package za.ac.nwu.as.logic.flow;  
  
import za.ac.nwu.as.domain.dto.AccountTransactionDto;  
import za.ac.nwu.as.domain.persistence.AccountTransaction;  
  
import java.util.List;  
  
public interface FetchAccountTransactionFlow {  
  
 List<AccountTransactionDto> getAllAccountTransactions();  
 AccountTransactionDto getAccountTransactionByMemberId(Long memberId);  
 AccountTransaction getAccountTransactionForMemberId(Long memberId);  
}

package za.ac.nwu.as.logic.flow;  
  
import za.ac.nwu.as.domain.dto.AccountTransactionDto;  
  
public interface CreateAccountTransactionFlow {  
 AccountTransactionDto createAdd(AccountTransactionDto accountTransaction);  
 AccountTransactionDto createWithdraw(AccountTransactionDto accountTransaction);  
}

package za.ac.nwu.as.logic.flow.impl;  
  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Component;  
import za.ac.nwu.as.domain.dto.AccountTransactionDto;  
import za.ac.nwu.as.domain.persistence.AccountTransaction;  
import za.ac.nwu.as.logic.flow.FetchAccountTransactionFlow;  
import za.ac.nwu.as.translator.AccountTransactionTranslator;  
  
import javax.transaction.Transactional;  
import java.util.List;  
  
@Transactional  
@Component  
public class FetchAccountTransactionFlowImpl implements FetchAccountTransactionFlow {  
  
 private final AccountTransactionTranslator accountTransactionTranslator;  
  
 @Autowired  
 public FetchAccountTransactionFlowImpl(AccountTransactionTranslator accountTransactionTranslator) {  
 this.accountTransactionTranslator= accountTransactionTranslator;  
 }  
  
  
 @Override  
 public List<AccountTransactionDto> getAllAccountTransactions() {  
 return accountTransactionTranslator.getAllAccountTransaction();  
 }  
  
 @Override  
 public AccountTransactionDto getAccountTransactionByMemberId(Long memberId){  
 return accountTransactionTranslator.getAccountTransactionDtoByMemberId(memberId);  
 }  
  
  
 @Override  
 public AccountTransaction getAccountTransactionForMemberId(Long memberId){  
 return accountTransactionTranslator.getAccountTransactionByMemberId(memberId);  
 }

package za.ac.nwu.as.logic.flow.impl;  
  
  
import org.springframework.stereotype.Component;  
import za.ac.nwu.as.domain.dto.AccountTransactionDto;  
import za.ac.nwu.as.logic.flow.CreateAccountTransactionFlow;  
import za.ac.nwu.as.translator.AccountTransactionTranslator;  
  
import javax.transaction.Transactional;  
import java.time.LocalDate;  
  
@Transactional  
@Component("createAccountTransactionFlowName")  
public class CreateAccountTransactionFlowImpl implements CreateAccountTransactionFlow {  
  
 private final AccountTransactionTranslator accountTransactionTranslator;  
  
 public CreateAccountTransactionFlowImpl(AccountTransactionTranslator accountTransactionTranslator){  
 this.accountTransactionTranslator = accountTransactionTranslator;  
 }  
  
 @Override  
 public AccountTransactionDto createAdd(AccountTransactionDto accountTransactionAdd) {  
 if (null == accountTransactionAdd.getCreationDates()){  
 accountTransactionAdd.setCreationDates(LocalDate.*now*());  
 }  
 try{  
 if (accountTransactionAdd.getAmounts() > 0){  
 accountTransactionAdd.setAmounts(accountTransactionAdd.getAmounts());  
 }  
 } catch (Exception e) {  
 throw new RuntimeException(e);  
 }  
  
 return accountTransactionTranslator.create(accountTransactionAdd);  
 }  
  
 @Override  
 public AccountTransactionDto createWithdraw(AccountTransactionDto accountTransactionWD) {  
 if (null == accountTransactionWD.getCreationDates()){  
 accountTransactionWD.setCreationDates(LocalDate.*now*());  
 }  
 try{  
 if (accountTransactionWD.getAmounts() > 0){  
 accountTransactionWD.setAmounts(-accountTransactionWD.getAmounts());  
 }  
 } catch (Exception e) {  
 throw new RuntimeException(e);  
 }  
  
 return accountTransactionTranslator.create(accountTransactionWD);  
 }  
}