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 Student. Он должен содержать поле типа String, хранящее фамилию студента, методы для получения и модификации фамилии студента, поле marks, хранящее массив типа int, содержащий оценки по предметам, методы для получения и модификации значения элемента массива, поле subjects, хранящее массив типа String, содержащий названия предметов, методы для получения и модификации значения элемента массива, метод добавления предмета и оценки в соответствующие массивы с увеличением их длин (путем создания новых массивов, использовать метод Arrays.copyOf()), метод для получения размера массивов.

Конструктор класса должен принимать в качестве параметров значение фамилии и размер массивов.

Написать класс Schoolboy (или Schoolgirl), реализующий функциональность, сходную с классом Student. Оценки и предметы должны быть представлены полями внутреннего класса Register, класс Schoolboy(или Schoolgirl) хранит массив Register'ов.

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

Написать класс Pupils со статическими методами таким образом, чтобы он работал со ссылками типа интерфейса. В классе должны быть методы вывода на экран предметов и оценок, а также метод, возвращающий среднее арифметическое оценок ученика.

Описать новый интерфейс PupilFactory, содержащий единственный метод createInstance(), создающий нового ученика. В качестве параметров метод принимает фамилию ученика и размер массивов предметов и оценок (или размер массива Register'ов).

В классе Pupils создать приватное статическое поле factory типа PupilFactory и соответствующий ему публичный метод setPupilFactory(), позволяющие, соответственно, хранить ссылку и устанавливать ссылку на текущую фабрику. По умолчанию поле должно ссылаться на объект некоторого класса StudentFactory (его также требуется описать), порождающего экземпляры класса Student.

В классе Pupils описать метод public static Pupil createInstance(String name, int size), с помощью текущей фабрики создающий новый экземпляр ученика.

Проверить работу фабричного метода в методе main().

1.3 Разработка приложения с использованием паттерна Prototype

Добавить в классы Student и Schoolboy 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 Pupil synchronizedPupil (Pupil p), возвращающего ссылку на класс-обертку указанного ученика, безопасный с точки зрения многопоточности. Для этого потребуется описать некий новый класс, реализующий интерфейс Pupil.

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 Pupil 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 ученика, а также метод установки следующего в цепочке. Первая реализация этого интерфейса в цепочке выводит информацию в одну строку, если количество предметов меньше или равно 3. Вторая реализация в цепочке выводит информацию в столбик, если количество предметов больше 3.

Проверить работу паттерна в методе main().

3.2 Разработка приложения с использованием паттерна Command

Реализовать паттерн Command, обеспечивающий вывод полей объекта типа Student 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 студент 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 Student.

Проверить работу паттерна в методе main().

3.3 Разработка приложения с использованием паттерна Iterator

Сделать класс Register в классе Schoolboy (или Schoolgirl) доступным на уровне пакета и статическим. Реализовать в нём метод toString(), возвращающий предмет и оценку.

Реализовать метод java.util.Iterator iterator() в классе Schoolboy (или Schoolgirl). Для этого следует описать некий дополнительный внутренний класс с некими соответствующими методами (SchoolboyIterator 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 Student. 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 setStudent() и getStudent(). В классе Student 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 учеников по значению среднего арифметического оценок ученика 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 Visitor

Реализовать паттерн Visitor, обеспечивающий печать полей объекта типа Pupil 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 Student (первый метод, выводит всё в одну строку) и Schoolboy (второй метод, выводит предметы с оценками в столбик). В интерфейсе Pupil добавить метод accept() с параметром типа Visitor. Каждый из потомков интерфейса Pupil 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 о студентах или школьниках. Первый файл хранит информацию в текстовом виде (фамилию ученика, количество предметов (оценок), а затем список предметов и оценок), 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).