
4) SavingsAccount
5) CurrentAccount
6) Customer
7) Address
8) Transaction

Relationships between Entities

Branch — HAS-A — Bank

SavingsAccount — IS-A — BankAccount

CurrentAccount — IS-A — BankAccount

BankAccount — HAS-A — Customer

Transaction — HAS-A — BankAccount

[Optional] Relationships of Bank, Branch & Customer with the Address class — HAS-A

Class Descriptions

Bank: (optional ListKBranch)

- ✓ Integer bankCode;
- String bankName;
- Address mainOfficeAddr;
- //constructors
- //getters & setters
- //override toString
- //override compareTo

Branch:

- 4) Integer branchCode;
- ✓ 4) Integer bankCode;
  - OR
  - Bank bank;
- 3) String manager;
- 4) Address address;
- //constructors, getters, setters, toString, compareTo, equals, hashCode.

Address!**

```
String street;
String city;
String state;
String country;
Integer pincode;
// other statutories
```

* can be a string too

Customer:

```
String name;
*Address address;
Integer customerId;
//other statutories
```

Transaction:

Integer accountNo;
OR
BankAccount account;
enum("credit", "debit") type;
Date date;
Double amount;
//other statutories

BankAccount (abstract):

1) Integer accountNo;
2) Integer customerId;
OR
Customer customer;
3) Integer branchCode;
OR
Branch branch;
4) enum("savings", "current") type;
//other statutories
// methods from diagram

SavingsAccount and CurrentAccount

Refer to Core Java discussions #14.

Additional fields to consider:-

i) In BankAccount, a List<Transaction>
ii) In Customer, a List<BankAccount>
iii) In Branch, a List<BankAccount>
iv) In Bank, a List<Branch>

Controllers

1) HomeController :-

index() - displays the landing page with a navigation bar.

Shows a list of all banks in a sidebar (clickable)

The navbar will have a "Customer Portal" button. which will take you to CustomersController#index

2) Transaction Controller

deposit() – GET(accountNo(PV), amt(QP))

withdraw() – POST (accountNo, amt)

admin – showTransaction(optional<startDate>,
optional<endDate>)

CustomerController

index() - shows a form to enter customerId

welcome() - GET (customerId(PV))

- Welcome message with <name>
<options> including [Bank Accounts]

viewProfile() - GET (customerId(PV))

and / updateProfile() - PATCH (name and/or address from a form)

\ editProfileForm() - GET (show a form populated with customer details)
L c/o/e

createU() - admin/employee

deleteU() - DELETE - (customerId(PV)) - c/a

showTransactions() - (optional<startDate>,
optional<endDate>

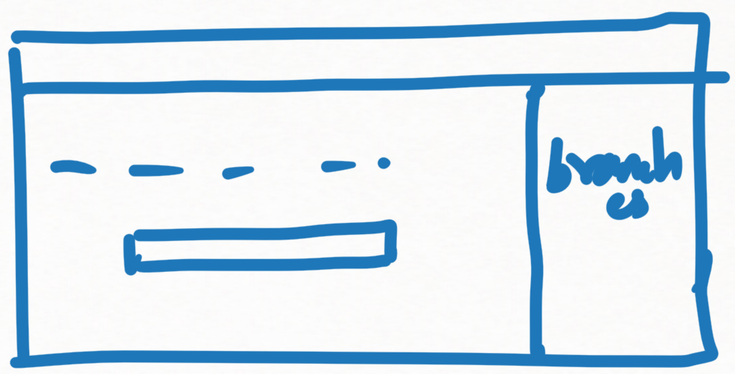
BankController

createU

updateU < GET / PATCH

deleteU

viewDetailsU

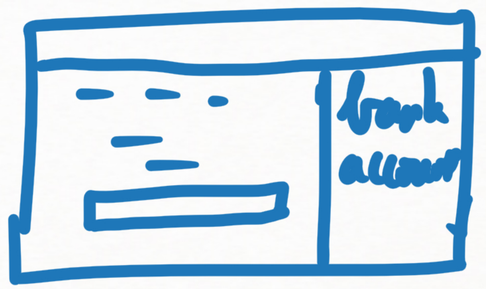


BranchController

→ can be done by both admins & employees of that branch

both - view()
admin - delete()
create()
update()

(branch(code))

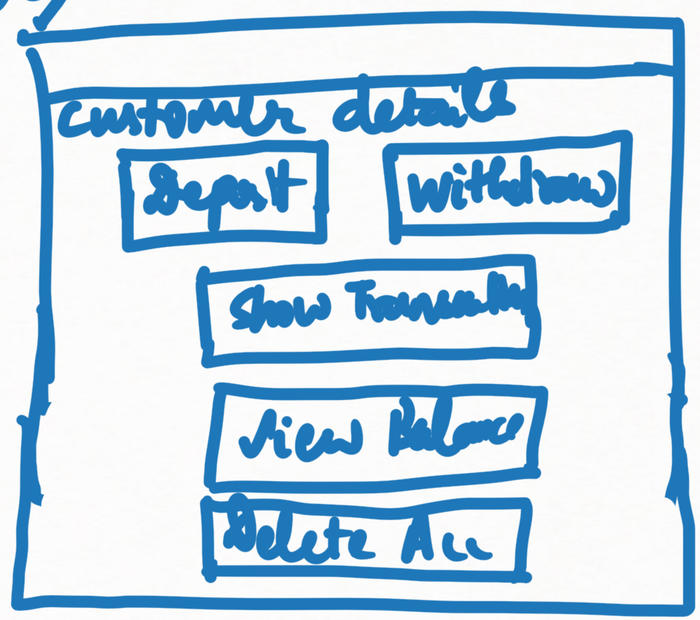


both - showTransactions(optional<startDate>, optional<endDate>)

both - showCustomers(optional<accountType>)

BankAccountController

- C (employee)
- R – viewDetails( ) – details
- U (employee)
- D (employee/customer)



EmployeeController

AdminController

C, U, D X

View Profile (Admin)

Frontend Pages