**Тема:** Порождающие паттерны: «Фабрика», «Абстрактная фабрика», «Фабричный метод.

**«Абстрактная фабрика»**

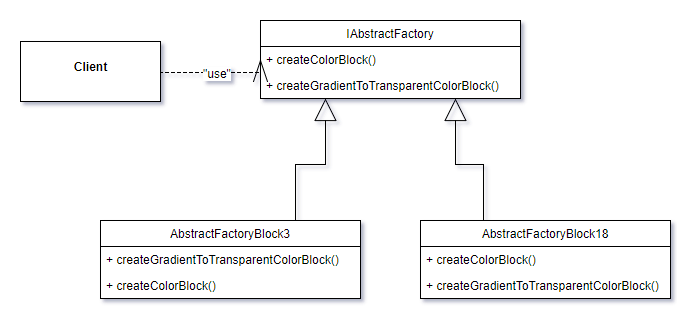
Описание:

Фабрика создает объекты, состоящие из нескольких составных элементов или обладающие несколькими различными свойствами. Клиенты создают комплексные объекты или комплексные свойства объектов, пользуясь исключительно интерфейсом AbstractFactory. Клиентам ничего не известно о классах, реализующих объекты конкретного типа. Другими словами, клиенты должны лишь придерживаться интерфейса, определенного абстрактным, а не конкретным классом.

Пример реализации:

Создадим интерфейс абстрактной фабрики по созданию ColorBlock’ов; создадим конкретные реализации, создающие ColorBlock’и шириной 3 и 18 пикселов.

Диаграмма классов:



Исходный код:

IAbstractFactory.js

**class** IAbstractFactoryBlock {  
 createColorBlock(){};  
 createGradientToTransparentBlock(){};  
}

AbstractFactoryBlock3.js

**class** AbstractFactoryBlock3 **extends** IAbstractFactoryBlock {  
 createColorBlock(colorRGB, colorH) {  
 **return** <ColorBlock color={[colorRGB,colorH]} width={3}/>  
 }  
 createGradientToTransparentBlock(colorRGB, colorH) {  
 **return** <TransparentBlock color={[colorRGB,colorH]} width={3}/>;}}

AbstractFactoryBlock18.js

**class** AbstractFactoryBlock18 **extends** IAbstractFactoryBlock {  
 createColorBlock(colorRGB, colorH) {  
 **return** <ColorBlock color={[colorRGB,colorH]} width={18}/>;  
 }  
  
 createGradientToTransparentBlock(colorRGB, colorH) {  
 **return** <TransparentBlock color={[colorRGB,colorH]} width={18}/>;  
 }  
}

**«Фабричный метод»**

Описание:

В данном паттерне строится параллельная иерархия классов, содержащих перегружаемый фабричный метод для создания соответствующих экземпляров классов-продуктов. Создание делегируется подклассам, при этом скрыта реализацию для достижения большей гибкости или возможности расширения функциональности.

Пример реализации:

В зависимости от переданных параметров, могут быть созданы различные наборы ColorBlock’ов.

Диаграмма классов:

Исходный код:

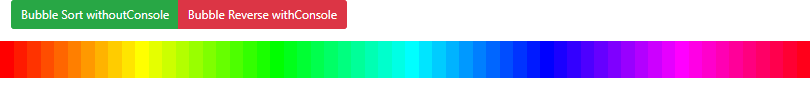
FactoryMethodScene.js

**class** FactoryMethodScene {  
 getColorBlocks(colorsNumber = 360, type = 'color') {  
 **function** getColorSet(colorsNumber = 360) {...}  
  
 **let** colors = getColorSet(colorsNumber),  
 colorBlocks = [],  
 fact18 = **new** AbstractFactoryBlock18(),  
 fact3 = **new** AbstractFactoryBlock3();  
 **let** colorsNumberAndType = colorsNumber + type;  
 **switch** (colorsNumberAndType) {  
 **case** '60color':  
 **for** (**let** i = 0; i < 60; i++) {  
 colorBlocks.push(fact18.createColorBlock(colors[i][0], colors[i][1]));  
 }  
 **break**;  
 **case** '60transparent':  
 **for** (**let** i = 0; i < 60; i++) {  
 colorBlocks.push(fact18.createGradientToTransparentBlock(colors[i][0], colors[i][1]));  
 }  
 **break**;  
 **case** '360transparent':  
 **for** (**let** i = 0; i < 360; i++) {  
 colorBlocks.push(fact3.createColorBlock(colors[i][0], colors[i][1]));  
 }  
 **break**;  
 **case** '360color':  
 **default**:  
 **for** (**let** i = 0; i < 360; i++) {  
 colorBlocks.push(fact3.createColorBlock(colors[i][0], colors[i][1]));  
 }  
 **break**;  
 }  
 **return** colorBlocks;  
 };  
}

FactoryMethodSceneRTL.js

**class** FactoryMethodSceneRTL **extends** FactoryMethodScene {  
 getColorBlocks(colorsNumber = 360, type = 'color') {  
 // Генерируем цвета в порядке справа на лево  
 **function** getColorSet(colorsNumber = 360) {...}  
  
 **let** colors = **this**.getColorSet(colorsNumber),  
 colorBlocks = [],  
 fact18 = **new** AbstractFactoryBlock18(),  
 fact3 = **new** AbstractFactoryBlock3();  
 **let** colorsNumberAndType = colorsNumber + type;  
 **switch** (colorsNumberAndType) {  
 **case** '60color':  
 **for** (**let** i = 0; i < 60; i++) {  
 colorBlocks.push(fact18.createColorBlock(colors[i][0], colors[i][1]));  
 }  
 **break**;  
 **case** '60transparent':  
 **for** (**let** i = 0; i < 60; i++) {  
 colorBlocks.push(fact18.createGradientToTransparentBlock(colors[i][0], colors[i][1]));  
 }  
 **break**;  
 **case** '360transparent':  
 **for** (**let** i = 0; i < 360; i++) {  
 colorBlocks.push(fact3.createColorBlock(colors[i][0], colors[i][1]));  
 }  
 **break**;  
 **case** '360color':  
 **default**:  
 **for** (**let** i = 0; i < 360; i++) {  
 colorBlocks.push(fact3.createColorBlock(colors[i][0], colors[i][1]));  
 }  
 **break**;  
 }  
 **return** colorBlocks;  
 };  
}

Снимки экрана:



RTL:

