Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГТУ»)**

**Институт компьютерных систем и информационной безопасности**

**Кафедра информационных систем и программирования**

**ОТЧЁТ**

**Дисциплина: Тестирование и отладка программного обеспечения**

**Тема: «Рефакторинг с использованием**

**порождающих паттернов проектирования»**

**Лабораторная: 6**

Работу выполнил: Басма Нассиф Хуссейн

Направление подготовки: 09.03.04 Программная инженерия

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е. В. Степанова

**Цель работы:** изучить техники рефакторинга с использованием порождающих паттернов проектирования.

**Контрольные вопросы**

**1) Что такое паттерн проектирования?**

Это повторяемое решение часто встречающейся проблемы в контексте архитектуры программного обеспечения. Паттерны помогают стандартизировать решения, упростить процесс разработки и улучшить коммуникацию между разработчиками за счёт общего словаря.

**2) Для чего используется паттерн Factory method?**

Используется для создания объектов без спецификации конкретных классов. Паттерн определяет интерфейс создания объекта, но позволяет подклассам изменять тип создаваемых объектов. Это обеспечивает гибкость и расширяемость кода, поскольку добавление новых классов продуктов не требует изменений в клиентском коде.

**3) Как проверять результаты работы программы в случае ее неприспособленности для автоматизированного тестирования?**

Для проверки результатов работы программы, не приспособленной для автоматизированного тестирования, можно использовать следующие подходы:

• Ручное тестирование: Прямое взаимодействие с программой для проверки её функциональности и поведения.

• Логирование: Вывод информации о работе программы в логи для последующего анализа.

• Отладка: Использование отладчиков для пошагового выполнения программы и наблюдения за изменениями в её состоянии.

• Exploratory Testing: Исследовательское тестирование, при котором тестировщик активно исследует программу в поисках ошибок без заранее подготовленных тестовых сценариев.

• Код-ревью и парное программирование: Ручная проверка кода другими разработчиками может помочь выявить ошибки и улучшить качество программы.

**Задание**

**1) Преобразовать процесс создания объектов в методе Main класса Program в соответствии с предложенными в пункте 2.2 изменениями, применив описанные в теории методы рефакторинга.**

1. **Перед началом преобразований необходимо добавить тесты для проверки создания объектов на основе заданного файла. Однако код создания объекта совершенно не приспособлен для тестирования, так как взаимодействует непосредственно с файлом для ввода и консолью для вывода. Поэтому нам необходимо выполнить некоторые предварительные преобразования для того, чтобы привести этот код в состояние, пригодное для тестирования. Для этого можно воспользоваться рефакторингом «Выделение метода». Выделим почти весь код основного метода Main так, чтобы он возвращал экземпляр класса Bill (вставив в Main только определение имени файла, его создание и вывод строки из метода statement на экран). Назовем его CreateBill и параметризуем TextReader для организации доступа к строкам файла.**

public class Program

{

static void Main(string[] args)

{

string filename = "BillInfo.yaml";

if (args.Length == 1)

filename = args[0];

using (FileStream fs = new FileStream(filename, FileMode.Open))

using (StreamReader sr = new StreamReader(fs))

{

BillGenerator bill = CreateBill(sr);

string billOutput = bill.GetBIll();

Console.WriteLine(billOutput);

}

}

//---Метод для преобразования данных из файла

public static BillGenerator CreateBill(TextReader sr)

{

// read customer

string line = sr.ReadLine();

string[] result = line.Split(':');

string name = result[1].Trim();

// read bonus

line = sr.ReadLine();

result = line.Split(':');

int bonus = Convert.ToInt32(result[1].Trim());

Customer customer = new Customer(name, bonus);

IView view = new TxtView();

BillGenerator b = new BillGenerator(customer, view);

// read goods count

line = sr.ReadLine();

result = line.Split(':');

int goodsQty = Convert.ToInt32(result[1].Trim());

Goods[] g = new Goods[goodsQty];

for (int i = 0; i < g.Length; i++)

{

// Пропустить комментарии

do

{

line = sr.ReadLine();

