

**Design Book Store**

โดย

รหัสประจำตัว 603020290-9 นางสาวณิชกานต์ พลหาญ

รหัสประจำตัว 603020305-2 นายปรเมษฐ์ ทองคนทา

รหัสประจำตัว 603021712-4 นางสาวอรอนงค์ ทะวงษ์ศรี

อาจารย์ที่ปรึกษา : ผศ.ดร.ปัญญาพล หอระตะ

รายงานนี้เป็นส่วนหนึ่งของการศึกษาวิชา SC313002 หลักการออกแบบพัฒนาซอฟต์แวร์

ภาคเรียนที่ 1 ปีการศึกษา 2562

ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์

มหาวิทยาลัยขอนแก่น

**คำนำ**

รายงาน ฉบับนี้เป็นส่วนหนึ่งของวิชา SC313002 หลักการออกแบบพัฒนาซอฟต์แวร์ นักศึกษาชั้นปีที่ 3 สาขาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยขอนแก่น โดยมีจุดประสงค์ เพื่อการศึกษาความรู้ที่ได้จากเรื่อง Memento Pattern และนำมาประยุกต์กับระบบร้านหนังสือ ทั้งนี้ ในรายงานนี้มีเนื้อหาประกอบด้วยความรู้เกี่ยวกับ ความหมาย การดำเนินงาน หลักการทำงาน ตัวอย่าง source code ตลอดจนการประยุกต์ใช้

ผู้จัดทำได้เลือกหัวข้อนี้ในการทำรายงาน เนื่องมาจากเป็นเรื่องที่น่าสนใจ ผู้จัดทำต้องขอขอบคุณอาจารย์ ปัญญาพล หอระตะ ผู้ให้ความรู้ และแนวทางการศึกษา หวังว่ารายงานฉบับนี้จะให้ความรู้ และเป็นประโยชน์แก่ผู้อ่านทุก ๆ ท่าน หากมีข้อเสนอแนะประการใด ผู้จัดทำขอรับไว้ด้วยความขอบพระคุณยิ่ง

ผู้จัดทำ

ณิชกานต์ พลหาญ

ปรเมษฐ์ ทองคนทา

อรอนงค์ ทะวงษ์ศรี

**สารบัญ**

**เรื่อง หน้า**

Memento Pattern 1

การดำเนินงาน 1

หลักการของ Memento 1

ข้อดี 2

ข้อจำกัด 2

Source code 2

Class Diagram 18

เอกสารอ้างอิง 19

**สารบัญภาพ**

**ภาพที่ หน้า**

ภาพที่ 1 โครงสร้างของ Memento Pattern 1

ภาพที่ 2 Book Store Diagram 18

**Memento Pattern**

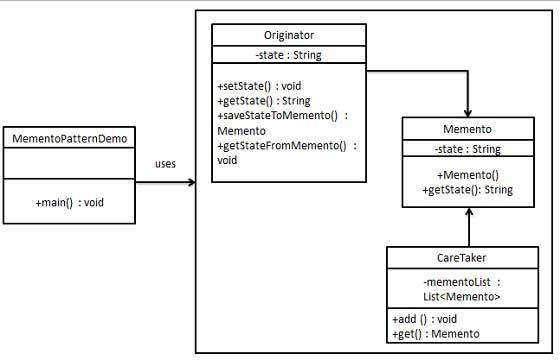
Memento เป็นรูปแบบการออกแบบซอฟต์แวร์ที่ให้ความสามารถในการกู้คืนวัตถุไปยังสถานะก่อนหน้า

**การดำเนินงาน**

รูปแบบของ memento จะใช้คลาสหลัก 3 คลาส คือ Memento, Originator และ CareTaker

* Memento จะทำหน้าที่เก็บสถานะของวัตถุที่จะกู้คืน
* Originator สร้างและจัดเก็บเก็บสถานะของวัตถุ
* CareTaker เป็นคนที่เก็บสถานะต่าง ๆ ของที่ Originator เป็นคนสร้างไว้ให้ และถ้าต้องการย้อน Originator กลับไปที่สถานะไหน ก็แค่ส่ง memento object ไปให้เท่านั้น

**หลักการของ Memento**



**ภาพที่ 1** โครงสร้างของ Memento Pattern

1. ให้ class ที่ต้องการย้อนกลับได้ สร้าง memento object เอาไว้เก็บสถานะต่าง ๆ ของตัวเองไว้
2. สร้าง class ที่เอาไว้เก็บ memento object เพื่อเอาไปทำเป็น history แต่เข้าถึง memento ผ่าน interface
3. เมื่อต้องการย้อนสถานะกลับ ให้ส่ง memento object กลับไปให้ class ที่มีความสามารถย้อนกลับ

**ข้อดี**

* สามารถเก็บสถานะของ object และย้อนสถานะไปมาได้โดยไม่ทำลายความเป็น encapsulation
* แยกหน้าที่ของตัวเก็บข้อมูลต่าง ๆ จาก Originator ได้ เพราะมี caretaker เป็นคนดูแล

**ข้อจำกัด**

* เปลือง Ram เพราะจะมีการเก็บข้อมูลไปเรื่อย ๆ

**Source Code**

* Class Books: ใช้เก็บข้อมูลหนังสือ เช่น ชื่อหนังสือ ราคา ฯลฯ

**package** BookStore;

**public** **class** Books {

**private** Integer price, uniqueID, quantity;

**private** String name;

**private** **static** **int** *count* = 0;

**public** Books(String newName, Integer newPrice, Integer newQuantity ){

**this**.name = newName;

**this**.price = newPrice;

**this**.uniqueID = ++*count*;

**this**.quantity = newQuantity;

}

**public** **void** changeQuantity(**int** change){

**this**.quantity += change;

}

**public** Integer getPrice() {

**return** price;

}

**public** **void** setPrice(Integer price) {

**this**.price = price;

}

**public** Integer getUniqueID() {

**return** uniqueID;

}

**public** **void** setUniqueID(Integer uniqueID) {

**this**.uniqueID = uniqueID;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** Integer getQuantity() {

**return** quantity;

}

**public** **void** setQuantity(Integer quantity) {

**this**.quantity = quantity;

}

}

* Class Memento: ใช้บันทึกสถานะของ object

**package** BookStore;

**import** java.util.ArrayList;

/\*\*

\* This class used to save/get state of inventory

\*/

**public** **class** Memento **implements** java.io.Serializable{

**private** ArrayList<Books> bookCollection;

**public** **void** saveState(ArrayList<Books> newBookCollection){

**this**.bookCollection = **new** ArrayList<Books>(newBookCollection);

}

**public** ArrayList<Books> getState(){

**return** **this**.bookCollection;

}

}

* Class Inventory: จะ implement Inventory\_Interface สามารถกำหนด memento และเรียกใช้ได้

• เพิ่มหนังสือใหม่

• ขายหนังสือในคลัง

• เพิ่มสำเนาหนังสือที่มีอยู่ใหม่

• เปลี่ยนราคาของหนังสือ

• ค้นหาราคาหรือปริมาณหนังสือตามชื่อหรือรหัส

**package** BookStore;

**import** java.io.File;

**import** java.io.IOException;

**import** java.util.ArrayList;

**public class** Inventory **implements** InventoryInterfaces{

**private** ArrayList<Books> book = **new** ArrayList<Books>();

**private** Memento memento = **new** Memento();

**private** InventoryDecorators invent ;

**private** String CommandFileName = "Command.ser";

**private** Integer numberOfState=0, timeToSave =3;

**private** Integer quality;

**private** Integer price;

**public** ArrayList<Books> getBookCollection() {

**return** book;

}

**public** **void** setBookCollection(ArrayList<Books> bookCollection) {

**this.**book = bookCollection;

}

@Override

**public** **void** addBook(Books book) {

**this**.book.add(book);

**if**(++numberOfState == timeToSave){

**this**.saveState();

numberOfState=0;

}

}

@Override

**public** **void** sellBook(String bookName) {

**for**(Books book : book){

**if**(book.getName().equals(bookName) && book.getQuantity() > 0){

book.changeQuantity(-1);

**if**(++numberOfState == timeToSave){

**this**.saveState();

numberOfState=0;

}

**return** ;

}

}

}

@Override

**public** **void** addCopy(String bookName, Integer NumberOfCopy) {

**for**(Books book : book){

**if**(book.getName().equals(bookName)){

book.changeQuantity(NumberOfCopy);

**if**(++numberOfState == timeToSave){

**this**.saveState();

numberOfState=0;

}

**return** ;

}

}

}

@Override

**public** **void** changePrice(String bookName, Integer newPrice) {

**for**(Books book : book){

**if**(book.getName().equals(bookName)){

book.setPrice(newPrice);

**if**(++numberOfState == timeToSave){

**this**.saveState();

numberOfState=0;

