Лабораторная работа № 1. Порождающие паттерны.

1.1 Разработка приложения с использованием паттерна Singleton

Создать файл настроек для приложения config.properties (использовать класс java.util.Properties для его чтения). Написать класс с использованием паттерна Singleton, который будет загружать данный файл (один раз) и отдавать экземпляр Properties по запросу. Продемонстрировать работу в методе main() через вывод считанных настроек в консоль.

1.2 Разработка приложения с использованием паттерна Factory Method

Написать класс Автомобиль. Он должен содержать:

поле типа String, хранящее марку автомобиля,

метод для получения марки автомобиля,

метод для модификации марки автомобиля,

внутренний класс Модель, имеющий поля название модели и её цену, а также конструктор (класс Автомобиль хранит массив Моделей),

метод для модификации значения названия модели,

метод, возвращающий массив названий всех моделей,

метод для получения значения цены модели по её названию,

метод для модификации значения цены модели по её названию,

метод, возвращающий массив значений цен моделей,

метод добавления названия модели и её цены (путем создания нового массива Моделей), использовать метод Arrays.copyOf(),

метод удаления модели с заданным именем и её цены, использовать методы System.arraycopy, Arrays.copyOf(),

метод для получения размера массива Моделей.

Конструктор класса должен принимать в качестве параметров значение Марки автомобиля и размер массива Моделей.

Написать класс Мотоцикл, реализующий функциональность, сходную с классом Автомобиль, основанный на двусвязном циклическом списке с головой.

```
public class Мотоцикл {
    private class Модель{
        String название модели = null;
        double цена = Double.NaN;
        Модель prev = null;
        Модель next = null;
    }
    private Модель head = new Модель();
    {
     head.prev = head;
     head.next = head;
    }
    private int size = 0;
// далее код по заданию
}
```

Описать классы ошибок задания несуществующего имени модели NoSuchModelNameException (объявляемое), дублирования названия моделей DuplicateModelNameException (объявляемое), задание неверной цены модели ModelPriceOutOfBoundsException (необъявляемое).

Изменить методы классов так, чтобы они корректно обрабатывали ошибки и выбрасывали исключения.

Описать интерфейс Транспортное средство, имеющий методы, соответствующие общей функциональности двух созданных классов. Сделать так, чтобы оба класса реализовывали этот интерфейс.

Написать класс со статическими методами таким образом, чтобы он работал со ссылками типа интерфейса. В классе должен быть метод, возвращающий среднее арифметическое цен моделей для заданного Транспортного средства и методы, обеспечивающие вывод на экран всех моделей и всех цен на модели для заданного Транспортного средства.

Описать новый интерфейс TransportFactory, содержащий единственный метод createInstance(), создающий новое транспортное средство. В качестве параметров метод принимает значение Марки транспортного средства и размер массива Моделей.

В классе со статическими методами создать приватное статическое поле factory типа TransportFactory и соответствующий ему публичный метод setTransportFactory(), позволяющие, соответственно, хранить ссылку и устанавливать ссылку на текущую фабрику. По умолчанию поле должно ссылаться на объект некоторого класса AutoFactory (его также требуется описать), порождающего экземпляры класса Автомобиль.

В классе со статическими методами описать метод public static Transport createInstance(String name, int size), с помощью текущей фабрики создающий новый экземпляр транспортного средства.

Проверить работу фабричного метода в методе main().

1.3 Разработка приложения с использованием паттерна Prototype

Добавить в классы Автомобиль и Мотоцикл реализации методов Object clone(). Клонирование должно быть глубоким. Использовать super.clone().

Проверить работу методов clone() в методе main().

Вопросы:

1. Группа, описание, назначение, область применения, особенности реализации и структурная схема паттерна Abstract Factory.

2. Группа, описание, назначение, область применения, особенности реализации и структурная схема паттерна Builder.
3. Группа, описание, назначение, область применения, особенности реализации и структурная схема паттерна Factory Method.
4. Группа, описание, назначение, область применения, особенности реализации и структурная схема паттерна Prototype.
5. Группа, описание, назначение, область применения, особенности реализации и структурная схема паттерна Singleton.

Лабораторная работа № 2. Структурные паттерны.

