1. Abstract Factory.

Реализовать лабиринт, который состоит и комнат, каждая из комнат состоит из стен. Предусмотреть несколько типов стен (стены, которые при взаимодействии меняют цвет, и стены, которые издают звуковой сигнал). Для выполнения задания использовать паттерн Abstract Factory.

1. Compositor.

Реализовать программу, в которой на экране двигаются различные объекты и при столкновении агрегируются (продолжают движение совместно). Для выполнения задания использовать паттерн Compositor.

1. Prototype.

Создать объект гистограмма, который обладет 4 конструкторами:

* Создание.
* конструктор по копированию.
* конструктор, загружаюжий объект из файла.
* Конструктор, загружающий объект из БД.

Соответственно, реализовать методы сохранения гистограммы в файл и БД, а также предусмотреть раличное кол-во столбцов. Демонстрация: при нажатии на гистограмму на экране она копируется конструктором по копированию в другой объект, который визуализируется на свободном пространстве.

Доп. Задание: реализовать возможность изменения размеров столбцов.

1. Prototype.

Реализовать в нулевом приближении музыкальный редактор:

Есть нотный стан, на который в произвольном месте можно размещать ноты. С помощью паттерна Prototype реализовать такую возможность.

1. Prototype.

Реализовать лабиринт с помощью диспетчера прототипов. Объекты на форме создаются следующим образом: при создании нового объекта, просматривается класс-диспетчер, если есть объект такого типа, то он копируется конструктором по копированию в новый объект.

1. Adapter.

Взять чужой код лабиринта, и с помощью паттерна использовать его в своей программе таким образом, чтобы представление модели осталось неизменным (все вызовы методов из представления имели такую же сигнатуру, как и в исходном лабиринте из лаб. 1). Доступ к объектам модели выполняется через Adapter.

1. Bridge.

Для лабиринта предусмотреть возможность запуска на различных платформах. Как вариант, реализовать возможность визуализации на форме и в консоли.

1. Decorator.

Реализовать в лабиринте 3 декоратора:

* Обводит стенку красным цветом.
* Делает границу штрихованной.
* Добавляет функцию поворота при щелчке мыши

Декоратор создается при помощи абстрактной фабрики.

1. Flyweight.

Переписать лабиринт как декартово произведение стенок на их свойства (выделить уникальные свойства, присущие всему семейству объектов, дополнительные свойства реализуются путем наследования).

1. Proxy.

В качестве стенок для лабиринта использовать загрузку объекта из файла. Все стенки грузятся по частям – до того момента, пока стенка не загрузится, отображается ее Proxy, а после завершения операции заменяется реальным объектом.

1. Chain of Responsibility.

Снабдить все стенки в лабиринте heper’ом – стенки должны рассказывать о своих свойствах. Две различные стенки должны обладать двумя открытыми helper’фми, в которых выводится соответствующая информация.

1. Command.

Предыдущее задание реализовать с помощью паттерна Command.

1. Iterator.

Снабдить лабиринт 2 типами итераторов:

* Внутренний. Проходит все стенки и красит их в зеленый цвет.
* Внешний. Проходит все окаймленные стенки и красит их окаймление в желтый, а сами стенки в синий.

1. Mediator.

Реализовать:

* При введении текстовой строки активирует 2 кнопки.
* При введении текстовой строки одна кнопка деактивируется, а во второй вызов диалога на открытие файла происходит с участием пути и имени файла, если таковы найдутся в строке ввода.

Реализовать смену функциональности путем смены одного медиатора другим.

1. Strategy.

* Реализовать три метода покраски стен:
* Заливка
* С левого края на правый
* Одновременно к середине.

Применить паттерн Strategy.