} while (line.StartsWith("#"));

result = line.Split(':');

result = result[1].Trim().Split();

string type = result[1].Trim();

switch (type)

{

case "REG":

g[i] = new RegularGoods(result[0]);

break;

case "SAL":

g[i] = new SaleGoods(result[0]);

break;

case "SPO":

g[i] = new SpecialGoods(result[0]);

break;

}

}

// read items count

// Пропустить комментарии

do

{

line = sr.ReadLine();

} while (line.StartsWith("#"));

result = line.Split(':');

int itemsQty = Convert.ToInt32(result[1].Trim());

for (int i = 0; i < itemsQty; i++)

{

// Пропустить комментарии

do

{

line = sr.ReadLine();

} while (line.StartsWith("#"));

result = line.Split(':');

result = result[1].Trim().Split();

int gid = Convert.ToInt32(result[0].Trim());

double price = Convert.ToDouble(result[1].Trim());

int qty = Convert.ToInt32(result[2].Trim());

b.addGoods(new Item(g[gid - 1], qty, price));

}

return b;

}

}

**После этого необходимо добавить тесты фиксирующие работу текущего парсинга файлов с использованием StringReader в качестве источника данных.**

[Test]

public void OneGoodTestCreateBill()

{

string testData = "CustomerName: Test\r\n" +

"CustomerBonus: 10\r\n" +

"GoodsTotalCount: 1\r\n" +

"# ID: NAME TYPE(REG/SAL/SPO)\r\n" +

"1: Cola REG\r\n" +

"ItemsTotalCount: 1\r\n" +

"# ID: GID PRICE QTY\r\n" +

"1: 1 80 20";

using (StringReader sr = new StringReader(testData))

{

BillGenerator billGenerator = Program.CreateBill(sr);

var result = billGenerator.GetBIll();

Assert.IsTrue(result.Contains("Сумма счета составляет 1542"));

Assert.IsTrue(result.Contains("Вы заработали 80 бонусных балов"));

}

}

[Test]

public void TestBillGeneratorCreation()

{

string testData = "CustomerName: Test\r\n" +

"CustomerBonus: 10\r\n" +

"GoodsTotalCount: 3\r\n" +

"# ID: NAME TYPE(REG/SAL/SPO)\r\n" +

"1: Cola REG\r\n" +

"2: Pepsi SAL\r\n" +

"3: Fanta SPO\r\n" +

"ItemsTotalCount: 3\r\n" +

"# ID: GID PRICE QTY\r\n" +

"1: 1 65 6\r\n" +

"2: 2 50 3\r\n" +

"3: 3 35 1";

using (StringReader sr = new StringReader(testData))

{

BillGenerator billGenerator = Program.CreateBill(sr);

var result = billGenerator.GetBIll();

Assert.IsTrue(result.Contains("Сумма счета составляет 553,3"));

Assert.IsTrue(result.Contains("Вы заработали 20 бонусных балов"));

}

}

1. **Создадим новый класс GoodsFactory (в отдельном файле в проекте).**

public class GoodsFactory { }

1. **Создадим параметризованный метод Create в классе GoodsFactory, и определим в нем процесс создания экземпляров наследников класса Goods. Поскольку в дальнейшем нам будет необходима поддержка различных форматов входных файлов, то для доступа к информации, хранимой в файле необходимо выделить код обработки файла в отдельный класс. Поскольку в дальнейшем нам будет необходима поддержка различных форматов входных файлов, то для доступа к информации, хранимой в файле необходимо выделить код обработки файла в отдельный класс.**

public Goods Create(string type, string title)

{

switch (type)

{

case "REG":

return new RegularGoods(title);

case "SAL":

return new SaleGoods(title);

case "SPO":

return new SpecialGoods(title);

default:

throw new ArgumentException("Неизвестный тип товара");

}

}

// PROGRAM

GoodsFactory goodsFactory = new GoodsFactory();

g[i] = goodsFactory.Create(type, result[0]);

