МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по учебной практике

Тема: UI-тестирование сайта Citilink

Студенты гр. 3384, 3381 Баяндин Д. С.
Рудаков А. Л.
Яковлев Д. С.

Руководитель Шевелева А. М.

Санкт-Петербург
2025

ЗАДАНИЕ

НА УЧЕБНУЮ ПРАКТИКУ

Студенты Баяндин Д. С., Рудаков А. Л., Яковлев Д. С.

Группы 3384, 3381

Тема практики: UI-тестирование сайта Citilink

Задание на практику:

Написать архитектуру для реализации 10 различных UI-тестов функционала сайта Citilink..

Сроки прохождения практики: 25.06.2025 – 08.07.2025

Дата сдачи отчета: 04.07.2025

Дата защиты отчета: 05.07.2025

Студенты Баяндин Д. С.
Рудаков А. Л.
Яковлев Д. С.

Руководитель Шевелева А. М.

АННОТАЦИЯ

В рамках практической работы необходимо разработать архитектуру, реализующую 10 различных UI-тестов сайта Ситилинк. Для этого необходимо реализовать:

1. Базовые классы для UI-элементов и страниц, на базе которых будут построены конкретные элементы страниц и страницы.
2. Наследуя от базового класса элемента, классы, реализующие функционал элементов интерфейса.
3. Наследуя от базового класса страницы, классы, представляющие страницы сайта.
4. Классы, проверяющие конкретный заданный функционал сайта.

SUMMARY

As part of the practical work, it is necessary to develop an architecture that implements 10 different UI tests of the Citylink website. To do this, you need to implement:

1. Base classes for UI elements and pages, on the basis of which specific page elements and pages will be built.
2. Inheriting from the base class of an element, classes that implement the functionality of interface elements.
3. Inheriting from the base class of the page, the classes representing the pages of the site.
4. Classes that check the specific specified functionality of the site.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В рамках практической работы необходимо спроектировать архитектуру автоматизированного UI-тестирования[3] для сайта Ситилинк и реализовать 10 различных тестов. В основе проекта должны лежать базовые классы, описывающие элементы пользовательского интерфейса. На их основе создаются конкретные реализации: кнопки, чекбоксы, поля ввода, строка поиска и различные карточки товаров и магазинов. Страницы сайта — такие как корзина, избранное, профиль, каталог, сравнение, настройки, магазины и конфигуратор. Далее разрабатываются тестовые классы, которые используют созданные элементы и страницы для проверки конкретных сценариев и функционала сайта.

1. РЕАЛИЗУЕМЫЕ ТЕСТЫ

1. Добавление товара в корзину: поиск товара по названию и добавление через кнопку “добавить в корзину” на странице поиска. Товар должен появиться в корзине.
2. Добавление товара в сравнение: поиск товара через каталог по названию и добавление через кнопку “добавить в сравнение”. Товар должен появиться в сравнении.
3. Добавление товара в избранное: поиск первого товара на странице акций и новинок и добавление в избранное через кнопку “добавить в избранное” на странице лучшие новинки. Товар должен появиться в избранном.
4. Смена имени пользователя: переход к странице настроек профиля и изменение Имени и Фамилии с подтверждением изменений. Имя и Фамилия должны поменяться.
5. Использование фильтров цены: выполнение поиска по запросу и выставление минимальной цены. Первый товар в поиске должен иметь цену не менее указанной минимальной.
6. Добавить в конфигуратор: создание новой конфигурации, добавление случайного процессора в конфигурацию. Случайно выбранный процессор должен появиться в конфигурации.
7. Удаление из корзины: на странице корзины удаление товара по названию через кнопку “удалить из корзины” или через чекбокс и кнопку “удалить выбранные”. Товара не должно быть в корзине.
8. Удаление из избранного: на странице избранное удаление товара по названию через кнопку “удалить из избранного” или через кнопку “очистить список”. Товара не должно быть в избранном.
9. Удаление из сравнения: на странице сравнения удаление товара по названию через кнопку “удалить из сравнения” или через кнопку “очистить список”. Товара не должно быть в сравнении.
10. Добавление магазина: переход на страницу “магазины”, нахождение нужного магазина по запросу, нажатие кнопки в форме сердца. Кнопка в

форме сердца должна стать закрашенной.

2. ОПИСАНИЕ КЛАССОВ И МЕТОДОВ

2.1. Классы и методы элементов страниц

Все классы элементов страниц наследуются от базового класса *BaseElement*, хранящего в себе *SelenideElement*[1] *baseElement* - DOM-элемент страницы, с которым происходят дальнейшие действия. Кроме этого в нем представлены методы: *isDisplayed()* - проверка, отображается ли элемент на странице, *waitNotDisplayed()* - проверка, перестал ли быть виден элемент, *scrollToElement()* - пролистывание страницы до момента, пока элемент не будет виден, *hover()* - наведение на элемент, и *getParamValue(..)* - метод, возвращающий заданный параметр элемента. UML-диаграмма наследуемых от *BaseElement* элементов представлена на рис. А.1.

Все классы элементов имеют поля констант вида *{param_name}_XPATH* - XPath шаблоны для поиска элемента по заданным параметрам, а также статические методы вида *by{param_name}(..)* возвращающие объект класса по заданным параметрам.

ButtonElement - класс элемента кнопки, наследуемый от *BaseElement*. В класс добавлены методы: *click()* - нажатие на кнопку, *isEnabled()* - проверка, доступна ли кнопка для взаимодействия.

StoreCardElement - класс элемента карточки магазина на странице магазинов, наследуемый от *BaseElement*. Содержит внутри себя объект кнопки добавления в избранное, а также методы *addToWishlist()* - нажатие на кнопку и *isInWishlist()* - проверка, нажата ли кнопка.

ProductConfiguratorListElement - класс элемента товара в списке конфигурации ПК, наследуется от *BaseElement*. Содержит метод получения кода товара *getAddedDataId()*.

InputElement - класс элемента ввода (кроме кнопок), наследуемый от *BaseTest*. Является основой для элементов ввода. UML-диаграмма классов, наследуемых от *InputElement*, представлена на рис. А.2.

SearchElement - класс элемента строки поиска, наследуемый от *InputElement*. Реализует метод *setValue(..)* для заполнения поля.

CheckBoxElement - класс элемента чекбокса, наследуемый от *InputElement*. Внутри реализованы методы *activate()* - активация чекбокса (если он еще не был активирован) и *isSelected()* - проверка на активность.

