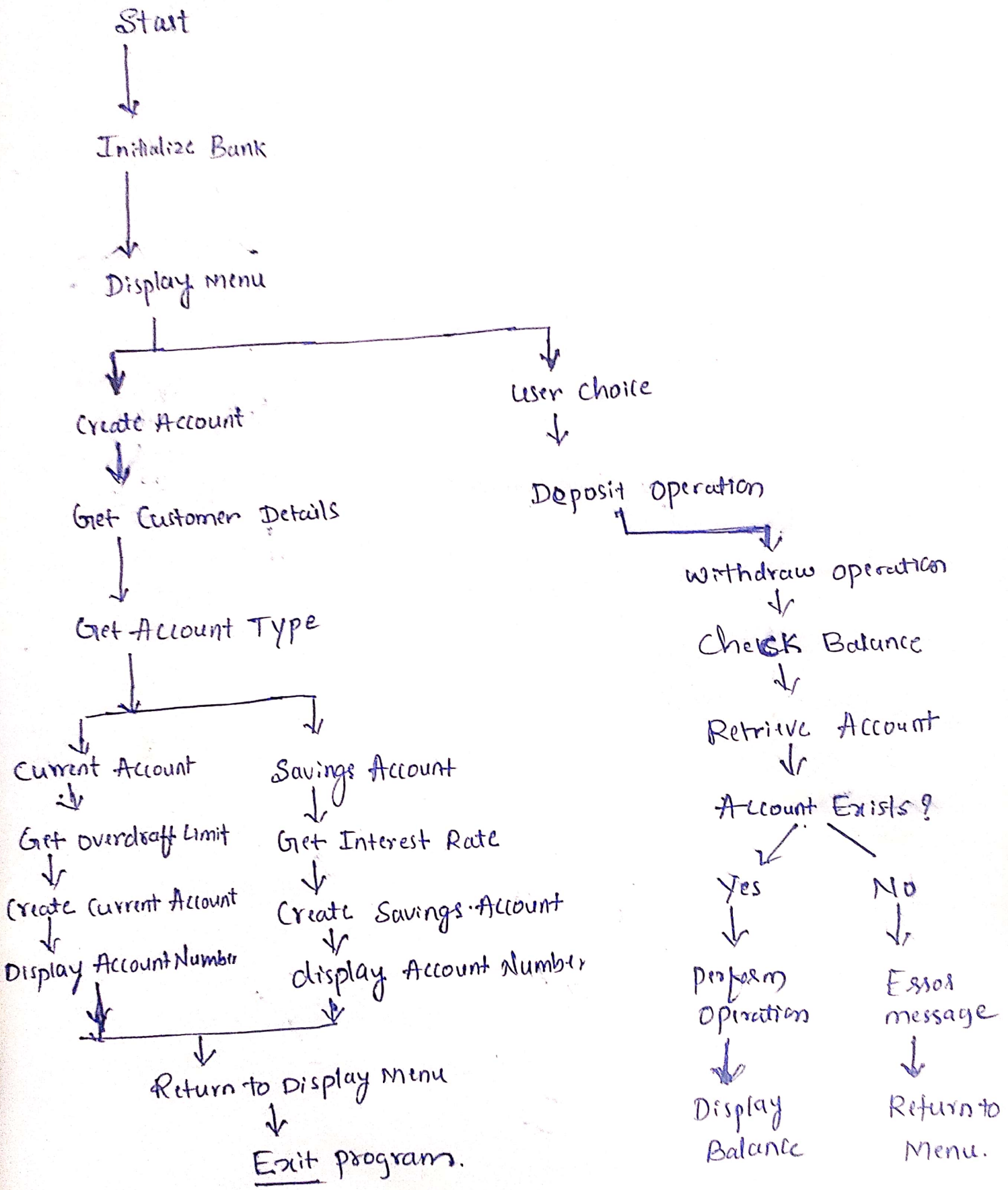
Banking System Operation



Structure of code

- main method.
- Create new Bank Object.
- display menu → while (true) {
  - Sop("welcome to Banking System");
  - Sop("1. create Account
  - Sop("2. Deposit");
  - Sop("3. withdraw");
  - Sop("4. check Balance");
  - Sop("5. Exit");

```
int choice = scanner.nextInt();
Switch (choice) {
case 1: → Create Account
```

