BANKACCOUNT.JAVA

```
package org.inherit;

import java.io.BufferedReader;
import java.io.InputStreamReader;

import javax.persistence.DiscriminatorColumn;
import javax.persistence.DiscriminatorType;
import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.Inheritance;
import javax.persistence.InheritanceType;
@Entity
@Inheritance(strategy = InheritanceType.JOINED)
@DiscriminatorColumn(name = "ACC_TYPE", discriminatorType = DiscriminatorType.STRING)
@DiscriminatorValue("acc")
public class BankAccount {
	@Id
	@GeneratedValue
	private int id;
	private long accountNumber;
	private String accountHolder;
	private String address;
	private long phoneNumber;
	protected double balance;
	public double amount;

	public BankAccount() {
		super();
	}

	public BankAccount(long accountNumber, String accountHolder, String address, long
phoneNumber, double balance) {
		super();
		this.accountNumber = accountNumber;
		this.accountHolder = accountHolder;
		this.address = address;
		this.phoneNumber = phoneNumber;
		this.balance = balance;
	}

	public int getId() {
		return id;
	}

	public void setId(int id) {
```

```
        this.id = id;
}

public long getAccountNumber() {
        return accountNumber;
}

public void setAccountNumber(long accountNumber) {
        this.accountNumber = accountNumber;
}

public String getAccountHolder() {
        return accountHolder;
}

public void setAccountHolder(String accountHolder) {
        this.accountHolder = accountHolder;
}

public String getAddress() {
        return address;
}

public void setAddress(String address) {
        this.address = address;
}

public long getPhoneNumber() {
        return phoneNumber;
}

public void setPhoneNumber(long phoneNumber) {
        this.phoneNumber = phoneNumber;
}

public double getBalance() {
        return balance;
}

public void setBalance(double balance) {
        this.balance = balance;
}

public boolean withdraw(double amount) {
        this.balance = this.balance - this.amount;
        return true;
}
```

```
	public boolean deposit(double amount) {
		this.balance = this.balance + this.amount;
		return true;
	}
}
```

CURRENTACCOUNT.JAVA

```
package org.inherit;

import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;

@Entity
@DiscriminatorValue("current")
public class CurrentAccount extends BankAccount {
	private String memberName;
	private static double minimumAmountCanTransfer = 500000.00;
	private static int minimumNumberOfTransactions = 7;
	private double amountTransferred;
	private int numberOfTransactionsHeld=0;
	public CurrentAccount() {
		super();
		this.numberOfTransactionsHeld++;
	}
	public CurrentAccount(long accountNumber, String accountHolder, String address, long
phoneNumber, double balance,
			String memberName, double amountTransferred) {
		super(accountNumber, accountHolder, address, phoneNumber, balance);
		this.memberName = memberName;
		this.amountTransferred = amountTransferred;
		//this.numberOfTransactionsHeld = numberOfTransactionsHeld;
	}
	public String getMemberName() {
		return memberName;
	}
	public void setMemberName(String memberName) {
		this.memberName = memberName;
	}
	public static double getMinimumAmountCanTransfer() {
		return minimumAmountCanTransfer;
	}
	public static void setMinimumAmountCanTransfer(double minimumAmountCanTransfer) {
		CurrentAccount.minimumAmountCanTransfer = minimumAmountCanTransfer;
	}
	public static int getMinimumNumberOfTransactions() {
		return minimumNumberOfTransactions;
	}
	public static void setMinimumNumberOfTransactions(int minimumNumberOfTransactions) {
		CurrentAccount.minimumNumberOfTransactions = minimumNumberOfTransactions;
```

```
	}
	public double getAmountTransferred() {
		return amountTransferred;
	}
	public void setAmountTransferred(double amountTransferred) {
		this.amountTransferred = amountTransferred;
	}
	public int getNumberOfTransactionsHeld() {
		return numberOfTransactionsHeld;
	}
	public void setNumberOfTransactionsHeld(int numberOfTransactionsHeld) {
		this.numberOfTransactionsHeld = numberOfTransactionsHeld;
	}
	@Override
	public boolean withdraw(double amount) {

		return super.withdraw(amount);
	}
	@Override
	public boolean deposit(double amount) {
		// TODO Auto-generated method stub
		return super.deposit(amount);
	}


}
```

SAVINGSACCOUNT.JAVA

```
package org.inherit;

import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;

@Entity
@DiscriminatorValue("savings")
public class SavingsAccount extends BankAccount {
	private String memberName;