}

**return** ;

}

}

}

@Override

**public** Integer findPriceByName(String bookName) {

**for**(Books book : book){

**if**(book.getName().equals(bookName)){

**this**.price = book.getPrice();

}

}

**return** **this**.price;

}

@Override

**public** Integer findQuantityByName(String bookName) {

**for**(Books book : book){

**if**(book.getName().equals(bookName)){

**this**.quality = book.getQuantity();

}

}

**return** **this**.quality;

}

@Override

**public** Integer findQuantityByID(Integer bookID) {

**for**(Books book : book){

**if**(book.getUniqueID().equals(bookID)){

**this**.quality = book.getQuantity();

}

}

**return** **this**.quality;

}

@Override

**public** Integer findPriceByID(Integer bookID) {

**for**(Books book : book){

**if**(book.getUniqueID().equals(bookID)){

**this**.price = book.getPrice();

}

}

**return** **this**.price;

}

@Override

**public** **void** saveState() {

memento.saveState(book);

File file = **new** File(CommandFileName);

file.delete();

**try** {

file.createNewFile();

} **catch** (IOException e) {

e.printStackTrace();

}

}

@Override

**public** **void** getState() {

invent.getState();

book = (invent.getInvent().getBookCollection());

}

}

* Class InventoryInterfaces: ใช้ในการเก็บ method การดำเนินงานต่าง ๆ

**package** BookStore;

**public** **interface** InventoryInterfaces {

**public** **void** addBook(Books book);

**public** **void** sellBook(String bookName);

**public** **void** addCopy(String bookName, Integer NumberOfCopy ) ;

**public** **void** changePrice(String bookName,Integer newPrice);

**public** Integer findPriceByName(String bookName) ;

**public** Integer findQuantityByName(String bookName) ;

**public** Integer findQuantityByID(Integer bookID) ;

**public** Integer findPriceByID(Integer bookID) ;

**public** **void** saveState();

**public** **void** getState();

}

* Class InventoryDecorator: เป็นส่วนเติมเต็มของคลาส Inventory ทำหน้าที่เพิ่มฟังก์ชันในการทำงาน

**package** BookStore;

**import** java.io.EOFException;

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.IOException;

**import** java.io.ObjectInputStream;

**import** java.util.ArrayList;

**public** **class** InventoryDecorators **implements** InventoryInterfaces{

**private** Inventory invent = **new** Inventory();

**private** FileInputStream fileIn ;

**private** ArrayList<Command> commandCollection = **new** ArrayList<Command>();

**private** CareTaker careTaker = **new** CareTaker();

**private** Memento memento = **new** Memento();

**private** String CommandFileName = "Command.ser";

**public** Inventory getInvent() {

**return** invent;

}

**public** **void** setInvent(Inventory invent) {

//นำ invent จากคลาส inventory เพื่อเรียกใช้ method จำพวกที่มีอยู่แต่เดิม

//พวก find ต่างๆ

**this**.invent = invent;

}

@Override

**public** **void** addBook(Books book) {

AddBookCommand addBook= **new** AddBookCommand(book);

addBook.execute(invent);

}

@Override

**public** **void** sellBook(String bookName) {

SellBookCommand sellBook = **new** SellBookCommand(bookName);

sellBook.execute(invent);

}

@Override

**public** **void** addCopy(String bookName, Integer NumberOfCopy) {

AddCopyCommand addCopy = **new** AddCopyCommand(bookName, NumberOfCopy);

addCopy.execute(invent);

}

@Override

**public** **void** changePrice(String bookName, Integer newPrice) {

ChangePriceCommand changePrice = **new** ChangePriceCommand(bookName, newPrice);

changePrice.execute(invent);

}

/\*

\* method จาก inventory

\*/

@Override

**public** Integer findPriceByName(String bookName) {

**return** invent.findPriceByName(bookName);

}

@Override

**public** Integer findQuantityByName(String bookName) {

**return** invent.findQuantityByName(bookName);

}

@Override

**public** Integer findQuantityByID(Integer bookID) {

**return** invent.findQuantityByID(bookID);

}

**private** **void** replyCommands(Inventory invent){

**try** {

fileIn = **new** FileInputStream(CommandFileName);

**while** (**true**) {

ObjectInputStream input = **new** ObjectInputStream(fileIn);

commandCollection.add((Command) input.readObject());

}

}**catch** (EOFException e) {

**try**{

fileIn.close();

}**catch**(IOException i)

{

i.printStackTrace();

}

}**catch**(IOException i)

{

i.printStackTrace();

}**catch**(ClassNotFoundException c)

{

System.***out***.println("class not found");

c.printStackTrace();

}

**for**(Command command : commandCollection){

command.execute(invent);

}

}

@Override

**public** Integer findPriceByID(Integer bookID) {

**return** invent.findQuantityByID(bookID);

}

@Override

**public** **void** saveState() {

memento.saveState(invent.getBookCollection());

careTaker.serialzeMemento(memento);

File file = **new** File(CommandFileName);

file.delete(); //เช็คว่าไฟล์เราถูกลบไหม ?

