1. Account Table (Base class)

Column Name	Data Type	Constraints
account_id	INT	PRIMARY KEY, AUTO_INCREMENT
customer_id	INT	FOREIGN KEY (`Customer.customer_id`)
account_type	VARCHAR(50)	NOT NULL
balance	DECIMAL(15,2)	NOT NULL
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP

2. SavingsAccount Table (Inherits from Account)

Column Name	Data Type	Constraints
account_id	INT	PRIMARY KEY, FOREIGN KEY (`Account.account_id`)
interest_rate	DECIMAL(5,2)	NOT NULL

3. CurrentAccount Table (Inherits from Account)

Column Name	Data Type	Constraints
account_id	INT	PRIMARY KEY, FOREIGN KEY (`Account.account_id`)
overdraft_limit	DECIMAL(15,2)	NOT NULL

4. Transaction Table (Base class)

Column Name	Data Type	Constraints
transaction_id	INT	PRIMARY KEY, AUTO_INCREMENT
account_id	INT	FOREIGN KEY (`Account.account_id`)
transaction_type	VARCHAR(50)	NOT NULL
amount	DECIMAL(15,2)	NOT NULL
transaction_date	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP

5. DepositTransaction Table (Inherits from Transaction)

Column Name	Data Type	Constraints
transaction_id	INT	PRIMARY KEY, FOREIGN KEY (`Transaction.transaction_id`)
deposit_method	VARCHAR(50)	NOT NULL

6. WithdrawalTransaction Table (Inherits from Transaction)

Column Name	Data Type	Constraints
transaction_id	INT	PRIMARY KEY, FOREIGN KEY (`Transaction.transaction_id`)
withdrawal_method	VARCHAR(50)	NOT NULL

7. Bank Table

Column Name	Data Type	Constraints
bank_id	INT	PRIMARY KEY, AUTO_INCREMENT
bank_name	VARCHAR(100)	NOT NULL
bank_branch	VARCHAR(100)	NOT NULL

Java - Requirements:

1. Basic Syntax and Control Structures:

- Use appropriate control structures (if-else, loops, switch-case) for decision-making and iteration within the banking operations.

2. Classes and Objects:

- Create classes such as ***Bank***, ***Account***, ***SavingsAccount***, ***CurrentAccount***, ***Transaction***, and more to model the banking system.
- Use objects to represent customers and bank accounts.

3. Inheritance:

- Create an abstract ***Account*** class that is inherited by the ***SavingsAccount*** and ***CurrentAccount*** classes.

4. Polymorphism:

- Use method overriding to define different rules for withdrawing money for ***SavingsAccount*** and ***CurrentAccount***.

5. Abstraction and Encapsulation:

- Encapsulate the account details using private variables and provide public getter and setter methods to manipulate account information.