1. **Создадим новый класс ContentFile (в отдельном файле в проекте).**

public class ContentFile { }

1. **Определим интерфейс нового класса. Нам потребуются методы SetSource, GetCustomer, GetGoodsCount, GetNextGood, GetItemsCount, GetNextItem, которые будут последовательно вызываться из метода CreateBill, содержащего последовательность вызовов методов. Создание экземпляров наследников класса Goods возложено на метод Create класса GoodsFactory.**

public class ContentFile

{

private TextReader reader;

string line;

string[] result;

// get source

public void SetSource(TextReader reader)

{

this.reader = reader;

}

// read customer info

public Customer GetCustomer()

{

// read customer

line = reader.ReadLine();

result = line.Split(':');

string name = result[1].Trim();

// read bonus

line = reader.ReadLine();

result = line.Split(':');

int bonus = Convert.ToInt32(result[1].Trim());

return new Customer(name, bonus);

}

// returns goods count

public int GetGoodsCount()

{

line = reader.ReadLine();

result = line.Split(':');

return Convert.ToInt32(result[1].Trim());

}

// reads each product

public Goods GetNextGood()

{

GoodsFactory factory = new GoodsFactory();

// Пропустить комментарии

do

{

line = reader.ReadLine();

} while (line.StartsWith("#"));

result = line.Split(':');

result = result[1].Trim().Split();

string type = result[1].Trim();

return factory.Create(type, result[0]);

}

// reads the amount of products

public int GetItemsCount()

{

do

{

line = reader.ReadLine();

} while (line.StartsWith("#"));

result = line.Split(':');

return Convert.ToInt32(result[1].Trim());

}

// reads each item

public Item GetNextItem(Goods[] g)

{

// Пропустить комментарии

do

{

line = reader.ReadLine();

} while (line.StartsWith("#"));

result = line.Split(':');

result = result[1].Trim().Split();

int gid = Convert.ToInt32(result[0].Trim());

double price = Convert.ToDouble(result[1].Trim());

int qty = Convert.ToInt32(result[2].Trim());

return new Item(g[gid - 1], qty, price);

}

}

// PROGRAM

public static BillGenerator CreateBill(TextReader sr)

{

ContentFile contentFile = new ContentFile();

Customer customer = contentFile.GetCustomer();

IView view = new TxtView();

BillGenerator b = new BillGenerator(customer, view);

// read goods count

int goodsQty = contentFile.GetGoodsCount();

Goods[] g = new Goods[goodsQty];

for (int i = 0; i < g.Length; i++)

{

g[i] = contentFile.GetNextGood();

}

// read items count

int itemsQty = contentFile.GetGoodsCount();

for (int i = 0; i < itemsQty; i++)

{

b.addGoods(contentFile.GetNextItem(g));

}

return b;

}

1. **Так как на данном этапе нет необходимости в различных форматах файлов, то на данном у нас будет иметься только одна реализация, работающая с YAML файлами.**

string filename = "BillInfo.yaml";

1. **В классе ContentFile также можно создать вспомогательные методы для работы с файлом (например, получение следующей строки с пропуском комментариев GetNextLine).**

// skip comments

public void GetNextLine()

{

do

{

line = reader.ReadLine();

}

while (line.StartsWith('#'));

}

1. **В основной части программы, внутри выделенного для тестирования метода, заменим создание объекта Bill на создание ContentFile и получение от него объекта Bill.**

//---Метод для преобразования данных из файла

public static BillGenerator CreateBill(TextReader sr)

{

ContentFile contentFile = new ContentFile();

contentFile.SetSource(sr);

Customer customer = contentFile.GetCustomer();

IView view = new TxtView();

BillGenerator b = new BillGenerator(customer, view);

// read goods count

int goodsQty = contentFile.GetGoodsCount();

Goods[] g = new Goods[goodsQty];

for (int i = 0; i < g.Length; i++)

{

g[i] = contentFile.GetNextGood();

}

// read items count

int itemsQty = contentFile.GetGoodsCount();

for (int i = 0; i < itemsQty; i++)

{

b.addGoods(contentFile.GetNextItem(g));

}

return b;

}

1. **Создадим новый класс BillFactory (в отдельном файле в проекте) перенеся туда код CreateBill.**

public class BillFactory

{

//---Метод для преобразования данных из файла

public BillGenerator CreateBill(TextReader sr)

