|  |  |
| --- | --- |
| Course | Advanced Software Design – CS525 |
| Assignment | Lab 8 |
| Week | 08 |
| Due | Apr 6, 2020 |
| Student name | Quan Hong Doan |
| Student ID | 986956 |

Online version:

<https://github.com/zithiat/asd/blob/master/labs%20%26%20assignments/Answers/Assignment_CS525_Week08_986956.docx>

**Problem 1**:

Problem 1 is the question of bank code from the PDF file.

**Answer**:

The source code on my GitHub <https://github.com/zithiat/asd/tree/master/codes/code%20for%20labs/lab8/bank>

The code changes at service and DAO level.

**AccountService**

**public** **class** AccountService **implements** IAccountService {

**private** IAccountDAO accountDAO = **new** AccountDAO();

ClassLoader loader = IAccountDAO.**class**.getClassLoader();

IAccountDAO loggingProxy = (IAccountDAO) Proxy.*newProxyInstance*(loader, **new** Class[] { IAccountDAO.**class** }, **new** LoggingProxy(accountDAO));

IAccountDAO timingProxy = (IAccountDAO) Proxy.*newProxyInstance*(loader, **new** Class[] { IAccountDAO.**class** }, **new** TimingProxy(loggingProxy));

**public** Account createAccount(**long** accountNumber, String customerName) {

Account account = **new** Account(accountNumber);

Customer customer = **new** Customer(customerName);

account.setCustomer(customer);

timingProxy.saveAccount(account);

**return** account;

}

**public** **void** deposit(**long** accountNumber, **double** amount) {

Account account = accountDAO.loadAccount(accountNumber);

account.deposit(amount);

timingProxy.updateAccount(account);

}

**public** Account getAccount(**long** accountNumber) {

Account account = timingProxy.loadAccount(accountNumber);

**return** account;

}

**public** Collection<Account> getAllAccounts() {

**return** timingProxy.getAccounts();

}

**public** **void** withdraw(**long** accountNumber, **double** amount) {

Account account = timingProxy.loadAccount(accountNumber);

account.withdraw(amount);

timingProxy.updateAccount(account);

}

**public** **void** transferFunds(**long** fromAccountNumber, **long** toAccountNumber, **double** amount, String description) {

Account fromAccount = timingProxy.loadAccount(fromAccountNumber);

Account toAccount = timingProxy.loadAccount(toAccountNumber);

fromAccount.transferFunds(toAccount, amount, description);

timingProxy.updateAccount(fromAccount);

timingProxy.updateAccount(toAccount);

}

}

**LoggingProxy**

**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;

}

}

**Logger**

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

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

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

}

}

**TimingProxy**

**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;

}

}

**Timing**

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

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

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

LocalDateTime now = LocalDateTime.*now*();

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

}

}

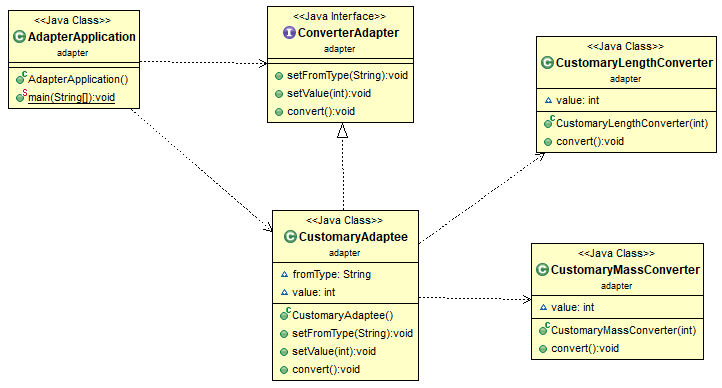
**Problem 2**:

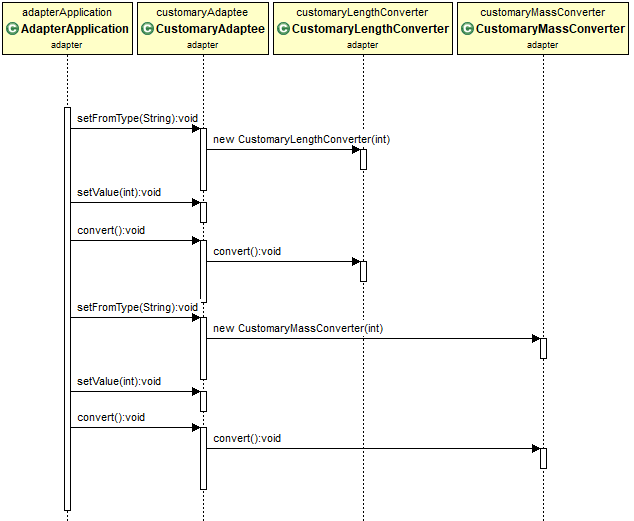
Problem 2 is the question of Adapter pattern from the PDF file.

**Answer**:

The source code on my GitHub <https://github.com/zithiat/asd/tree/master/codes/code%20for%20labs/lab8/MyProxyAdapter>

The question is similar in my testing implementation. Same approach and same way of solution.





**AdapterApplication**

**public** **class** AdapterApplication {

**public** **static** **void** main(String[] args) {

ConverterAdapter adapter = **new** CustomaryAdaptee();

adapter.setFromType("lb");

adapter.setValue(10);

adapter.convert();

adapter.setFromType("yd");

adapter.setValue(100);

adapter.convert();

}

}

**ConverterAdapter**

**public** **interface** ConverterAdapter {

**void** setFromType(String fromType);

**void** setValue(**int** value);

**void** convert();

}

**CustomaryAdaptee**

**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();

}

}

}

**CustomaryLengthConverter**

**public** **class** CustomaryLengthConverter {

**int** value;

**public** CustomaryLengthConverter(**int** value) {

**this**.value = value;

}

**public** **void** convert() {

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

}

}

**CustomaryMassConverter**

**public** **class** CustomaryMassConverter {

**int** value;

**public** CustomaryMassConverter(**int** value) {

**this**.value = value;

}

**public** **void** convert() {

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

}

}