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.