NameElement - класс элемента поля имени и фамилии в настройках аккаунта, наследуемый от *InputElement*. Имеет методы *setValue(..)* для заполнения и *getValue()* для возвращения значения внутри элемента.

PriceFilterElement - класс элемента поля цены в фильтрах цены, наследуемый от *InputElement*. Реализованы методы *getValue()* для возвращения текущей цены и *setValue(..)* для указания новой.

ProductSnippetElement - класс элементов карточек (сниппетов) товаров, наследуемый от *BaseElement*. Является основой для элементов карточек товаров разных страниц. UML-диаграмма классов, наследуемых от *ProductSnippetElement*, представлена на рис. А.3.

ProductBasketElement - класс элемента карточки товара в разделе корзины, наследуемый от *ProductSnippetElement*. Хранит внутри себя кнопку удаления товара (мусорный бак) и чекбокс для выделения товара. Реализованы соответствующие методы нажатия на кнопку *removeElementWithBin()* и активации чекбокса *clickRemoveCheckBox()*.

ProductCardElement - класс элемента карточки товара на остальных страницах, наследуемый от *ProductSnippetElement*. Является основой для сниппетов товаров на других страницах, хранит в себе кнопку добавления товара в корзину, метод нажатия на нее *addToCart()* и метод возврата цены товара *getPrice()*.

ProductCatalogElement - класс элемента карточки товара в каталоге, наследуется от *ProductCardElement*. Содержит в себе кнопки добавления в избранное и в сравнение, а также методы для нажатия на них *addToWishlist()* и *addToCompare()*.

ProductConfiguratorElement - класс элемента карточки товара в разделе конфигуратора, наследуемый от *ProductCatalogElement*. Содержит методы *getDataId()* - получение id товара и *addToConfigurator()* - добавление товара в конфигурацию.

ProductActionElement - класс элемента карточки товара в разделе лучших новинок, наследуемый от *ProductCardElement*. Хранит в себе кнопку добавления в избранное, метод, активирующий ее *addToWishlist()* и метод, возвращающий название товара *getTitle()*.

ProductListElement - класс элемента карточки товара в разделах избранное и сравнение, наследуемый от *ProductCardElement*. Является основой карточек данных разделов. Хранит кнопку удаления элемента из раздела (крестик) и метод для нажатия на нее *clickCrossButton()*.

ProductWishlistElement - класс элемента карточки товара на странице избранного, наследуемый от *ProductListElement*. Содержит кнопку добавления в сравнение и метод нажатия на нее *addToCompare()*.

ProductCompareElement - класс элемента карточки товара на странице сравнения, наследуемый от *ProductListElement*. Содержит кнопку добавления в избранное и метод нажатия на нее *addToWishlist()*.

2.2. Классы и методы страниц

Все классы страниц наследуются от базового класса *BasePage*, хранящего в себе *Class<? extends BasePage> pageClass* - класс страницы для возвращения объекта текущей страницы. Также внутри хранится *String expectedUrlPart* - часть URL, которая должна быть на конкретной странице. Кроме этого в нем представлены методы: *refresh()* - обновление страницы, *verifyPageUrl()* - проверка, содержит ли текущая открытая страница идентифицирующую ее часть URL, *page(..)* - создание нового объекта класса. UML-диаграмма наследуемых от BasePage классов страниц представлена на рис. А.4.

Все классы страниц имеют поля констант *URL_PART* - части URL, необходимые для идентификации данной страницы. Кроме этого у всех страниц

имеется метод вида *open{page_name}()*, возвращающий объект данной страницы.

BasketPage - класс страницы корзины, наследуемый от *BasePage*. Содержит кнопку “Удалить выбранные” и элемент карточки товара, а также методы: *IsEmptyOrder()* - проверка на пустоту корзины, *removeProductWithBin(..)* - активация сценария удаления товара через кнопку мусорного бака, *clickRemoveSelectedButton()* - нажатия на кнопку “Удалить все”, *selectProductWithCheckBox(..)* - активация чекбокса товара, *isProductRemoved()* - проверка, удалился ли товар, *containsProductWithName(..)* - проверка, содержится ли товар в корзине, *setUpElementByName(..)* - получение карточки товара по названию, *setUpElement()* - получение первой карточки товара на странице.

HomePage - класс главной страницы, наследуемый от *BasePage*. Хранит в себе элемент поля поиска и метод для его использования *search(..)*, кнопки “Войти”, “Избранное”, “Сравнение”, “Корзина”, “Каталог”, “Магазины”, “Лучшие новинки”, кнопки категорий и подкатегорий в каталоге, кнопка принятия cookie, а также методы вида *click{button_name}()* и *open{page_name}*, нажимающие на них. Кроме этого реализован метод login() - вход в аккаунт.

StorePage - класс страницы магазинов, наследуемый от *BasePage*. Хранит поле карточки магазина, а также методы для добавления магазина в избранное *addStore(..)* и проверка, добавлен ли магазин сейчас в избранное *isAdded()*.

SearchPage - класс страницы поиска, наследуемый от *BasePage*. Имеет методы: *addProductTo{page_name}ByName(..)* - добавление товара в корзину, избранное, сравнение, *getProductCardByName(..)* - получение карточки товара по названию, *usePriceFilter(..)* - использование фильтра цены, *getPrice()* - возврат цены первого товара, *getMaxPrice()* - возврат максимальной цены на странице.

SearchPage - класс страницы каталога, наследуемый от *BasePage*. Имеет метод *getProductCardByName(..)* - получение карточки товара по названию и

методы вида *addProductTo{page_name}ByName(..)* - добавление товара в корзину, избранное, сравнение.

SettingsPage - класс страницы настроек, наследуемый от *BasePage*. Содержит кнопку “Сохранить”, поля для ввода имени и фамилии, метод, изменяющий их *changeName(..)*, методы получения значений полей *getFirstname()* и *getLastname()* соответственно и метод нажатия на кнопку “Сохранить” *sumbitChanges()*.

ProfilePage - класс страницы профиля, наследуемый от *BasePage*. Содержит кнопку перехода в настройки и метод нажатия на нее *openSettings()*.