{

ContentFile contentFile = new ContentFile();

contentFile.SetSource(sr);

Customer customer = contentFile.GetCustomer();

IView view = new TxtView();

BillGenerator b = new BillGenerator(customer, view);

// read goods count

int goodsQty = contentFile.GetGoodsCount();

Goods[] g = new Goods[goodsQty];

for (int i = 0; i < g.Length; i++)

{

g[i] = contentFile.GetNextGood();

}

// read items count

int itemsQty = contentFile.GetGoodsCount();

for (int i = 0; i < itemsQty; i++)

{

b.addGoods(contentFile.GetNextItem(g));

}

return b;

}

}

1. **Протестируем вновь создаваемые объекты с помощью уже существующих тестов.**



Рисунок 1 – Результат тестирование

**2) Проверить согласованность результатов и корректность работы нового варианта программы с результатами старой версии.**

**До:**

static void Main(string[] args)

{

string filename = "BillInfo.yaml";

if (args.Length == 1)

filename = args[0];

FileStream fs = new FileStream(filename, FileMode.Open);

StreamReader sr = new StreamReader(fs);

// read customer

string line = sr.ReadLine();

string[] result = line.Split(':');

string name = result[1].Trim();

// read bonus

line = sr.ReadLine();

result = line.Split(':');

int bonus = Convert.ToInt32(result[1].Trim());

Customer customer = new Customer(name, bonus);

IView view = new TxtView();

BillGenerator b = new BillGenerator(customer, view);

// read goods count

line = sr.ReadLine();

result = line.Split(':');

int goodsQty = Convert.ToInt32(result[1].Trim());

Goods[] g = new Goods[goodsQty];

for (int i = 0; i < g.Length; i++)

{

// Пропустить комментарии

do

{

line = sr.ReadLine();

} while (line.StartsWith("#"));

result = line.Split(':');

result = result[1].Trim().Split();

string type = result[1].Trim();

switch (type)

{

case "REG":

g[i] = new RegularGoods(result[0]);

break;

case "SAL":

g[i] = new SaleGoods(result[0]);

break;

case "SPO":

g[i] = new SpecialGoods(result[0]);

break;

}

}

// read items count

// Пропустить комментарии

do

{

line = sr.ReadLine();

} while (line.StartsWith("#"));

result = line.Split(':');

int itemsQty = Convert.ToInt32(result[1].Trim());

for (int i = 0; i < itemsQty; i++)

{

// Пропустить комментарии

do

{

line = sr.ReadLine();

} while (line.StartsWith("#"));

result = line.Split(':');

result = result[1].Trim().Split();

int gid = Convert.ToInt32(result[0].Trim());

double price = Convert.ToDouble(result[1].Trim());

int qty = Convert.ToInt32(result[2].Trim());

b.addGoods(new Item(g[gid - 1], qty, price));

}

string bill = b.GetBIll();

Console.WriteLine(bill);

}

**После:**

// PROGRAM

public class Program

{

static void Main(string[] args)

{

string filename = "BillInfo.yaml";

if (args.Length == 1)

filename = args[0];

using (FileStream fs = new FileStream(filename, FileMode.Open))

using (StreamReader sr = new StreamReader(fs))

{

BillFactory factory = new BillFactory();

BillGenerator bill = factory.CreateBill(sr);

string billOutput = bill.GetBIll();

Console.WriteLine(billOutput);

}

}

}

// UNITTEST1

private BillGenerator bill;

private Customer customer;

private BillFactory billFactory;

[SetUp]

public void Setup()

{

billFactory = new BillFactory();

customer = new Customer("Тестовый клиент",10);

IView view = new TxtView();

bill = new BillGenerator(customer, view);

}

[Test]

public void OneGoodTestCreateBill()

{

string testData = "CustomerName: Test\r\n" +

"CustomerBonus: 10\r\n" +

"GoodsTotalCount: 1\r\n" +

"# ID: NAME TYPE(REG/SAL/SPO)\r\n" +

"1: Cola REG\r\n" +

"ItemsTotalCount: 1\r\n" +

"# ID: GID PRICE QTY\r\n" +

"1: 1 80 20";

using (StringReader sr = new StringReader(testData))