	private static double maximumAmountCanTransfer = 100000.00;
	private static int maximumNumberOfTransactions = 5;
	private double amountTransferred;
	private int numberOfTransactionsHeld=0;

	public SavingsAccount() {
		super();
		this.numberOfTransactionsHeld++;
	}

	public SavingsAccount(long accountNumber, String accountHolder, String address, long
phoneNumber, double balance,
```

```
                String memberName, double amountTransferred) {
        super(accountNumber, accountHolder, address, phoneNumber, balance);
        this.memberName = memberName;
        this.amountTransferred = amountTransferred;
        //this.numberOfTransactionsHeld = numberOfTransactionsHeld;
}

public String getMemberName() {
        return memberName;
}

public void setMemberName(String memberName) {
        this.memberName = memberName;
}

public static double getMaximumAmountCanTransfer() {
        return maximumAmountCanTransfer;
}

public static void setMaximumAmountCanTransfer(double maximumAmountCanTransfer) {
        SavingsAccount.maximumAmountCanTransfer = maximumAmountCanTransfer;
}

public static int getMaximumNumberOfTransactions() {
        return maximumNumberOfTransactions;
}

public static void setMaximumNumberOfTransactions(int maximumNumberOfTransactions) {
        SavingsAccount.maximumNumberOfTransactions = maximumNumberOfTransactions;
}

public double getAmountTransferred() {
        return amountTransferred;
}

public void setAmountTransferred(double amount) {
        this.amountTransferred = amountTransferred;
}

public int getNumberOfTransactionsHeld() {
        return numberOfTransactionsHeld;
}

public void setNumberOfTransactionsHeld(int numberOfTransactionsHeld) {
        this.numberOfTransactionsHeld = numberOfTransactionsHeld;
}

@Override
```

```
	public boolean withdraw(double amount) {
		// TODO Auto-generated method stub
		return super.withdraw(amount);
	}

	@Override
	public boolean deposit(double amount) {
		// TODO Auto-generated method stub
		return super.deposit(amount);
	}

}
```

SOLUTIONMAIN.JAVA

```
package org.main;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;
import org.inherit.CurrentAccount;
import org.inherit.SavingsAccount;
import org.inherit.BankAccount;

public class SolutionMain {

	public static void main(String[] args) throws IOException {
		SessionFactory sf = new Configuration().configure().buildSessionFactory();
		Session session = sf.openSession();
		BufferedReader bf = new BufferedReader(new InputStreamReader(System.in));
		session.beginTransaction();

		String memberName;
		double amountTransferred;

		System.out.println("Enter the account number:");
		long accountNumber = Long.valueOf(bf.readLine());
		System.out.println("Enter the account holder:");
		String accountHolder = bf.readLine();
		System.out.println("Enter the address:");
		String address = bf.readLine();
		System.out.println("Enter the phone number:");
		long phoneNumber = Long.valueOf(bf.readLine());
		System.out.println("Enter the balance:");
		double balance = Double.valueOf(bf.readLine());
```

```
                BankAccount ba = new BankAccount(accountNumber, accountHolder, address,
phoneNumber, balance);
                ba.setAccountNumber(accountNumber);
                ba.setAccountHolder(accountHolder);
                ba.setAddress(address);
                ba.setPhoneNumber(phoneNumber);
                ba.setBalance(balance);

                System.out.println("enter your choice:\n1.withdraw\n 2.deposit");
                int ch = Integer.valueOf(bf.readLine());
                switch (ch) {

                case 1:
                        System.out.println("Enter the amount to withdraw");
                        amountTransferred = Double.valueOf(bf.readLine());

                        System.out.println("Enter the type of account:");
                        System.out.println("1.Savings\n 2. Current");
                        int type = Integer.valueOf(bf.readLine());
                        switch (type) {
                        case 1:
                                System.out.println("SAVINGS ACCOUNT");
                                System.out.println("Enter the member name:");
                                memberName = bf.readLine();

                                SavingsAccount sa = new SavingsAccount(accountNumber,
accountHolder, address, phoneNumber, balance,
                                                memberName, amountTransferred);
                                sa.withdraw(amountTransferred);