ConfiguratorPage - класс страницы конфигуратора, наследуемый от *BasePage*. Хранит элемент товара в конфигураторе, метод получения кода данного товара *getDataIdProcessor()*, метод перехода в каталог выбора товара *goToProcessorCatalog()*, получения рандомного товара из списка *getRandomProcessor()*, метод добавления товара в сборку *addProcessorToConfigurator()* а также метод создания сборки *createConfiguration()*.

ListPage - абстрактный класс страниц избранного и сравнения, наследуемый от *BasePage*. Содержит в себе элемент сниппета товара и кнопку “Удалить все”, метод, запускающий сценарий удаления товаров через кнопку “Удалить все” *removeProductWithRemoveAll()*, метод, запускающий сценарий удаления товара через кнопку крестика по названию *removeProductWithCrossButton(..)*, проверка на пустоту списка *isEmpty()*, проверка, удалился ли товар *isProductRemoved()*, проверка есть ли товар в списке *containsProductWithName(..)*, метод нажатия на кнопку “Удалить все” *clickRemoveListButton()*, абстрактные методы получения первой карточки товара и карточки товара по названию *setUpElement()* и *setUpElementByName(..)* соответственно.

WishlistPage - класс страницы избранного, наследуемый от *ListPage*. реализует методы получения первой карточки товара и карточки товара по

названию на странице избранного *setUpElement()* и *setUpElementByName(..)* соответственно.

ComparePage - класс страницы сравнения, наследуемый от *ListPage*. реализует методы получения первой карточки товара и карточки товара по названию на странице сравнения *setUpElement()* и *setUpElementByName(..)* соответственно.

2.3. Классы загрузки данных и их методы

LoadLoginData - класс для загрузки данных пользователя для авторизации. Содержит поля *username* и *password* - имя пользователя и пароль соответственно. Метод *loadLoginData()* загружает данные из json файла, при помощи методов *getUsername()* и *getPassword()* можно получить значения загруженных полей.

LoadWriteData - класс для загрузки и записи данных товаров и магазинов, используемых в тестах. Инициализирует enum *ActionType* - тип действия, которое требуется выполнить, *data* - список считанных данных и *actionType* - текущее действие. При создании элемента сразу задается тип действия, благодаря которому становится понятно, из какого json файла требуется чтение. *loadData(..)* загружает данные из определенного json файла, метод *getRandomData()* случайным образом выбирает элемент из загруженного списка и возвращает его, метод *doActionLogic()* определяет последующий сценарий, зависящий от выполняемого действия: это может быть как добавить текущий выбранный элемент в какой-то json файл, убрать текущий элемент из какого-то json файла, так и не делать больше ничего. Метод *removeProducts(..)* удаляет поданный элемент из поданного json файла, метод *writeProducts(..)* записывает поданный элемент из поданного json файла. UML-диаграммы классов *LoadLoginData* и *loadWriteData* представлены на рис. А.5.

2.4. Классы и методы тестов

Все классы тестов наследуются от базового класса *BaseTest*, хранящего в себе *HomePage homePage* - объект главной страницы сайта. Кроме этого в нем представлены методы с аннотацией *@BeforeEach*, что означает, что данный метод должен выполняться перед каждым тестом: *setup()* - настройка браузера, в котором будет происходить UI-тестирование, *login()* - открытие браузера и вход в аккаунт. Также представлен метод с аннотацией *@AfterEach*, что означает, что данный метод должен выполняться после каждого теста: *tearDown()* - закрытие браузера. UML-диаграмма наследуемых от *BaseTest* тестов представлена на рис. А.6.

TestWithDataName - класс, наследуемый от *BaseTest*, хранящий в себе загрузчик данных *LoadWriteData loader* и название запроса *String dataName*, а также метод создания загрузчика по поданному типу действия *loadByActionType*. Класс является базовым для тестов со случайным выбором данных.

AddProcessorToConfiguratorTest - класс теста, проверяющего добавление процессора в конфигуратор ПК, наследуемы от *BaseTest*. Содержит метод *addProcessorToConfigurator()*, объединяющий последовательность действий: открытие страницы конфигуратора, создание новой конфигурации, переход в каталог выбора процессора, выбор случайный процессора в каталоге, запоминание кода товара выбранного процессора, добавление его в конфигурацию, проверка того, что процессор действительно был добавлен в конфигурацию.

AddProductToBasketTest - класс теста, проверяющего добавление в корзину случайного выбранного товара посредством его поиска, наследуемый от *TestWithDataName*. Содержит метод *addProductToBasket()*, объединяющий последовательность действий: поиск товара по тестируемому запросу, полученному случайным образом из JSON файла, добавление товара в корзину, проверка, находиться ли товар в корзине.

AddProductToCompareTest - класс теста, проверяющего добавление в сравнение тестового товара через каталог, наследуемый от *TestWithDataName*.

Содержит в себе метод *addProductToCompare()*, объединяющий последовательность действий: открытие каталога, выбор тестовой категории, выбор тестовой подкатегории, нахождение тестового товара, добавление тестового товара в сравнение, проверка, находиться ли тестовый товар в разделе сравнения.

AddProductToWishListTest - класс теста, проверяющего добавление в избранное случайного товара через раздел акций, наследуемый от *TestWithDataName*. Содержит метод *addProductToWishList()*, объединяющий последовательность действий: поиск раздела лучших новинок, добавление первого товара из раздела лучших новинок в избранное, проверка находиться ли товар в избранном.

AddStoreTest - класс теста, проверяющего добавление товара в избранное, наследуемый от *TestWithDataName*.. Содержит метод *addStore()*, объединяющий последовательность действий: открытие страницы магазинов, выбор магазина, нажатие на кнопку сердечка, проверка что магазин добавлен в избранные.

ChangeProfileNameTest - класс теста, проверяющего корректность изменения имени и фамилии в профиле, наследуемый от *BaseTest.* Содержит метод *changeProfileName()*, объединяющий последовательность действий: заход в аккаунт, переход на страницу профиля, переход на страницу настроек, смена имени и фамилии в соответствующих полях, подтверждение изменения, проверка изменилось или нет.

RemoveBasketProductTest - тест-класс, проверяющий функционал удаления товара из корзины, наследуемый от *TestWithDataName.* Содержит методы *openBasketPage()* - переход в корзину и проверка ее на пустоту, метод *removeBasketProductWithCheckBox()*, объединяющий последовательность действий: переход в корзину, выбор тестового товара, выделение чекбокса тестового товара, нажатие на кнопку "Удалить выбранные", проверка, удалился ли товар. Метод *removeBasketProductWithBin()*, объединяющий последовательность действий: переход в корзину, выбор тестового товара,

удаление тестового товара при помощи кнопки мусорного бака, проверка, удалился ли товар.