2.1 Разработка приложения с использованием паттерна Adapter

Реализовать класс адаптера, метод которого принимает в качестве параметра массив строк и записывает их по очереди в выходной байтовый поток (OutputStream), который он «адаптирует». Продемонстрировать работу в методе main().

2.2 Разработка приложения с использованием паттерна Decorator

Добавить в класс со статическими методами реализацию метода Transport synchronizedTransport (Transport t), возвращающего ссылку на класс-обертку указанного транспортного средства, безопасный с точки зрения многопоточности. Для этого потребуется описать некий новый класс, реализующий интерфейс Транспортное средство.

2.3 Разработка приложения с использованием паттерна Proxy

Написать два приложения с использованием сокетов: серверное и клиентское. Серверное приложение должно прослушивать порт 5000 и выполнять операцию умножения двух вещественных чисел для подключающихся клиентов. На клиенте разработать прокси-класс, содержащий метод для перемножения двух вещественных чисел, но не осуществляющий собственно перемножение, а отправляющий эти два числа в серверную часть (порт 5000) и возвращающий ответ сервера в качестве результата. Проиллюстрировать работу клиента в методе main().

Вопросы:

1. Группа, описание, назначение, область применения, особенности реализации и структурная схема паттерна Adapter.
2. Группа, описание, назначение, область применения, особенности реализации и структурная схема паттерна Bridge.
3. Группа, описание, назначение, область применения, особенности реализации и структурная схема паттерна Composite.
4. Группа, описание, назначение, область применения, особенности реализации и структурная схема паттерна Decorator.
5. Группа, описание, назначение, область применения, особенности реализации и структурная схема паттерна Façade.
6. Группа, описание, назначение, область применения, особенности реализации и структурная схема паттерна Flyweight.
7. Группа, описание, назначение, область применения, особенности реализации и структурная схема паттерна Proxy.

Лабораторная работа № 3. Образцы поведения.

3.1 Разработка приложения с использованием паттерна Chain of Responsibility

Реализовать паттерн Chain of Responsibility, обеспечивающий вывод полей объекта типа Transport в текстовый файл в столбик или в одну строку. Для этого нужно разработать интерфейс Chain of Responsibility и два класса-наследника, каждый из которых осуществляет вывод соответствующим образом. В интерфейсе должен быть описан метод записи, в качестве параметра принимающий Транспортное средство, а также метод установки следующего в цепочке. Первая реализация этого интерфейса в цепочке выводит информацию в одну строку, если количество моделей меньше или равно 3. Вторая реализация в цепочке выводит информацию в столбик, если количество моделей больше 3.

Проверить работу паттерна в методе main().

3.2 Разработка приложения с использованием паттерна Command

Реализовать паттерн Command, обеспечивающий вывод полей объекта типа Автомобиль в текстовый файл в столбик или в одну строку. Для этого нужно разработать интерфейс Command и два класса-наследника, каждый из которых осуществляет печать соответствующим образом. В классе Автомобиль описать метод print(), которому в качестве параметра передавать поток, куда должна производиться печать. Метод должен обращаться к экземпляру класса, реализующего интерфейс команды (один из двух классов-наследников). Для задания команды добавить метод setPrintCommand() у класса Автомобиль.

Проверить работу паттерна в методе main().

3.3 Разработка приложения с использованием паттерна Iterator

Сделать класс Модель в классе Автомобиль доступным на уровне пакета и статическим. Реализовать в нём метод toString(), возвращающий название и цену модели.

Реализовать метод java.util.Iterator iterator() в классе Автомобиль. Для этого следует описать некий дополнительный внутренний класс с некими соответствующими методами (AutoIterator implements java.util.Iterator), экземпляр которого и будет возвращаться методом iterator().

Проверить работу итератора в методе main().

3.4 Разработка приложения с использованием паттерна Memento

Реализовать паттерн Memento, обеспечивающий сохранение текущего состояния объекта типа Автомобиль. Для этого нужно разработать соответствующий публичный статический внутренний класс, который будет сохранять состояние текущего объекта в сериализованном виде в массив байт (использовать класс ByteArrayOutputStream) и затем считывать сохраненное состояние. Соответствующие методы назвать setAuto() и getAuto(). В классе Автомобиль описать методы createMemento() и setMemento(), которые будут обращаться к соответствующим методам класса Memento. Проверить работу паттерна в методе main().

