SmartBankApp Code

```
import java.util.*;
import java.util.regex.*;

// ============================
// Main Class: SmartBankApp
// ============================
public class SmartBankApp {
    public static void main(String[] args) {
        // Create an instance of the Bank class
        Bank bank = new Bank();

        System.out.println("====== WELCOME TO SMART BANK ======");
        // Show the main menu
        bank.menu();

        System.out.println("Thank you for banking with us! Come again.");
    }
}

// ============================
// Bank Class for Core Operations
// ============================
class Bank {
    // Data members
    private final Map<Long, Account> accounts = new HashMap<>();
    private final Scanner scanner = new Scanner(System.in);

    // Regular expressions for validation
    private static final Pattern MOBILE_PATTERN = Pattern.compile("^[0-9]{10}$");
    private static final Pattern NAME_PATTERN = Pattern.compile("^[a-zA-Z ]+$");

    // Main menu method
    public void menu() {
        while (true) {
```

```
        System.out.println("\n1: Create Account");
        System.out.println("2: Login to Account");
        System.out.println("3: Exit");
        System.out.print("Select an option: ");

        try {
            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume the newline

            switch (choice) {
                case 1:
                    createAccount(); // Call method to create an account
                    break;

                case 2:
                    loginToAccount(); // Call method to log in to an account
                    break;

                case 3:
                    return; // Exit the menu

                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        } catch (InputMismatchException e) {
            System.out.println("Invalid input. Please enter a number.");
            scanner.nextLine(); // Clear the invalid input
        }
    }
}

// ============================
// Create Account Section
// ============================
private void createAccount() {
```

```
        // Step 1: Validate name
        String name;
        while (true) {
            System.out.print("Enter Name: ");
            name = scanner.nextLine().trim();
            if (NAME_PATTERN.matcher(name).matches()) {
                break;
            } else {
                System.out.println("Invalid name. Please enter only valid name.");
            }
        }

        // Step 2: Validate mobile number
        String mobile;
        while (true) {
            System.out.print("Enter Mobile Number: ");
            mobile = scanner.nextLine().trim();
            if (MOBILE_PATTERN.matcher(mobile).matches()) {
                break;
            } else {
                    System.out.println("Invalid mobile number. Please enter exactly 10
numeric digits.");
            }
        }

        // Step 3: Validate initial amount
        double initialAmount;
        while (true) {
            System.out.print("Enter Initial Amount (minimum 500): ");
            try {
                initialAmount = scanner.nextDouble();
                scanner.nextLine(); // Consume the newline
                if (initialAmount >= 500) break;
                System.out.println("Initial amount must be at least 500.");
            } catch (InputMismatchException e) {
```

```
            System.out.println("Invalid input. Please enter a valid number.");
            scanner.nextLine(); // Clear the invalid input
        }
    }

    // Step 4: Generate account number
    long accountNumber = generateRandomAccountNumber();

    // Step 5: Select account type
    int type;
    while (true) {
        System.out.println("Select Account Type:");
        System.out.println("1. Savings");
        System.out.println("2. Fixed Deposit");
        System.out.print("Enter your choice (1 or 2): ");
        try {
            type = scanner.nextInt();
            scanner.nextLine(); // Consume the newline
            if (type == 1 || type == 2) break;
            System.out.println("Invalid choice. Please select 1 or 2.");
        } catch (InputMismatchException e) {
            System.out.println("Invalid input. Please enter 1 or 2.");
            scanner.nextLine(); // Clear the invalid input
        }
    }

    // Step 6: Create the account and store it
    Account account = (type == 1) ?
