ЛАБОРАТОРНА РОБОТА №5

ПОЛОНЕВИЧ Д.В.

СТВОРЕННЯ ПАТЕРНУ ПРОЕКТУВАННЯ

Мета роботи: створити патерн проектування.

ХІД РОБОТИ

Було обрано паттерн «Фабрика». Цей паттерн дозволяє спростити створення об’єктів, схожих за призначенням, і спростити роботу з ними, фактично об’єднавши частину цієї роботи разом, там де це можливо. Написана мною невелика програма створює за допомогою фабрики двох товарів. Створивши таку фабрику, ми отримуємо можливість працювати з сервісами, і навіть всередині них уніфікованим, однаковим чином. Почнемо зі створення наших конструкторів. Ці функції відповідатимуть за повернення нових об’єктів певного типу під час виклику.

В папці factory створимо Poster.js файл.

```
const Poster = function({ name }) {
  this.name = name || "";
};
module.exports = Poster;
```

У цьому файлі ми створюємо функцію конструктора Постера. Він приймає об’єкт як параметр із атрибутами для створення екземпляра об’єкта з різними специфікаціями, які ми хочемо захопити — у цьому випадку ім’я. Після цього ми експортуємо функцію Laptop constructor з модуля. Створимо ще один файл під назвою shirt.js Ми зробимо те саме, але зі специфікаціями, більш відповідними футболці.

```
const Shirt = function({ name, size }) {
  this.name = name || "";
  this.size = size || "";
};
module.exports = Shirt;
```

					ДУ «Житомирська політехніка».22.121.06.000 – Лр5		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Полоневич Д.В.			Звіт з лабораторної роботи №5	Літ.	Арк. 1 / Аркушів 2
Перевір.		Сугоняк І. І.					
Керівник						ФІКТ Гр. ІПЗ-19-1	
Н. контр.							
Зав. каф.							

Тепер, коли у нас є наші конструктори, створимо функцію фабрики, яка відкриватиме API для створення нових екземплярів цих елементів. Додайте новий файл під назвою movieFactory.js

```
const Shirt = require("./shirt");
const Poster = require("./poster");
const movie = { Shirt, Poster };
module.exports = {
    createGadget(type, attributes) {
        const movieType = movie[type];
        return new movieType(attributes);
    }
};
```

Тут ми починаємо з імпортування конструкторів для створення об'єктів «Shirt» і «Poster». Потім ми створюємо об'єкт «movie», використовуючи імена конструкторів як ключі. Це дає нам змогу отримати доступ до потрібного типу конструктора за допомогою movie [тип] — де в цьому прикладі типом буде «Poster» або «Shirt». Нарешті, ми експортуємо об'єкт із цього модуля за допомогою методу createMovie. Цей метод приймає тип «movie» як перший параметр і викликає вказаний тип конструктора, передаючи йому атрибути.

На цьому етапі ми можемо створити файл, який використовуватиме наш заводський шаблон API.

Index. js

```
const movieFactory = require("./movieFactory");
const myPoster = movieFactory.createMovie("Poster", {
    name: "Spider Man"
});
const myShirt = movieFactory.createMovie("Shirt", {
    name: "Spider Man",
    size: "56"
});
console.log(myPoster);
console.log(myShirt);
```

Перше, що можна помітити, це те, що в цьому файлі нам не потрібні безпосередньо конструктори для «Poster» і «Shirt». Все, що нам потрібно, це модуль movieFactory (з його методом createMovie). Використовуючи цей метод, ми створюємо два екземпляри Poster та Shirt відповідно та виходимо з консолі.