RemoveCompareProductTest - тест-класс, проверяющий функционал удаления товара из сравнения, наследуемый от *TestWithDataName.* Содержит методы *openComparePage()* - переход в сравнение и проверка его на пустоту, метод *removeCompareProductWithRemoveAll()*, объединяющий последовательность действий: переход в сравнение, нажатие на кнопку “Очистить список”, проверка, удалился ли товар. Метод *removeCompareProductWithCross()*, объединяющий последовательность действий: переход в сравнение, выбор тестового товара, удаление тестового товара при помощи крестика, проверка, удалился ли товар.

RemoveWishlistProductTest - тест-класс, проверяющий функционал удаления товара из избранного, наследуемый от *TestWithDataName.* Содержит методы *openWishlistPage()* - переход в избранное и проверка его на пустоту, метод *removeWishlistProductWithCross()*, объединяющий последовательность действий: переход в избранное, выбор тестового товара, удаление тестового товара при помощи крестика, проверка, удалился ли товар. Метод *removeWishlistProductWithRemoveAll()*, объединяющий последовательность действий: переход в избранное, нажатие на кнопку “Очистить список”, проверка, удалился ли товар.

UsingPriceFiltersTest - класс теста, проверяющего корректность работы фильтров цены, наследуемый от *TestWithDataName.* Содержит метод *usePriceFilters()*, объединяющий последовательность действий: поиск товара по тестируемому запросу, применение фильтра цены (выставляет минимальную цену), проверка, что первый товар на странице поиска имеет цену не менее тестовой.

UML-диаграммы классов (рис. А.1-А.6) см. в приложении А.

3. ТЕСТИРОВАНИЕ

3.1. Демонстрация работы теста №1

В рамках первого теста проверяется добавление товара в корзину через поисковую систему сайта. Последовательность действий включает:

- Авторизацию в аккаунт с использованием входных данных (см. рисунки 1–3).
- Ввод поискового запроса в соответствующее поле и нажатие на кнопку поиска (см. рисунки 4 и 5).
- Выбор нужного товара из результатов поиска и его добавление в корзину (см. рисунок 6).
- Проверку успешного добавления товара в корзину (см. рисунок 7).