                                sa.setAccountNumber(accountNumber);
                                sa.setAccountHolder(accountHolder);
                                sa.setAddress(address);
                                sa.setPhoneNumber(phoneNumber);
                                sa.setMemberName(memberName);
                                sa.setAmountTransferred(amountTransferred);
                                sa.setBalance(balance);

                                System.out.println("Transaction saved");

                                session.save(sa);
                                break;

                        case 2:

                                System.out.println("CURENT ACCOUNT");
                                System.out.println("Enter the member name:");
                                memberName = bf.readLine();
```

```
                        CurrentAccount ca = new CurrentAccount(accountNumber,
accountHolder, address, phoneNumber, balance,
                                        memberName, amountTransferred);
                        ca.withdraw(amountTransferred);
                        ca.setAccountNumber(accountNumber);
                        ca.setAccountHolder(accountHolder);
                        ca.setAddress(address);
                        ca.setPhoneNumber(phoneNumber);
                        ca.setMemberName(memberName);
                        ca.setAmountTransferred(amountTransferred);
                        ca.setBalance(balance);

                        System.out.println("Transaction saved");
                        session.save(ca);
                        break;
                }
                break;
        case 2:
                System.out.println("Enter the amount to deposit");
                amountTransferred = Double.valueOf(bf.readLine());

                System.out.println("Enter the type of account:");
                System.out.println("1.Savings\n 2. Current");
                int type1 = Integer.valueOf(bf.readLine());
                switch (type1) {
                case 1:
                        System.out.println("SAVINGS ACCOUNT");
                        System.out.println("Enter the member name:");
                        memberName = bf.readLine();
                        SavingsAccount sa = new SavingsAccount(accountNumber,
accountHolder, address, phoneNumber, balance,
                                        memberName, amountTransferred);
                        sa.deposit(amountTransferred);
                        System.out.println("Transaction saved");
                        sa.setAccountNumber(accountNumber);
                        sa.setAccountHolder(accountHolder);
                        sa.setAddress(address);
                        sa.setPhoneNumber(phoneNumber);
                        sa.setMemberName(memberName);
                        sa.setAmountTransferred(amountTransferred);
                        sa.setBalance(balance);
                        session.save(sa);
                        break;

                case 2:

                        System.out.println("CURENT ACCOUNT");
                        System.out.println("Enter the member name:");
```

```
                                        memberName = bf.readLine();
                                        CurrentAccount ca = new CurrentAccount(accountNumber,
accountHolder, address, phoneNumber, balance,
                                                                memberName, amountTransferred);
                                        ca.deposit(amountTransferred);
                                        ca.setAccountNumber(accountNumber);
                                        ca.setAccountHolder(accountHolder);
                                        ca.setAddress(address);
                                        ca.setPhoneNumber(phoneNumber);
                                        ca.setMemberName(memberName);
                                        ca.setAmountTransferred(amountTransferred);
                                        ca.setBalance(balance);

                                        System.out.println("Transaction saved");
                                        session.save(ca);
                                        break;
                                }
                                break;

                        }
                        session.getTransaction().commit();
                        session.close();
                        sf.close();

                }
}
```