**Паттерн Fabric Method**

Паттерн Фабричный метод был применён при реализации приложения для отправки уведомлений. Логика приложения состоит в следующем: у вас имеется два класса: *генератор* *уведомлений* (**NotificationCreator**) и само *уведомление* (**Notification**). Генератор уведомлений создаёт, сохраняя информацию о тексте уведомления. Уведомление же имеет только один метод – отправление этого уведомления.

Приложение предоставляет различные типы уведомлений (если конкретно, в описанном примере – это Push (которые на самом деле Toast), Telegram и Email). Без паттерна было реализовано просто 3 пары не связанных между собой классов для каждого из типов уведомления. При этом генераторы предоставляли один и тот же функционал, но работали с разными типами уведомлений.

После применения паттерна были добавлены два абстрактных класса, предоставляющих общие поля и методы генераторов уведомлений (ссылка на уведомление *notification* и метод, создающий уведомление *GenerateNotification(txt)*, создающий уведомление по тексту и записывающее ссылку на него в соответствующее поле) и уведомлений (строка с текстом *text*, метод отправки уведомления *SendNotification()*). Остальные классы реализуют эти два абстрактных класса для конкретных типов уведомлений. Теперь в генераторах, кроме абстрактного метода *GenerateNotification(txt)* и общего поля *notification* имеются поля, необходимые для генерации уведомлений и конструкторы каждого отдельного генератора для заполнения этих полей.

Отдельно отмечу, что в данной версии приложения у генератора имеется ссылка только на одно уведомление. При первом создании в неё записывается именно первое уведомление, при последующих просто заменяется его текст, так как в демонстрационной версии используется только один вариант отправителя и получателя. В дальнейшем можно создавать коллекцию уведомлений для более гибкой работы с отправителями и получателями, что уже предусмотрено конструкторами генераторов.

Паттерн был применён для того, чтобы обобщить функциональность классов и гарантировать то, что пользовательские классы будут реализовывать необходимые методы. Более того, теперь во «внешних функциях», которые вызывают необходимые функции нет необходимости писать каждый отдельный тип уведомления, а просто ставить *Notification*, а компилятор сам вызовет необходимый метод в зависимости от того, какой генератор был запущен ранее.