Министерство Образования Республики Молдова

Технический Университет Молдовы

Департамент программной инженерии и автоматики

**Отчет**

Лабораторная работа №5

Тема: «Abstract factory»

Выполнил: TI-197-Герца Николай

Проверил: Булдумаг Олег

Кишинев-2021

**Суть паттерна** **Abstract factory:**

• Абстрактная фабрика - шаблон, который можно использовать для определения взаимосвязанных экземпляров или объектов, не специфицируя их конкретных классов.

• Шаблон реализуется созданием абстрактного класса Factory, который представляет собой интерфейс для создания компонентов системы

**Что будет происходить:**

• Абстрактная фабрика reputation в зависимости от входных данных будет возвращать свою собственную фабрику: badReputation() и goodReputation(). И сами эти фабрики вызывают классы-конструкторы.

• Абстрактной фабрике придет запрос, на создание определенной репутации водителю. В свою очередь, абстрактная фабрика отправит запрос на создание определенной репутации (обычный объект) на соответствующую фабрику, которая и создаст экземпляр нужной нам репутации.

**Диаграмма UML:**

**•**



**Реализация:**

**•**

//Абстрактная фабрика

function reputation(score) {

  return score === "good" ? goodReputation : badReputation;

}

//Обычные фабрики

function badReputation() {

  return new BadRate();

}

function goodReputation() {

  return new GoodRate();

}

class GoodRate {

  info() {

    return "Данный водитель имеет хорошую репутацию";

  }

}

class BadRate {

  info() {

    return "Данный водитель имеет плохную репутацию";

  }

}

//Инициализируем абстрактную фабрику

const rep = reputation("good");

//

const repOfDriver = new rep();

console.log(repOfDriver.info());

**Применение:**

• Когда бизнес-логика программы должна работать с разными видами связанных друг с другом продуктов, не завися от конкретных классов продуктов.

• Когда в программе уже используется Фабричный метод, но очередные изменения предполагают введение новых типов продуктов.

**Преимущества:**

• Гарантирует сочетаемость создаваемых продуктов.

• Избавляет клиентский код от привязки к конкретным классам продуктов.

• Выделяет код производства продуктов в одно место, упрощая поддержку кода.

• Упрощает добавление новых продуктов в программу.

• Реализует принцип открытости/закрытости.

**Недостатки:**

• Усложняет код программы из-за введения множества дополнительных классов.

• Требует наличия всех типов продуктов в каждой вариации.