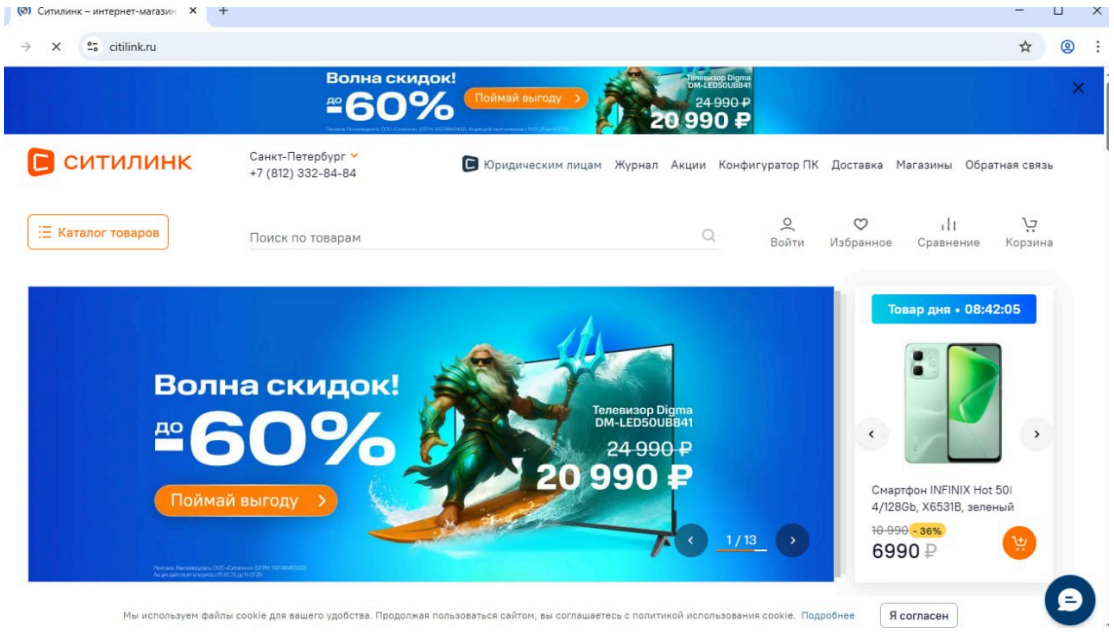

Рисунок 1 - главная страница сайта

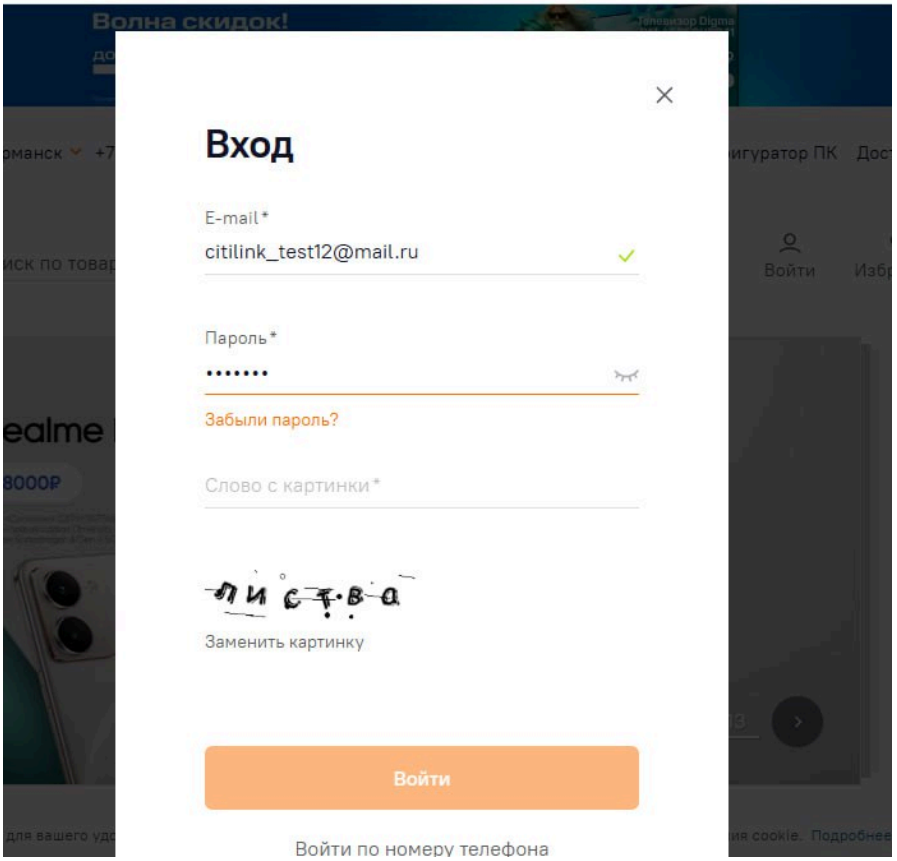

Рисунок 2 - ввод E-mail и пароля

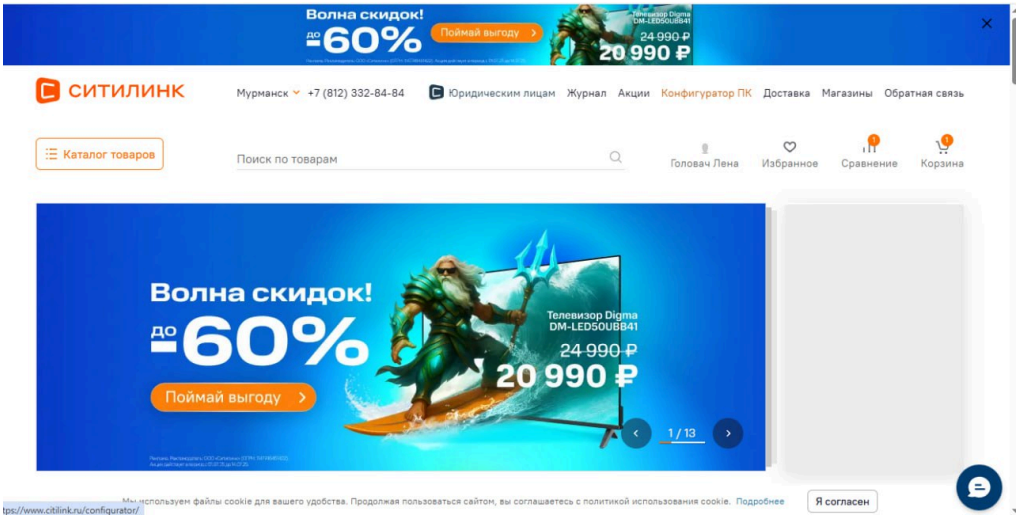

Рисунок 3 - главная страница сайта после авторизации

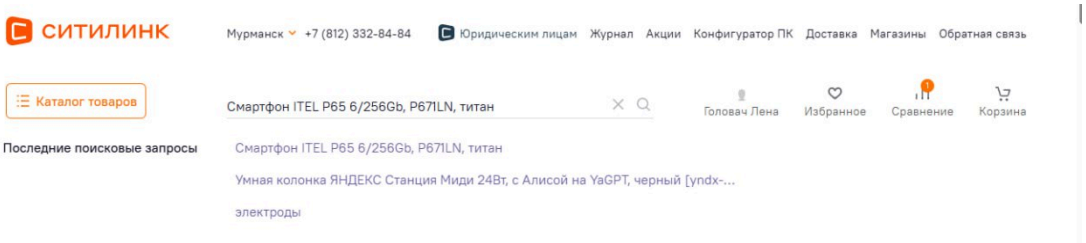

Рисунок 4 - ввод поискового запроса

