**ЛАБОРАТОРНАЯ РАБОТА №2**

по дисциплине «Модульное тестирование»

# «Проектирование модульного тестирования»

Автор: студент гр. 10701118 Воробей И.А.

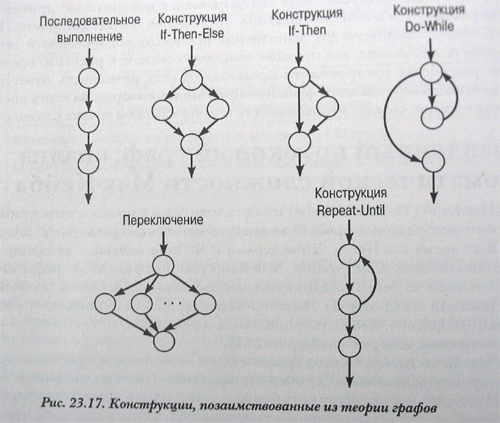
Проверил: доцент, к.т.н. Попова Ю.Б.

**Цель работы**: закрепление практических навыков по проектированию модульного тестирования.

**Теоретические основы:**

Как правило, тестирование структуры программного модуля происхо-дит с помощью методов графического отображения модуля, например, диаграммы Чейпина или Несси Шнайдермана, ориентированнные графы Мак-Кейба. Подобно Чейпину, Мак-Кейб применяет следующие конструкции: if-then-else, if then, do-while, case и repeat-until. Эти конструкции изображены на рисунке ниже:

.



Метрический показатель сложности или цикломатическое число ***G*** потокового графа определяется по формуле:

***G=R-V+2,***

где ***R*** – количество ребер графа;

***V*** – количество вершин графа.

Чтобы вычислить этот коэффициент вручную для большой программы, потребуется немало усилий. К счастью, существуют автоматические средст-ва, которые вычисляют метрический показатель сложности Мак-Кейба путем анализа существующего исходного кода.

Метрический показатель сложности не только может пригодиться при установлении проблемных областей, но также может использоваться при создании контрольных примеров. Как только потоковый граф создан, очевидными становятся пути, ведущие через программу, которые предоставляют информацию, необходимую для их выполнения в тесте.