3.5 Разработка приложения с использованием паттерна Observer

Реализовать приложение, которое рисует на экране «рожицу». При клике мышкой в области глаза глаз должен закрываться (если был открыт) или открываться (если был закрыт). При клике мышкой в области носа его цвет должен измениться. При клике мышкой в области рта рожица должна улыбаться.

3.6 Разработка приложения с использованием паттерна Strategy

Реализовать паттерн Strategy, обеспечивающий сортировку массива Транспортных средств по значению среднего арифметического цен моделей Транспортных средств двумя разными способами. Для этого нужно описать интерфейс и два дочерних класса, каждый из которых будет реализовывать соответствующий алгоритм сортировки. Проверить работу паттерна в методе main().

3.7 Разработка приложения с использованием паттерна Template Method

Реализовать многопоточное приложение, которое будет анимировать прыгающий мяч, постоянно перемещая его по экрану и меняя направление движения, когда он встретит преграду в виде стены. Должны быть предусмотрены две кнопки: «Пуск» и «Закрыть». При щелчке по кнопке «Пуск» мяч выбрасывается из правого нижнего угла и начинает прыгать. При каждом нажатии на кнопку «Пуск» добавляется новый мяч. При щелчке по кнопке «Закрыть» приложение завершает свою работу. С помощью паттерна Template Method обеспечить возможность работы приложения не только с мячом, но и с другими фигурами (квадрат, звезда).

Проверить работу паттерна в методе main().

3.8 Разработка приложения с использованием паттерна Visitor

Реализовать паттерн Visitor, обеспечивающий печать полей объекта типа Транспортное средство в консоль в столбик или в одну строку. Для этого нужно описать интерфейс Visitor и его реализацию PrintVisitor с двумя вариантами метода visit(), с входным параметром типа Автомобиль (первый метод, выводит всё в одну строку) и Мотоцикл (второй метод, выводит модели и цены в столбик). В интерфейсе Транспортное средство добавить метод accept() с параметром типа Visitor. Каждый из потомков интерфейса

Транспортное средство внутри реализации этого метода будет вызывать соответствующий метод visit(). Проверить работу паттерна в методе main().

Вопросы:

1. Группа, описание, назначение, область применения, особенности реализации и структурная схема паттерна Chain of Responsibility.
2. Группа, описание, назначение, область применения, особенности реализации и структурная схема паттерна Command.
3. Группа, описание, назначение, область применения, особенности реализации и структурная схема паттерна Interpreter.
4. Группа, описание, назначение, область применения, особенности реализации и структурная схема паттерна Iterator.
5. Группа, описание, назначение, область применения, особенности реализации и структурная схема паттерна Mediator.
6. Группа, описание, назначение, область применения, особенности реализации и структурная схема паттерна Memento.
7. Группа, описание, назначение, область применения, особенности реализации и структурная схема паттерна Observer.
8. Группа, описание, назначение, область применения, особенности реализации и структурная схема паттерна State.
9. Группа, описание, назначение, область применения, особенности реализации и структурная схема паттерна Strategy.
10. Группа, описание, назначение, область применения, особенности реализации и структурная схема паттерна Template Method.
11. Группа, описание, назначение, область применения, особенности реализации и структурная схема паттерна Visitor.

Лабораторная работа № 4. Другие виды паттернов.

4.1 Разработка приложения с использованием паттерна MVC

Написать приложение, выводящее на экран график и таблицу значений некоторой функции y=f(x) (нелинейной). При изменении значений в таблице (добавлении, удалении, редактировании) график должен тоже изменяться. В таблице задаются x, значения y должны вычисляться автоматически при добавлении или редактировании x.

4.2 Разработка приложения с использованием паттерна DAO

Создать два файла, хранящих информацию об Автомобилях или Мотоциклах. Первый файл хранит информацию в текстовом виде (название марки, количество моделей, а затем список моделей и цен на них), второй – в виде сериализованного объекта. Реализовать паттерн DAO, обеспечивающий чтение данных из файлов указанного типа.

Вопросы:

1. Причины перепроектирования. Каркасы. Паттерны. Отличия каркасов от паттернов. Обзор паттернов проектирования.
2. Группа, описание, назначение, область применения, особенности реализации и структурная схема паттерна Model-View-Controller(MVC).
3. Группа, описание, назначение, область применения, особенности реализации и структурная схема паттерна Data Access Object (DAO).