- customer name
- customer Address
- customer Contact

```
Customer customer = new Customer (name, address, contact);
Sop ("Enter account type (1 for saving, 2 for current):");
int accountType = scanner.nextInt();
if (accountType == 1)
    Sop ("Enter interest rate:");
Account account = new SavingsAccount (customer, interestRate);
bank. addAccount (account);
Sop ("Savings Account created succesfully") Acount Number: " + account.
                                              getAccountNumber());
else if (accountType == 2)
    Sop ("Enter overdraft limit:");
    Account account = new CurrentAccount (customer, overdraftLimit);
    bank. addAccount (account);
    Sop ("current Aaccc create succ. Account number :" + account. get
                                              AccountNumber());
else
    Sop ("Invalid account Type.").
    break;
```

```
Case 2: → Deposit to an account
    sop("enter account number");
    Account acc = bank.getAccount(accountNumber);
    if (acc != null)
        sop("Enter deposit amount:");
        acc.deposit(amount);
    else
        sop("Account not found.");
    break;
Case 3: → withdraw from an account
    sop("Enter account number:");
    acc = bank.getAccount(accountNumber);
    if (acc != null)
        sop("Enter withdrawal amount:");
        acc.withdraw(amount);
    else
        sop("Account not found");
        break;
case 4: → Check account Balance
    sop("Enter the account number:");
    acc = bank.getAccount(accountNumber);
    if (acc != null)
        sop("Account Balance:" + acc.getBalance());
    else
        sop("Account not found.");
    break;
Case 5: → Exit the application
    sop("Thank you for using the Banking System.");
    scanner.close()
    System.exit(0);
    break;
default:
    sop("Invalid choice. please try again.");
```

```
Class Bank {
    private ArrayList<Account> accounts;
    Public Bank(){
        accounts = new ArrayList<>();
    }
    Public void addAccount(Account account){
        accounts.add(account);
    }
    public Account getAccount(int accountNumber){
        for(Account account : accounts){
            if(account.getAccountNumber() == accountNumber){
                return account;
            }
        }
        return null;
    }
}

Class Customer {
    private String name;
    private String address;
    private String contact;

    public Customer(String name, String address, String contact)
    {
        this.name = name;
        this.address = address;
        this.contact = contact;
    }
    Getters and setters
}

abstract class Account{
    private static int accountCounter = 1000;
    private int accountNumber;
    private Customer customer;
    private double balance;
    public Account(Customer customer){
        this.accountNumber = ++accountCounter;
        this.customer = customer;
    } this.balance = 0.0;
```

```
    public int getAccountNumber() {
    }  return accountNumber;

    public double getBalance() {
    }  return balance;

    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
            Sop("Deposit Successfully" + balance);
        else
            ↳ sop("Invalid deposit");
        }}

    public abstract void withdraw(double amount);
}

class SavingsAccount Extends Account {
    private double InterestRate;

    public SavingsAccount(Customer customer, double interestRate) {
        Super(customer);
        this.interestRate = interestRate;
    }

    public void withdraw(double amount) {
        if(ammount >0 && ammount <=getBalance()) {
            double newBalance = getBalance() - amount;
            Sop("withdraw Succe. New bala" + newBalance);
        else{
            ↳ sop("Invalid withdraw ammount or insufficient funds.");
        }}
    }}

class currentAccount Extends Account {
    private double overdraftLimit;
    Same as above Continue.
```

```
public void addInterest(){
double interest = getBalance * inR/
deposit(interest);
sop("Interest add. newBal =getB
```