{

BillGenerator billGenerator = billFactory.CreateBill(sr);

var result = billGenerator.GetBIll();

Assert.IsTrue(result.Contains("Сумма счета составляет 1542"));

Assert.IsTrue(result.Contains("Вы заработали 80 бонусных балов"));

}

}

[Test]

public void TestBillGeneratorCreation()

{

string testData = "CustomerName: Test\r\n" +

"CustomerBonus: 10\r\n" +

"GoodsTotalCount: 3\r\n" +

"# ID: NAME TYPE(REG/SAL/SPO)\r\n" +

"1: Cola REG\r\n" +

"2: Pepsi SAL\r\n" +

"3: Fanta SPO\r\n" +

"ItemsTotalCount: 3\r\n" +

"# ID: GID PRICE QTY\r\n" +

"1: 1 65 6\r\n" +

"2: 2 50 3\r\n" +

"3: 3 35 1";

using (StringReader sr = new StringReader(testData))

{

BillGenerator billGenerator = billFactory.CreateBill(sr);

var result = billGenerator.GetBIll();

Assert.IsTrue(result.Contains("Сумма счета составляет 553,3"));

Assert.IsTrue(result.Contains("Вы заработали 20 бонусных балов"));

}

}

// CONTENTFILE

public class ContentFile

{

private TextReader reader;

string line;

string[] result;

// get source

public void SetSource(TextReader reader)

{

this.reader = reader;

}

// read customer info

public Customer GetCustomer()

{

// read customer

GetNextLine();

result = line.Split(':');

string name = result[1].Trim();

// read bonus

GetNextLine();

result = line.Split(':');

int bonus = Convert.ToInt32(result[1].Trim());

return new Customer(name, bonus);

}

// returns goods count

public int GetGoodsCount()

{

GetNextLine();

result = line.Split(':');

return Convert.ToInt32(result[1].Trim());

}

// reads each product

public Goods GetNextGood()

{

GoodsFactory factory = new GoodsFactory();

GetNextLine();

result = line.Split(':');

result = result[1].Trim().Split();

string type = result[1].Trim();

return factory.Create(type, result[0]);

}

// reads the amount of products

public int GetItemsCount()

{

GetNextLine();

result = line.Split(':');

return Convert.ToInt32(result[1].Trim());

}

// reads each item

public Item GetNextItem(Goods[] g)

{

GetNextLine();

result = line.Split(':');

result = result[1].Trim().Split();

int gid = Convert.ToInt32(result[0].Trim());

double price = Convert.ToDouble(result[1].Trim());

int qty = Convert.ToInt32(result[2].Trim());

return new Item(g[gid - 1], qty, price);

}

// skip comments

public void GetNextLine()

{

do

{

line = reader.ReadLine();

}

while (line.StartsWith('#'));

}

}

// GOODSFACTORY

public class GoodsFactory

{

public Goods Create(string type, string title)

{

switch (type)

{

case "REG":

return new RegularGoods(title);

case "SAL":

return new SaleGoods(title);

case "SPO":

return new SpecialGoods(title);

default:

throw new ArgumentException("Неизвестный тип товара");

}

}

}

// BILLFACTORY

public class BillFactory

{

//---Метод для преобразования данных из файла

public BillGenerator CreateBill(TextReader sr)

{

ContentFile contentFile = new ContentFile();

contentFile.SetSource(sr);

Customer customer = contentFile.GetCustomer();

IView view = new TxtView();

BillGenerator b = new BillGenerator(customer, view);

// read goods count

int goodsQty = contentFile.GetGoodsCount();

Goods[] g = new Goods[goodsQty];

for (int i = 0; i < g.Length; i++)

{

g[i] = contentFile.GetNextGood();

}

// read items count

int itemsQty = contentFile.GetGoodsCount();

for (int i = 0; i < itemsQty; i++)

{

b.addGoods(contentFile.GetNextItem(g));

}

return b;

}

}

**Заключение:**

Были изучены техники рефакторинга с использованием порождающих паттернов проектирования.