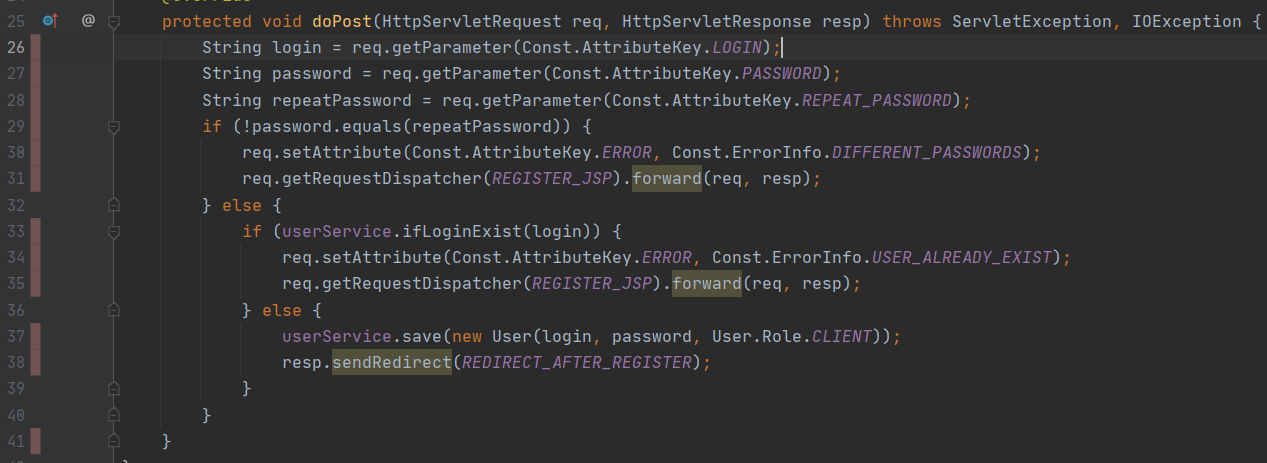
**Задание:**

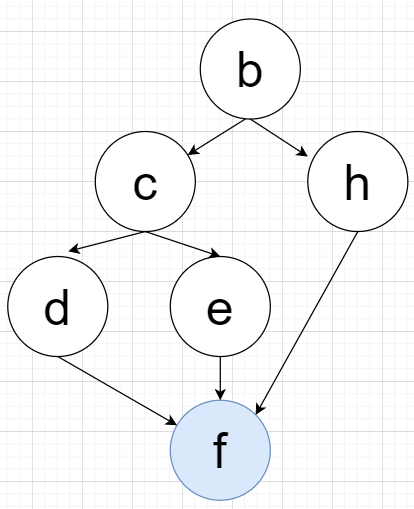
1. Провести обзор разработанного программного кода, привести критерии обзора. Сделать отчет о найденных в процессе обзора ошибках и занести их в таблицу:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Название модуля** |  |  | **Описание ошибки** | **Важность**  **ошибки**  **(низкая, средняя, высокая)** | **Ошибка исправлена**  **Да/Нет** |
| 1 | RegisterPageServlet.doPost(…) |  |  | Не происходит валидация по длине login & password. | средняя | Нет |
| 2 | editProduct.jsp |  |  | Отсутствует класс-модификатор в блоке вывода ошибки. | низкая | нет |
| 3 | ProductListPageServlet.doGet(…) |  |  | Если не найден ни один продукт, не происходит добавления сообщения “There are no products conforming your query” в ответ сервера. | низкая | нет |
|  |  |  |  |  |  |  |
| 4 | RegisterPageServlet.doPost(…) |  |  | Не происходит валидация по составу введенного логина. Метод должен пропускать только латинские буквы, цифры и знак подчёркивания | высокая | нет |
|  |  |  |  |  |  |  |

1. Для каждого модуля (как минимум, 5 сложно структурированных) построить ориентированный граф Мак-Кейба.

Case 1.

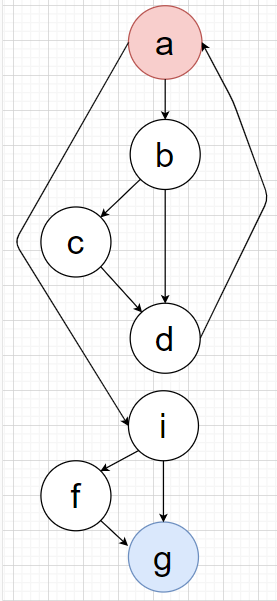
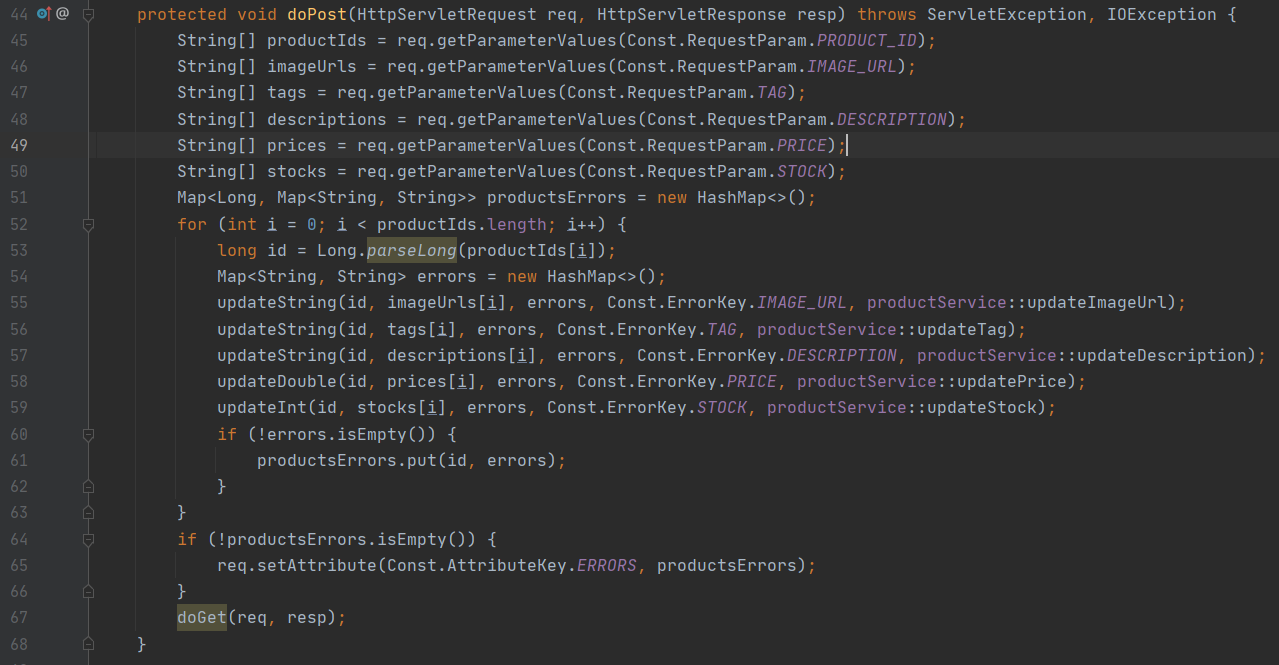