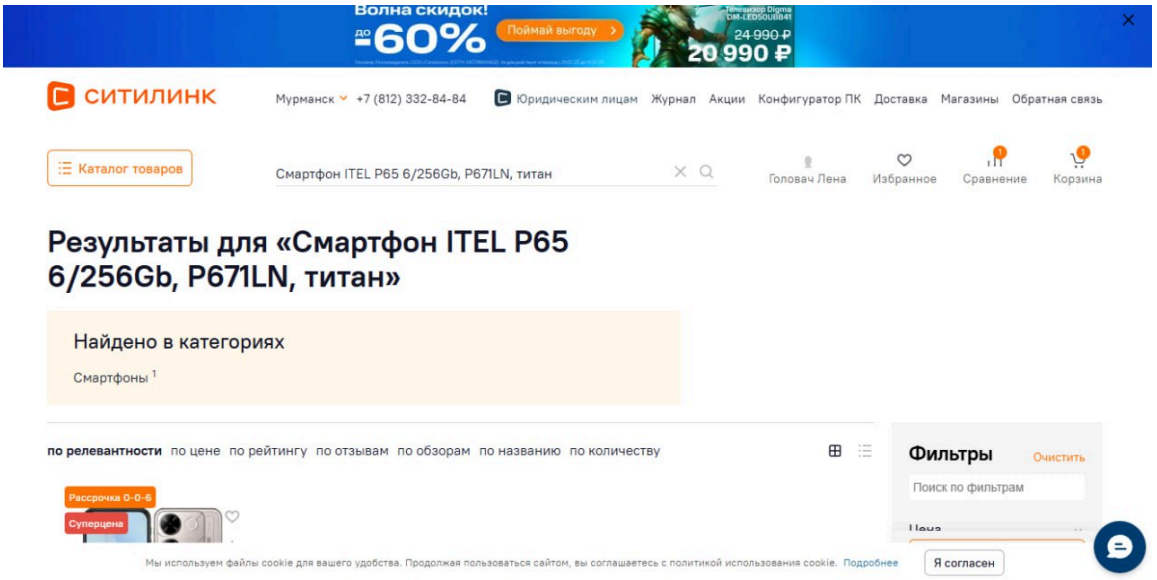

Рисунок 5 - результат поискового запроса

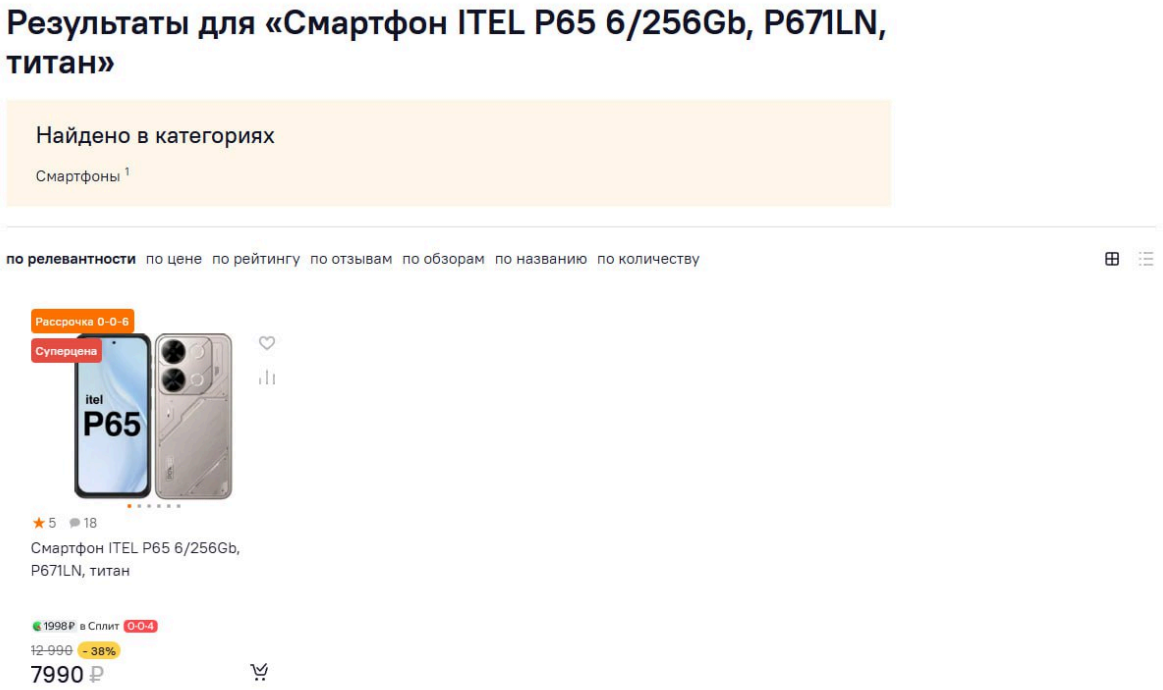

Рисунок 6 - добавление товара в корзину

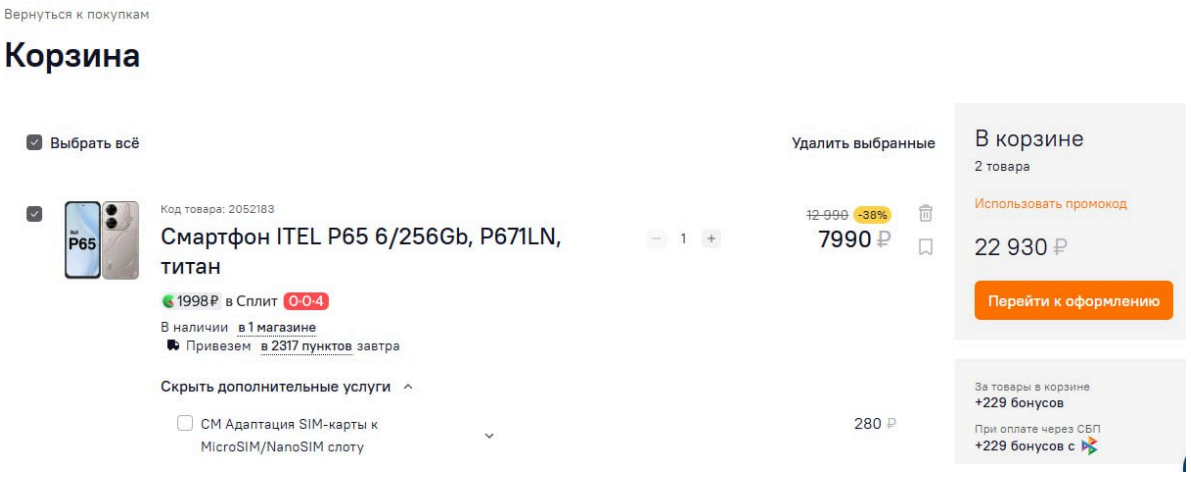

Рисунок 7 - товар появился в корзине

3.2. Демонстрация работы всех тестов.

На рис. 8 продемонстрирован результат работы всех тестов.

