**Assignment: Implement Logging and Exception Handling in a Funds Transfer Service**

You have been asked to implement a funds transfer service that allows users to transfer money between accounts. Your task is to implement logging and exception handling in the service using Spring Boot AOP.

1. Create a **TransferService** class with a **transferFunds** method that takes in two **Account** objects and a **double** amount, and performs the funds transfer between the two accounts. If an invalid account number is provided, throw an **InvalidAccountException**.
2. Create a Spring Boot aspect called **TransferLoggingAspect** that logs the method invocation, arguments, and result of the **transferFunds** method.
3. Modify the **TransferLoggingAspect** to handle the case where an invalid account number is provided. If an invalid account number is detected, log an error message and throw the **InvalidAccountException**.
4. Create a Spring Boot controller that handles the transfer request from the user. Use the **TransferService** to transfer the funds, and handle any exceptions thrown by the service.
5. Create a Spring Boot exception handler that handles the **InvalidAccountException** and returns an appropriate error message to the user.
6. Test the application by sending a valid transfer request and an invalid transfer request, and verify that the logging and exception handling works as expected.

You can use the following code as a starting point for the **TransferService** and **TransferLoggingAspect** classes:

@Service

public class TransferService {

public void transferFunds(Account fromAccount, Account toAccount, double amount) throws InvalidAccountException {

// Check if toAccount exists

if (!toAccount.isValid()) {

throw new InvalidAccountException("Invalid account number provided");

}

// Perform funds transfer

}

}

@Aspect

@Component

public class TransferLoggingAspect {

private static final Logger logger = LoggerFactory.getLogger(TransferLoggingAspect.class);

@Around("execution(\* com.example.TransferService.transferFunds(..))")

public Object logTransfer(ProceedingJoinPoint joinPoint) throws Throwable {

// Get method arguments

Object[] args = joinPoint.getArgs();

Account fromAccount = (Account) args[0];

Account toAccount = (Account) args[1];

double amount = (double) args[2];

// Check if toAccount exists

if (!toAccount.isValid()) {

logger.error("Invalid account number provided for funds transfer");

throw new InvalidAccountException("Invalid account number provided");

}

// Log method invocation

logger.info("Transfering funds from account {} to account {} for amount {}",

fromAccount.getAccountNumber(), toAccount.getAccountNumber(), amount);

// Call the original method

Object result = joinPoint.proceed();

// Log method result

logger.info("Transfer completed successfully");

return result;

}

}