G = 7 – 6 + 2 = 3

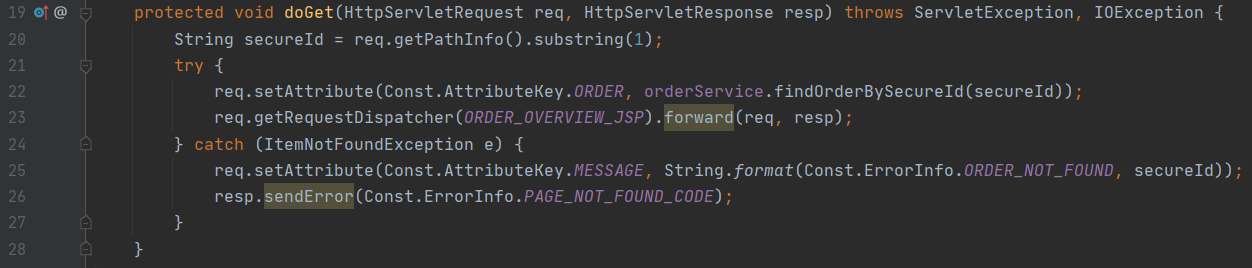
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **G** | **№ прохода** | **Описание прохода** | **Контрольные примеры, позволяющие реализовать описанную ситуацию** | **Тест пройден**  **Да/Нет** |
| **G=3** | **1**  **2**  **3** | **b-h-f**  **b-c-d-f**  **b-c-e-f** | !password.equals(repeatPassword)  password.equals(repeatPassword) && userService.ifLoginExist(login)  password.equals(repeatPassword) && !userService.ifLoginExist(login) | **да** |

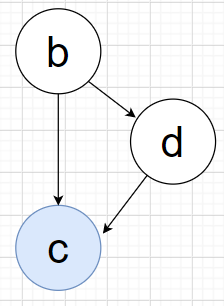
Case 2.



G = 9 – 7 + 2 = 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **G** | **№ прохода** | **Описание прохода** | **Контрольные примеры, позволяющие реализовать описанную ситуацию** | **Тест пройден**  **Да/Нет** |
| **G=4** | **1**  **2**  **3**  **4** | **A-b-d-a-I-g**  **A-b-d-a-I-f-g**  **A-b-c-d-a-i-g**  **A-b-c-d-a-i-f-g** | errors.isEmpty() && productsErrors.isEmpty()  errors.isEmpty() && !productsErrors.isEmpty()  !errors.isEmpty() && productsErrors.isEmpty()    !errors.isEmpty() && !productsErrors.isEmpty() | **да** |

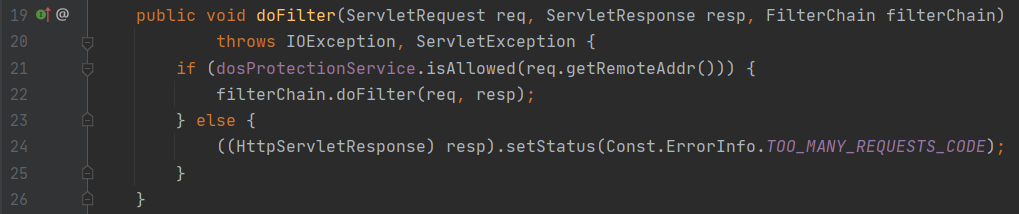
Case 3.