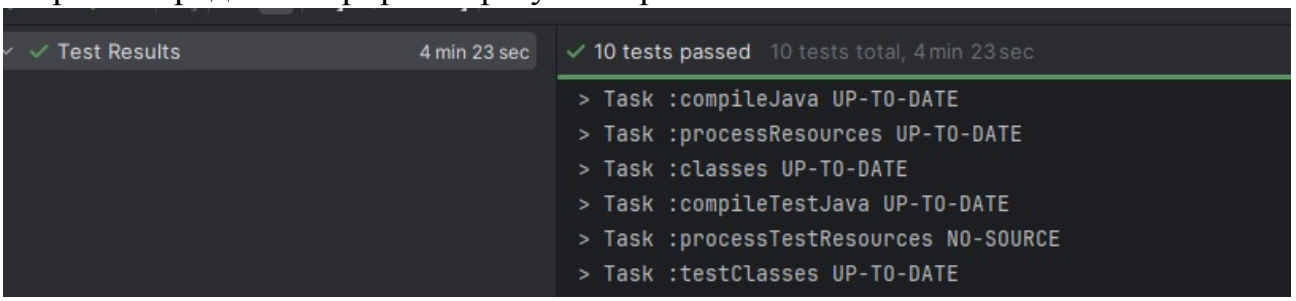

Рисунок 8 - демонстрация работы тестов

ЗАКЛЮЧЕНИЕ

В ходе практической работы была создана архитектура для реализации десяти UI-тестов сайта «Ситилинк», в основе которой лежат базовые классы элементов и страниц. Базовый класс элемента определяет общий интерфейс для всех компонентов и включает методы ожидания, наведения и проверки видимости. От него наследуются классы, такие как «Кнопка», «Поисковая строка», «Карточка товара в корзине», «Карточка товара в сравнении», «Карточка товара в каталоге», «Карточка товара в конфигураторе», «Карточка товара в избранном», «Карточка товаров в разделе лучших новинок», «Карточка магазина», «Чекбокс», «Поле ввода имени», «Поле ввода фильтра», которые расширяют и конкретизируют поведение базового интерфейса. Аналогичным образом построен базовый класс страницы, отвечающий за создание нового элемента, обновление страницы и проверка URL. От него наследуются классы — «Главная страница», «Страница каталога», «Страница избранного», «Страница корзины», «Страница сравнения», «Страница результатов поиска», «Страница настроек», «Страница списка магазинов», «Страница конфигуратора ПК», и «Страница профиля». В итоге была получена архитектура, которая позволяет быстро добавлять новые тесты, наследуя от уже созданных классов. В случае необходимости изменения общего механизма взаимодействия с элементами (например, алгоритма ожидания полной загрузки страницы) корректировка вносится единожды в базовом классе, без необходимости правок в других классах. Таким образом получилась гибкая и масштабируемая архитектура для быстрой реализации UI-тестов. В тестовых классах реализованы десять сценариев: проверка добавления товара через поиск, фильтрация по цене, смена имени пользователя и другие.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Пишем автотесты для UI на базе Selenide. Часть 2. URL: https://testit.software/blog/post/pishem-avtotesty-dlya-ui-na-baze-selenide-chast-2 (дата обращения: 26.06.2025).
2. Java SE 11-17 Documentation. Oracle. URL: https://docs.oracle.com/javase/ (дата обращения: 26.06.2025).
3. Как проводить UI-тестирование мобильных и верстки + сравнение инструментов автоматизации. URL: https://ux-journal.ru/kak-provodit-ui-testirovanie-sravnenie-instrumentov.html (дата обращения 26.06.25).
4. Как написать UI-автотесты? URL: https://habr.com/ru/companies/rostelecom/articles/707710/ (дата обращения 26.06.25).

