|  |  |
| --- | --- |
| Course | Advanced Software Design – CS525 |
| Assignment | Lab 0 |
| Week | 08 |
| Due | 8/24/2025 |
| Student name | Toe Toe Aung |
| Student ID | 618090 |

# a. Proxy pattern of the given bank application

1. Write a logging proxy that logs all method calls on the AccountDAO class.

2. Write a timing proxy that computes the time for every service level method we

invoke

3. Apply the timing proxy also for DAO method calls.

**public** **class** Logger {

**public** **void** log(String msg) {

System.***out***.println("Logger: " + msg);

}

}

**public** **class** LoggingProxy **implements** InvocationHandler {

Object obj;

Logger log = **new** Logger();

**public** LoggingProxy(Object obj) {

**this**.obj = obj;

}

@Override

**public** Object invoke(Object proxy, Method method, Object[] args) **throws** Throwable {

log.log(method.getName());

Object returnObj = method.invoke(obj, args);

**return** returnObj;

}

}

**public** **class** Timing {

**public** **void** timeNow() {

DateTimeFormatter dtf = DateTimeFormatter.*ofPattern*("yyyy/MM/dd HH:mm:ss");

LocalDateTime now = LocalDateTime.*now*();

System.***out***.println("Your Action at " + dtf.format(now));

}

}

**public** **class** TimingProxy **implements** InvocationHandler {

Object obj;

**public** TimingProxy(Object obj) {

**this**.obj = obj;

}

@Override

**public** Object invoke(Object proxy, Method method, Object[] args) **throws** Throwable {

Timing time = **new** Timing();

time.timeNow();

Object returnValue = method.invoke(obj, args);

**return** returnValue;

}

}

# b. Adapter pattern

Modify the code from part a (proxy pattern) so that the client is not coupled to the

domain objects. So the service methods getAccount and getAccounts should not return

the domain class Account, but an AccountDTO. The adapter should to the conversion

between Account and AccountDTO.

public interface ConverterAdapter {

void setFromType(String fromType);

void setValue(int value);

void convert();

}

public class CustomaryAdaptee implements ConverterAdapter {

String fromType;

int value;

@Override

public void setFromType(String fromType) {

this.fromType = fromType;

}

@Override

public void setValue(int value) {

this.value = value;

}

@Override

public void convert() {

if (this.fromType.equals("lb")) {

CustomaryMassConverter cmc = new CustomaryMassConverter(this.value);

cmc.convert();

} else if (this.fromType.equals("yd")) {

CustomaryLengthConverter clc = new CustomaryLengthConverter(this.value);

clc.convert();

}

}

}

public class CustomaryLengthConverter {

int value;

public CustomaryLengthConverter(int value) {

this.value = value;

}

public void convert() {

System.out.println("In meter: " + this.value \* 0.9144);

}

}

public class CustomaryMassConverter {

int value;

public CustomaryMassConverter(int value) {

this.value = value;

}

public void convert() {

System.out.println("In kg: " + this.value \* 0.453592);

}

}