http://developer.alexanderklimov.ru/android/theory/ble.php – Дата доступа: 10.03.2020.

3. Подключаемся к Intel Edison через Android с Bluetooth LE (BLE) [Электронный ресурс] // Блог компании Intel, – 2015. –URL: https://habr.com/ru/company/intel/blog/252919/ – Дата доступа: 10.03.2020.

А. Ю. Никонович (ГГУ имени Ф. Скорины, Гомель)
Науч. рук. **О. М. Дерюжкова**, канд. физ.-мат. наук, доцент

ПАТТЕРНЫ ПРОЕКТИРОВАНИЯ

Паттерны проектирования – типичные решения часто возникающих проблем в разработке программного обеспечения. Многие разработчики и авторы книг иногда именуют паттерны как шаблоны проектирования. Они похожи на готовые чертежи, которые можно настроить для решения повторяющейся проблемы проектирования разрабатываемого кода. При этом нельзя просто найти шаблон и скопировать его в свою программу, что возможно с помощью стандартных функций или библиотек. Шаблон – это не конкретная часть кода, а общая концепция для решения конкретной проблемы. В данном случае можно следить за деталями шаблона и реализовать решение, которое соответствует реалиям собственной программы. Шаблоны часто путают с алгоритмами, потому что обе концепции описывают типичные решения некоторых известных проблем. В то время как алгоритм всегда определяет четкий набор действий, которые могут достичь определенной цели, шаблон является более высокоуровневым описанием решения. Код одного и того же шаблона, применяемый к двум разным программам, может отличаться. Аналогия с алгоритмом – рецепт приготовления: у обоих есть четкие шаги для достижения цели. С другой стороны, шаблон больше похож на план: можно видеть, каков результат и его особенности, но точный порядок реализации зависит от разработчика.

Шаблоны являются типичными решениями общих проблем в объектно-ориентированном дизайне. Когда решение повторяется снова и снова в различных проектах, кто-то, в конце концов, ставит ему имя и подробно описывает решение.

Концепция шаблонов была впервые представлена Кристофером Александром в книге «Язык шаблонов: города, здания, строитель-

ство» [1], опубликованной в 1977 году. В ней предложен «язык» для проектирования городской среды. Единицами этого языка являются шаблоны. Они могут описывать, какие высокие окна должны быть, сколько уровней должно иметь здание, насколько большими должны быть зеленые зоны в окрестностях и так далее.

Идея была подхвачена четырьмя авторами: Эрихом Гаммой, Джоном Влиссидесом, Ральфом Джонсоном и Ричардом Хелмом. В 1994 году они опубликовали шаблоны проектирования: элементы многоразового объектно-ориентированного программного обеспечения, в которых применили концепцию шаблонов проектирования к программированию. Книга содержит 23 шаблона, решающих различные задачи объектно-ориентированного дизайна, поэтому очень быстро стала бестселлером [2].

С тех пор были разработаны десятки других объектно-ориентированных моделей. «Шаблонный подход» стал очень популярным во всех областях программирования, поэтому в настоящее время существует множество разнообразных шаблонов вне объектно-ориентированного проектирования.

Несмотря на огромную пользу паттернов в ООП (объектно-ориентированном программировании) у них есть и слабые стороны:

- Костыли для слабого языка программирования.

Обычно необходимость в шаблонах возникает, если выбирают язык программирования или технологию, в которой отсутствует определенный уровень абстракции. В этом случае шаблоны становятся костылём, который дает языку столь необходимые сверхспособности.

Например, некоторые шаблоны могут быть реализованы с помощью простой анонимной (лямбда) функции в большинстве современных языков программирования.

- Неэффективные решения.

Шаблоны пытаются систематизировать подходы, которые уже широко используются. Это объединение рассматривается многими как догма, и они реализуют шаблоны «точно», не адаптируя их к контексту своего проекта.

- Неоправданное использование.

Это проблема преследует многих новичков, которые только что ознакомились с шаблонами. Узнав о шаблонах, они стараются применять их повсюду, даже в ситуациях, когда более простой код вполне подойдет.

Допустим при создании приложения по управлению перевозками разработчики ориентировались на перевозку товаров автомобилями, но в один прекрасный день появилась необходимость добавления функционала морских перевозок. Большая часть программы жёстко привязана к классу автомобильных перевозок и для добавления нового функционала требуется переделать существенную часть программы. А вдруг после этого потребуется добавить ещё какой-либо вид транспорта? Тут на помощь и приходят паттерны проектирования, а именно шаблон фабричного метода, который обеспечивает интерфейс для создания объектов в суперклассе, но позволяет подклассам изменять тип создаваемых объектов. Структура данного паттерна такова [3]:

Класс Creator объявляет фабричный метод, который возвращает новые объекты продукта. Важно, чтобы возвращаемый тип этого метода соответствовал интерфейсу продукта.

Можно объявить метод фабрики как абстрактный, чтобы заставить все подклассы реализовывать свои собственные версии метода. В качестве альтернативы метод базовой фабрики может возвращать некоторый тип продукта по умолчанию.

Несмотря на название, создание продукта не является основной обязанностью создателя. Обычно класс создателя уже имеет некоторую основную бизнес-логику, связанную с продуктами. Фабричный метод помогает отделить эту логику от конкретных классов продуктов.

Можно привести следующую аналогию: крупная компания по разработке программного обеспечения может иметь учебный отдел для программистов. Тем не менее, основной функцией компании в целом по-прежнему является написание кода, а не обучение программистов.

Литература

1. Александер, К. Язык шаблонов: города, здания, строительство / К. Александер, С. Исикава, М. Силверстайн. – Из-во: Студии Артемия Лебедева, 2014. – 1096 с.

2. Гамма, Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес. – Из-во: Питер, 2016. – 366 с.

3. Швец, А. Погружение в паттерны проектирования / А. Швец. – Refactoring.Guru, 2019. – 399 с.