**try** {

//ถ้าไฟล์ถูกลบ ให้สร้างไฟล์ใหม่

file.createNewFile();

} **catch** (IOException e) {

e.printStackTrace();

}

}

@Override

**public** **void** getState() {

memento = careTaker.deserialseMemento();

invent.setBookCollection(memento.getState());

**this**.replyCommands(invent);

}

}

* Class Command: เป็น abstract class หน้าที่ คือ ประมวลผลเมธอด execute()

**package** BookStore;

**public** **abstract** **class** Command {

**public** **abstract** **void** execute(Inventory newInvent);

}

* Class CareTaker: เป็น ArrayList ทำหน้าที่แสดงลิสต์ของ Memento และคืนค่าสถานะของ object

**package** BookStore;

**import** java.io.File;

**import** java.io.FileInputStream;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**import** java.io.ObjectInputStream;

**import** java.io.ObjectOutputStream;

**public** **class** CareTaker {

**private** Object object;

**private** String temporaryFileName = "temporaryInventory.ser";

**private** String fileName = "Inventory.ser";

**private** File tempFile = **new** File(temporaryFileName);

**private** File file = **new** File(fileName);

**byte**[] buf = **new** **byte**[1024];

**int** bytesRead;

**public** **void** serialzeMemento(Memento state){

**try**

{

FileOutputStream temporaryFileOut = **new** FileOutputStream(temporaryFileName);

ObjectOutputStream out = **new** ObjectOutputStream(temporaryFileOut);

out.writeObject(state);

out.close();

temporaryFileOut.close();

tempFile.renameTo(file);

}**catch**(IOException i)

{

i.printStackTrace();

}

}

**public** Memento deserialseMemento(){

**try**

{

FileInputStream fileIn = **new** FileInputStream(fileName);

ObjectInputStream in = **new** ObjectInputStream(fileIn);

object = in.readObject();

in.close();

fileIn.close();

**return** (Memento)object;

}**catch**(IOException i)

{

i.printStackTrace();

**return** **null**;

}**catch**(ClassNotFoundException c)

{

System.***out***.println("class not found");

c.printStackTrace();

**return** **null**;

}

}

}

* MatchNotFoundException: ทำหน้าที่ดักจับและ checked ความผิดพลาดที่เกิดขึ้น

**package** BookStore;

@SuppressWarnings("serial")

**public** **class** MatchNotFoundException **extends** Exception {

**public** MatchNotFoundException(String s){

**super**(s);

}

}

* AddBookCommand

**package** BookStore;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**import** java.io.ObjectOutputStream;

**public** **class** AddBookCommand **extends** Command **implements** java.io.Serializable{

**private** Books book;

**private** String fileName = "Command.ser";

AddBookCommand(Books newBook){

**this**.book = newBook;

}

@Override

**public** **void** execute(Inventory newInvent) {

newInvent.addBook(book);

**try**

{

FileOutputStream fileOut = **new** FileOutputStream(fileName,**true**);

ObjectOutputStream out = **new** ObjectOutputStream(fileOut);

// out.writeObject(this);

out.writeObject(fileName);

out.close();

fileOut.close();

}**catch**(IOException i){

i.printStackTrace();

}

}

}

* SellBookCommand

**package** BookStore;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**import** java.io.ObjectOutputStream;

@SuppressWarnings("serial")

**public** **class** SellBookCommand **extends** Command **implements** java.io.Serializable{

**private** String bookName;

**private** String fileName = "Command.ser";

**public** SellBookCommand(String bookName){

**this**.bookName = bookName;

}

@Override

**public** **void** execute(Inventory newInvent) {

newInvent.sellBook(bookName);

**try**

{

FileOutputStream fileOut = **new** FileOutputStream(fileName,**true**);

ObjectOutputStream out = **new** ObjectOutputStream(fileOut);

out.writeObject(fileName);

out.close();

fileOut.close();

}**catch**(IOException i)

{

i.printStackTrace();

}

}

}

* ChangePriceCommand

**package** BookStore;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**import** java.io.ObjectOutputStream;

**public** **class** ChangePriceCommand **extends** Command **implements** java.io.Serializable{

**private** String bookName;

**private** Integer numberOfCopy;

**private** String fileName = "Command.ser";

ChangePriceCommand(String newbookName, Integer newNumberOfCopy){

**this**.bookName = newbookName;

**this**.numberOfCopy = newNumberOfCopy;

}

@Override

**public** **void** execute(Inventory newInvent) {

newInvent.changePrice(bookName,numberOfCopy);

