|  |  |
| --- | --- |
| Course | Advanced Software Design – CS525 |
| Assignment | Lab 9 |
| Week | 09 |
| Due | 8/31/2025 |
| Student name | Toe Toe Aung |
| Student ID | 618090 |

# a. Factory Pattern

An abstract factory class with the following functionality:

1. Whenever the amount of an account changes, we send an email to the customer

(simulate with System.out.println()).

2. For testing, we want to use a MockAccountDAO and a MockEmailSender

3. For production we use the AccountDAO and en EmailSender class

4. We specify the environment (production or test) in config.properties file.

**public** **class** DaoFactory {

**private** BankFactory factory;

**public** DaoFactory() {

**try** (InputStream in = **new** FileInputStream(**new** File("").getAbsolutePath() + "/config.properties")) {

Properties prop = **new** Properties();

prop.load(in);

String env = prop.getProperty("db.password");

**if** (env.equals("test")) {

factory = **new** MockFactory();

} **else** **if** (env.equals("production")) {

factory = **new** ProductionFactory();

} **else** {

System.***out***.println("Checking the environment configuration:");

}

} **catch** (IOException ex) {

ex.printStackTrace();

}

}

**public** BankFactory getFactory() {

**return** factory;

}

}

**public** **interface** BankFactory {

IAccountDAO getAccountDAO();

EmailDAO getEmailDAO();

}

**public** **class** MockFactory **implements** BankFactory {

@Override

**public** IAccountDAO getAccountDAO() {

**return** **new** MockAccountDAO();

}

@Override

**public** EmailDAO getEmailDAO() {

**return** **new** MockEmailSender();

}

}

**public** **class** ProductionFactory **implements** BankFactory {

@Override

**public** IAccountDAO getAccountDAO() {

**return** **new** AccountDAOLogger();

}

@Override

**public** EmailDAO getEmailDAO() {

**return** EmailSender.*getInstance*();

}

}

# b. Builder pattern

Modification of the code so that we use a Builder to create the AccountEntry objects

**package** bank.domain;

**import** java.util.Date;

**public** **class** AccountEntry {

**private** Date date;

**private** **double** amount;

**private** String description;

**private** String fromAccountNumber;

**private** String fromPersonName;

**public** AccountEntry() {

}

**public** AccountEntry(Date date, **double** amount, String description, String fromAccountNumber, String fromPersonName) {

**super**();

**this**.date = date;

**this**.amount = amount;

**this**.description = description;

**this**.fromAccountNumber = fromAccountNumber;

**this**.fromPersonName = fromPersonName;

}

**public** **double** getAmount() {

**return** amount;

}

**public** **void** setAmount(**double** amount) {

**this**.amount = amount;

}

**public** Date getDate() {

**return** date;

}

**public** **void** setDate(Date date) {

**this**.date = date;

}

**public** String getDescription() {

**return** description;

}

**public** **void** setDescription(String description) {

**this**.description = description;

}

**public** String getFromAccountNumber() {

**return** fromAccountNumber;

}

**public** **void** setFromAccountNumber(String fromAccountNumber) {

**this**.fromAccountNumber = fromAccountNumber;

}

**public** String getFromPersonName() {

**return** fromPersonName;

}

**public** **void** setFromPersonName(String fromPersonName) {

**this**.fromPersonName = fromPersonName;

}

**public** AccountEntry(Builder builder) {

**this**.date = builder.date;

**this**.amount = builder.amount;

**this**.description = builder.description;

**this**.fromAccountNumber = builder.fromAccountNumber;

**this**.fromPersonName = builder.fromPersonName;

}

**public** **static** **class** Builder {

Date date;

**double** amount;

String description;

String fromAccountNumber;

String fromPersonName;

**public** Builder withDate(Date d) {

**this**.date = d;

**return** **this**;

}

**public** Builder withAmount(**double** a) {

**this**.amount = a;

**return** **this**;

}

**public** Builder withDescription(String desc) {

**this**.description = desc;

**return** **this**;

}

**public** Builder withFromAccount(String fromAccount) {

**this**.fromAccountNumber = fromAccount;

**return** **this**;

}

**public** Builder withFromName(String fromName) {

**this**.fromPersonName = fromName;

**return** **this**;

}

**public** AccountEntry build() {

**return** **new** AccountEntry(**this**);

}

}

}

# c. Singleton pattern

Modification the code so that the EmailSender is a singleton that is thread safe

**package** bank.dao;

**public** **class** EmailSender **implements** EmailDAO {

**private** **static** EmailDAO *sender*;

**private** EmailSender() {

**if** (*sender* != **null**) {

**throw** **new** RuntimeException("It has already initiated, use getInstance() to get the instance instead.");

}

}

**public** **static** EmailDAO getInstance() {

**if** (*sender* == **null**) {

**synchronized** (EmailSender.**class**) {

**if** (*sender* == **null**) {

*sender* = **new** EmailSender();

}

}

}

**return** *sender*;

}

@Override

**public** **void** sendEmail() {

System.***out***.println("Email Sender");

}

}

# d. Decorator pattern

Decorate the AccountDAO with an AccountDAOLogger that logs all actions we

perform on the AccountDAO.

**public** **class** AccountDAOLogger **extends** AccountDAO {

**public** AccountDAOLogger() {

**super**();

}

**public** **void** saveAccount(Account account) {

System.***out***.println("AccountDAOLogger : saveAccount");

**super**.saveAccount(account);

}

**public** **void** updateAccount(Account account) {

System.***out***.println("AccountDAOLogger : updateAccount");

**super**.updateAccount(account);

}

**public** Account loadAccount(**long** accountnumber) {

System.***out***.println("AccountDAOLogger : loadAccount");

**return** **super**.loadAccount(accountnumber);

}

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

System.***out***.println("AccountDAOLogger : getAccounts");

**return** **super**.getAccounts();

}

}