            new SavingsAccount(name, mobile, initialAmount, accountNumber) :
            new FDAccount(name, mobile, initialAmount, accountNumber);

    accounts.put(accountNumber, account);

      System.out.println("Account created successfully. Your account number is: " +
accountNumber);
```

```
}

// ============================
// Login to Account Section
// ============================
private void loginToAccount() {
    if (accounts.isEmpty()) {
        System.out.println("No accounts found. Please create an account first.");
        return;
    }

    System.out.print("Enter Account Number: ");
    try {
        long accountNumber = scanner.nextLong();
        scanner.nextLine(); // Consume the newline

        // Retrieve the account
        Account account = accounts.get(accountNumber);
        if (account == null) {
            throw new InvalidAccountException("Account number not found.");
        }

        System.out.println("Welcome, " + account.getName() + "!");
        account.menu(scanner); // Show account-specific menu
    } catch (InputMismatchException e) {
        System.out.println("Invalid input. Please enter a valid account number.");
        scanner.nextLine(); // Clear the invalid input
    } catch (InvalidAccountException e) {
        System.out.println("Error: " + e.getMessage());
    }
}

// ============================
// Utility: Generate Random Account Number
// ============================
```

```
    private long generateRandomAccountNumber() {
        Random random = new Random();
        long accountNumber;
        do {
            accountNumber = 1000000000L + (long) (random.nextDouble() * 9000000000L);
        } while (accounts.containsKey(accountNumber));
        return accountNumber;
    }
}

// ============================
// Abstract Class: Account
// ============================
abstract class Account {
    private final String name, mobileNumber;
    private double balance;
    private final long accountNumber;

    // Constructor
    Account(String name, String mobileNumber, double balance, long accountNumber) {
        this.name = name;
        this.mobileNumber = mobileNumber;
        this.balance = balance;
        this.accountNumber = accountNumber;
    }

    public String getName() { return name; }

    public double getBalance() { return balance; }

    public long getAccountNumber() { return accountNumber; }

    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
```

```
            System.out.println("Deposited " + amount + ". Current balance: " + balance);
        } else {
            System.out.println("Invalid deposit amount.");
        }
    }

    public void withdraw(double amount) throws InsufficientBalanceException {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
            System.out.println("Withdrew " + amount + ". Current balance: " + balance);
        } else if (amount > balance) {
            throw new InsufficientBalanceException("Insufficient balance.");
        } else {
            System.out.println("Invalid withdrawal amount.");
        }
    }

    public void statement() {
        System.out.println("Account Statement:");
        System.out.println("Name: " + name);
        System.out.println("Mobile Number: " + mobileNumber);
        System.out.println("Account Number: " + accountNumber);
        System.out.println("Balance: " + balance);
    }

    public abstract void menu(Scanner scanner);
}

// ============================
// SavingsAccount Class
// ============================
class SavingsAccount extends Account {
    SavingsAccount(String name, String mobile, double balance, long accountNumber) {
        super(name, mobile, balance, accountNumber);
    }
```

```
    @Override
    public void menu(Scanner scanner) {
        while (true) {
                 System.out.println("\n1: Check Balance\n2: Deposit Amount\n3: Withdraw
Amount\n4: View Statement\n5: Exit");
            System.out.print("Select an option: ");

            try {
                int choice = scanner.nextInt();
                scanner.nextLine();

                switch (choice) {
                    case 1:
                        System.out.println("Current balance: " + getBalance());
                        break;

                    case 2:
                        System.out.print("Enter deposit amount: ");
                        deposit(scanner.nextDouble());
                        scanner.nextLine();
                        break;

                    case 3:
                        System.out.print("Enter withdrawal amount: ");
                        double amt = scanner.nextDouble();
                        scanner.nextLine();
                        try {
                            withdraw(amt);
                        } catch (InsufficientBalanceException e) {
                            System.out.println("Error: " + e.getMessage());
                        }
                        break;

                    case 4:
```

```
                    statement();
                    break;

                case 5:
                    System.out.println("Exiting Savings Account menu...");
                    return;

                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        } catch (InputMismatchException e) {
            System.out.println("Invalid input. Please enter a number.");
            scanner.nextLine();
        }
    }
}

// ============================
// FDAccount Class
// ============================
class FDAccount extends Account {
    FDAccount(String name, String mobile, double balance, long accountNumber) {
        super(name, mobile, balance, accountNumber);
    }

    @Override
    public void menu(Scanner scanner) {
        while (true) {
            System.out.println("\n1: Check Balance\n2: View Statement\n3: Exit");
            System.out.print("Select an option: ");

            try {
                int choice = scanner.nextInt();
                scanner.nextLine();
```

```
                switch (choice) {
                    case 1:
                        System.out.println("Current balance: " + getBalance());
                        break;

                    case 2:
                        statement();
                        break;

                    case 3:
                        System.out.println("Exiting FD Account menu...");
                        return;

                    default:
                        System.out.println("Invalid choice. Please try again.");
                }
            } catch (InputMismatchException e) {
                System.out.println("Invalid input. Please enter a number.");
                scanner.nextLine();
            }
        }
    }
}

// ============================
// Custom Exceptions
// ============================
class InsufficientBalanceException extends Exception {
    public InsufficientBalanceException(String message) {
        super(message);
    }
}

class InvalidAccountException extends Exception {
```

```
    public InvalidAccountException(String message) {
        super(message);
    }
}
```