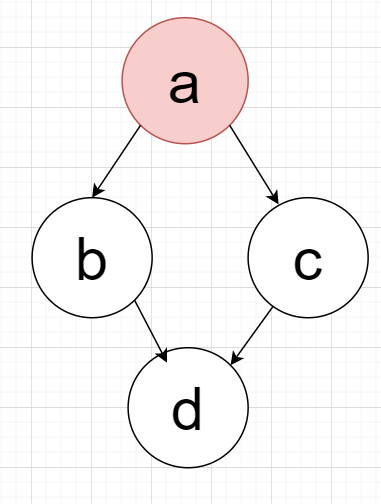


G= 3 – 3 + 2 = 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **G** | **№ прохода** | **Описание прохода** | **Контрольные примеры, позволяющие реализовать описанную ситуацию** | **Тест пройден**  **Да/Нет** |
| **G=2** | **1**  **2** | **B-c**  **B-d-c** | No exceptions  ItemNotFoundException has been thrown | **да** |

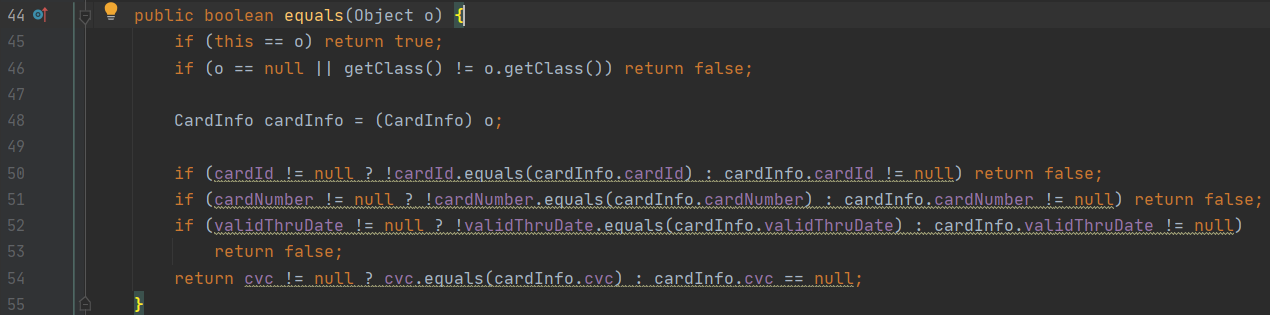
Case 4.

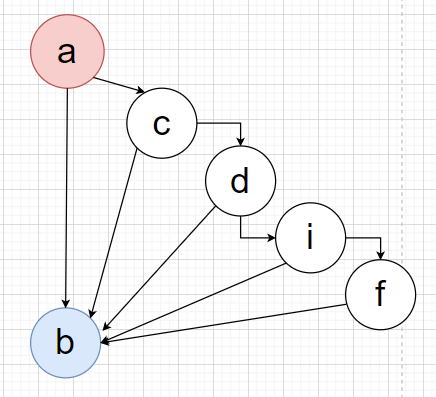




G = 4 – 4 + 2 = 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **G** | **№ прохода** | **Описание прохода** | **Контрольные примеры, позволяющие реализовать описанную ситуацию** | **Тест пройден**  **Да/Нет** |
| **G=2** | **1**  **2** | **A-b-d**  **A-c-d** | dosProtectionService.isAllowed(req.getRemoteAddr()) = true  dosProtectionService.isAllowed(req.getRemoteAddr()) = false | **да** |

Case 5.



G = 9 – 6 + 2 = 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **G** | **№ прохода** | **Описание прохода** | **Контрольные примеры, позволяющие реализовать описанную ситуацию** | **Тест пройден**  **Да/Нет** |
| **G=5** | **1**  **2**  **3**  **4**  **5** | **A-b**  **A-c-b**  **A-c-d-b**  **A-c-d-i-b**  **A-c-d-i-f-b** | this == o  previous && o == null || getClass() != o.getClass()  previous && cardId != null ? !cardId.equals(cardInfo.cardId) : cardInfo.cardId != null  previous && cardNumber != null ? !cardNumber.equals(cardInfo.cardNumber) : cardInfo.cardNumber != null    previous && validThruDate != null ? !validThruDate.equals(cardInfo.validThruDate) : cardInfo.validThruDate != null | **да** |

Выводы.

Научился определять цикломатичискую сложность кода. Расчитал цикломатичискую сложность и построил графы МакКейба для пяти методов из своего проекта. Формула G=R-V+2 не работает для сложных конструкций с try catch.