**try**

{

FileOutputStream fileOut = **new** FileOutputStream(fileName,**true**);

ObjectOutputStream out = **new** ObjectOutputStream(fileOut);

out.writeObject(fileName);

out.close();

fileOut.close();

}**catch**(IOException i)

{

i.printStackTrace();

}

}

}

* AddCopyCommand

**package** BookStore;

**import** java.io.FileOutputStream;

**import** java.io.IOException;

**import** java.io.ObjectOutputStream;

**public** **class** AddCopyCommand **extends** Command **implements** java.io.Serializable{

**private** String bookName;

**private** Integer numberOfCopy;

**private** String fileName = "Command.ser";

AddCopyCommand(String newbookName, Integer newNumberOfCopy){

**this**.bookName = newbookName;

**this**.numberOfCopy = newNumberOfCopy;

}

@Override

**public** **void** execute(Inventory newInvent) {

newInvent.addCopy(bookName,numberOfCopy);

**try**

{

FileOutputStream fileOut = **new** FileOutputStream(fileName,**true**);

ObjectOutputStream out = **new** ObjectOutputStream(fileOut);

out.writeObject(fileName);

out.close();

fileOut.close();

}**catch**(IOException i)

{

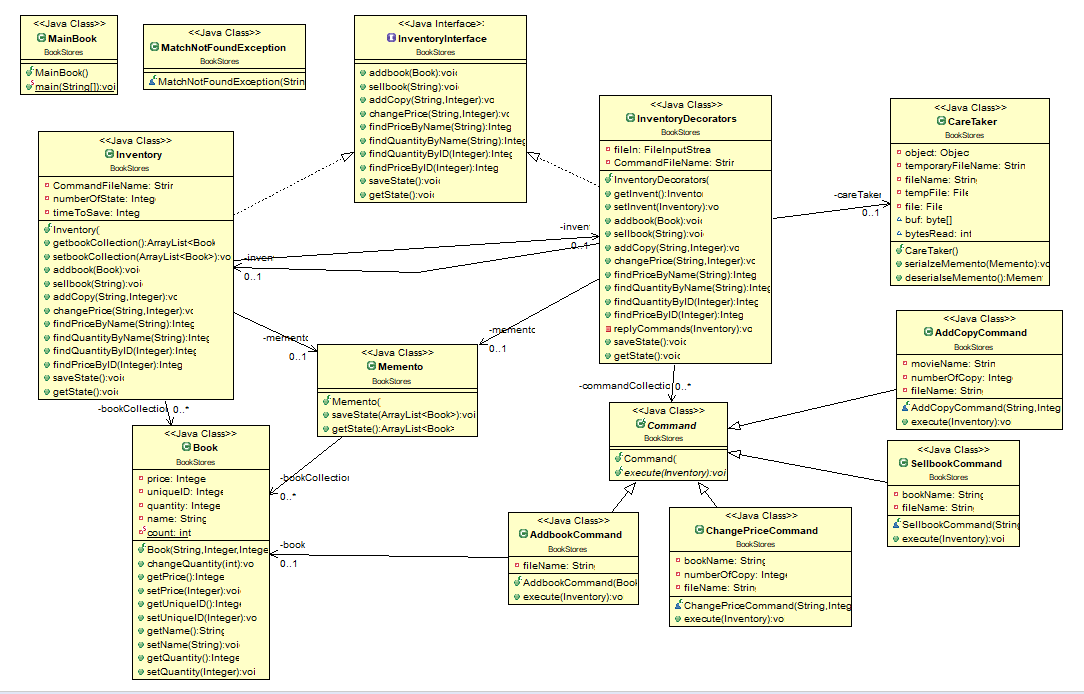
i.printStackTrace();

}

}

}

**Class Diagram**



**ภาพที่ 2** Book Store Diagram

**เอกสารอ้างอิง**

Jaruthanaset S. (2019). Memento. Retrieved November 11, 2019, from

https://saladpuk.gitbook.io/learn/software-design/designpatterns/behavioral-

patterns/memento

Tutorialspoint. [n.p.]. Design Patterns - Memento Pattern. Retrieved November 11, 2019, from

https://www.tutorialspoint.com/design\_pattern/memento\_pattern.htm

ProjectPrinciple. (2019). Retrieved November 11, 2019, from

https://github.com/inwwei/ProjectPrinciple/tree/master/ProjectPrinciple/src/Menoto