ПРИЛОЖЕНИЕ А

UML-ДИАГРАММЫ КЛАССОВ

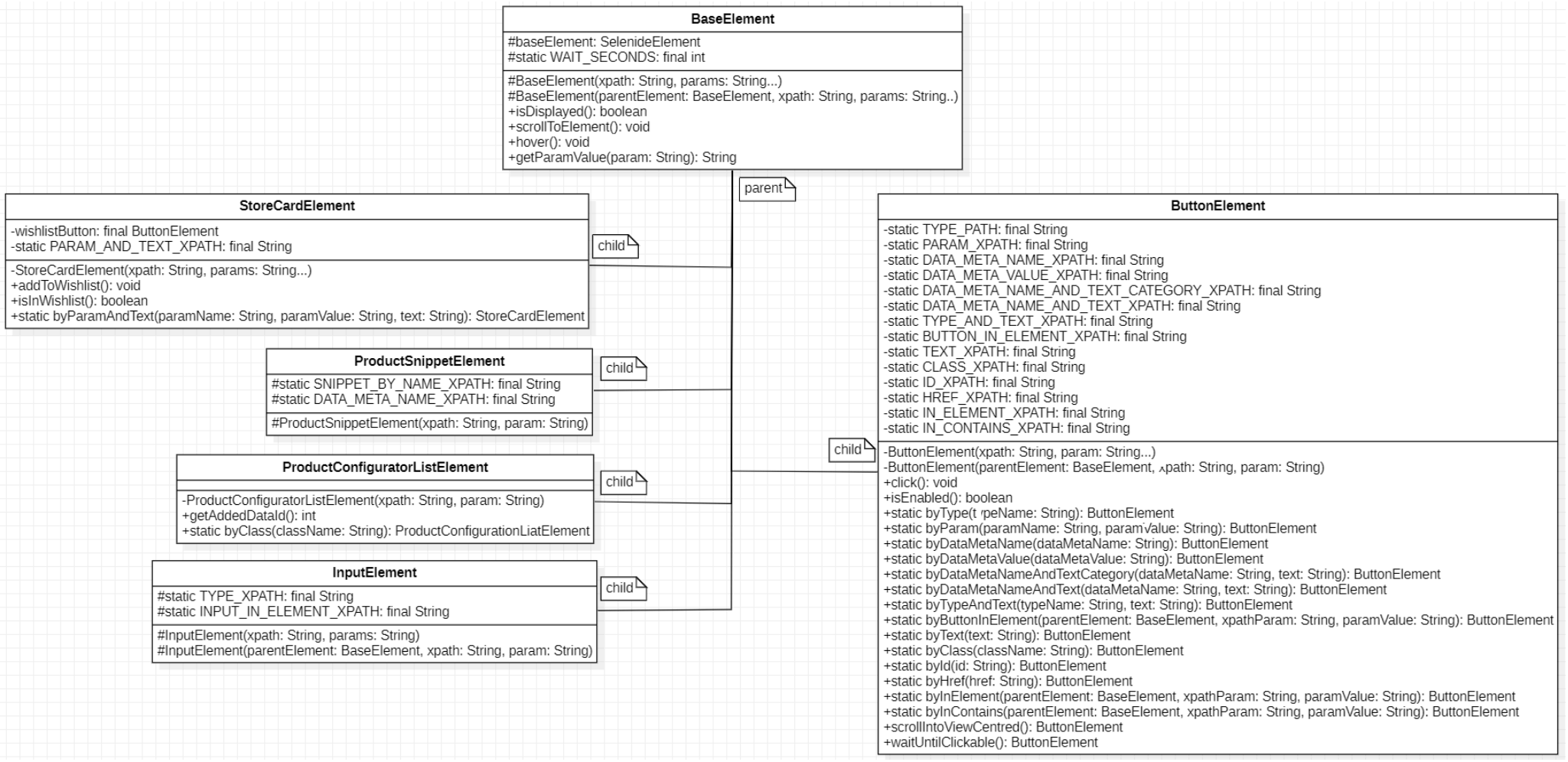

Рисунок А.1 - UML-диаграмма наследников BaseElement

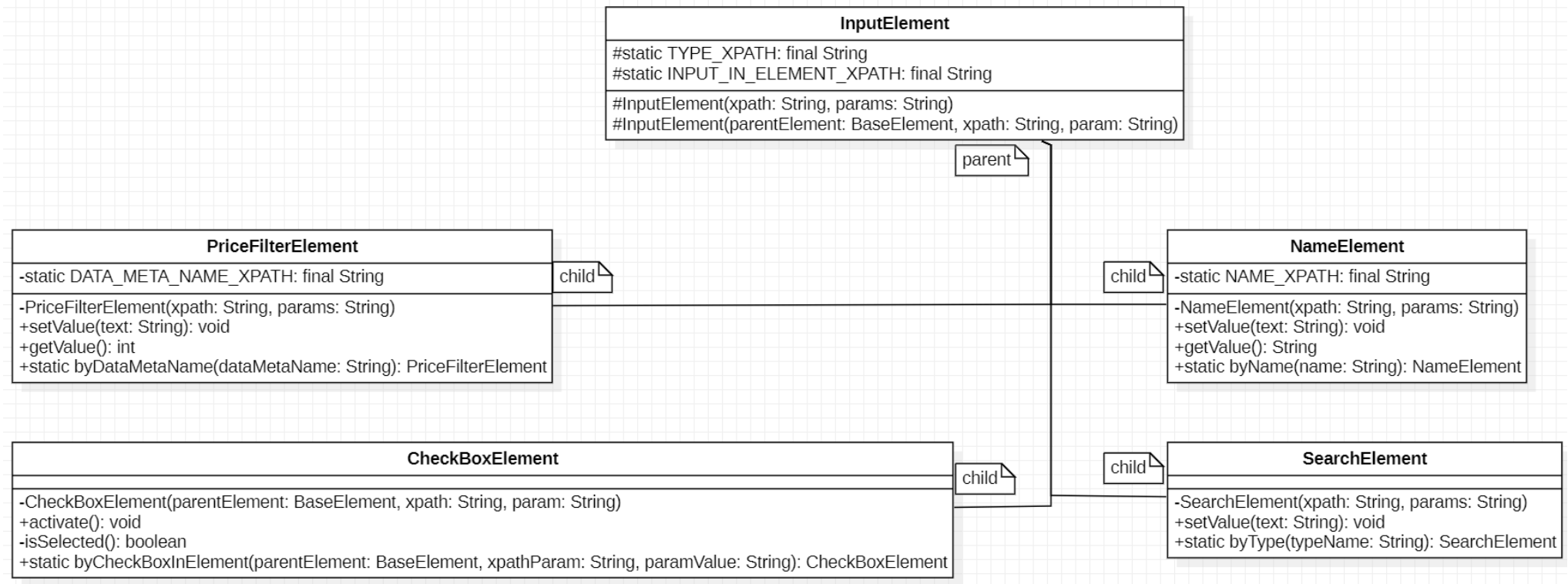

Рисунок А.2 - UML-диаграмма наследников InputElement

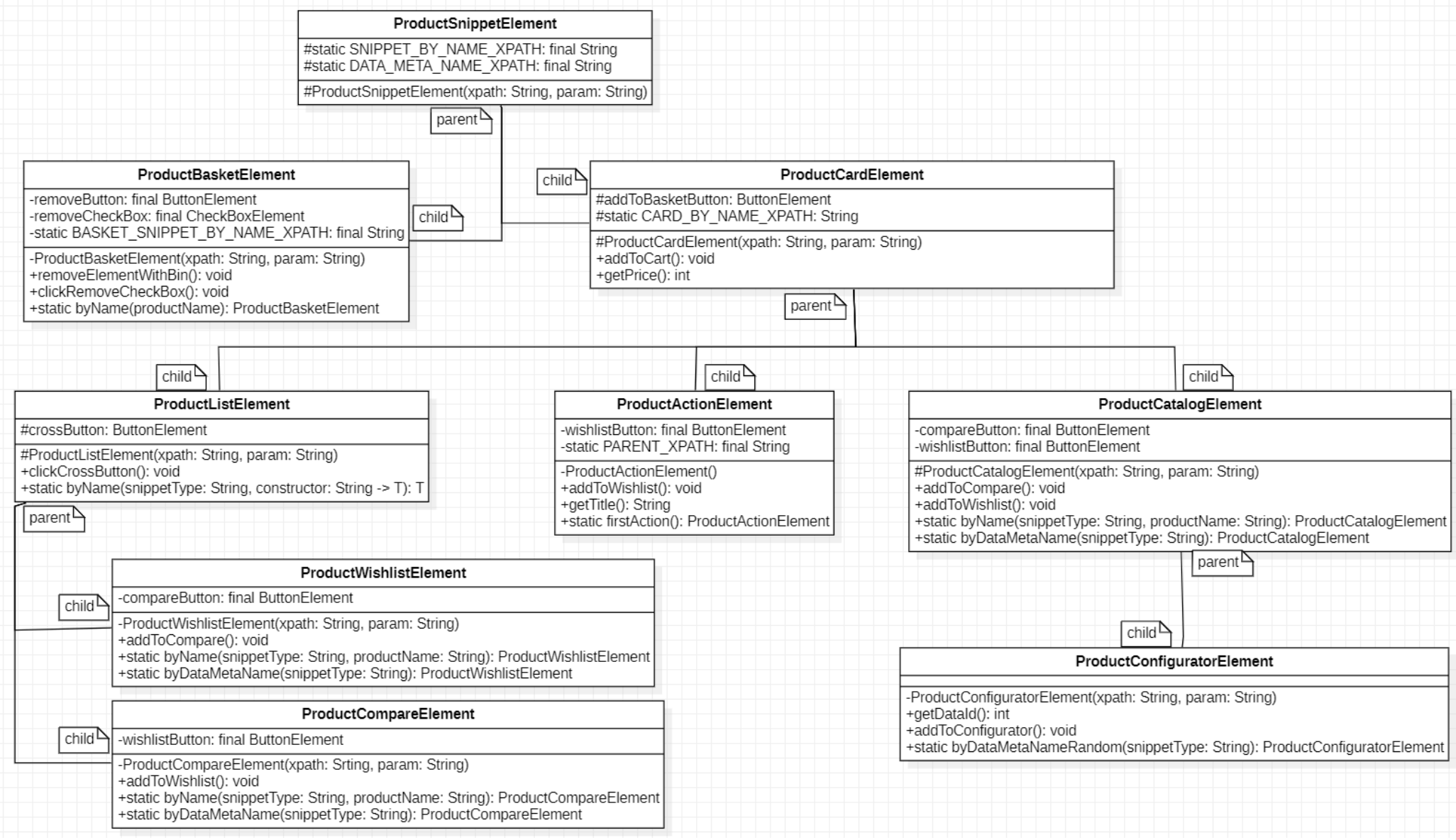

Рисунок А.3 - UML-диаграмма наследников ProductSnippetElement

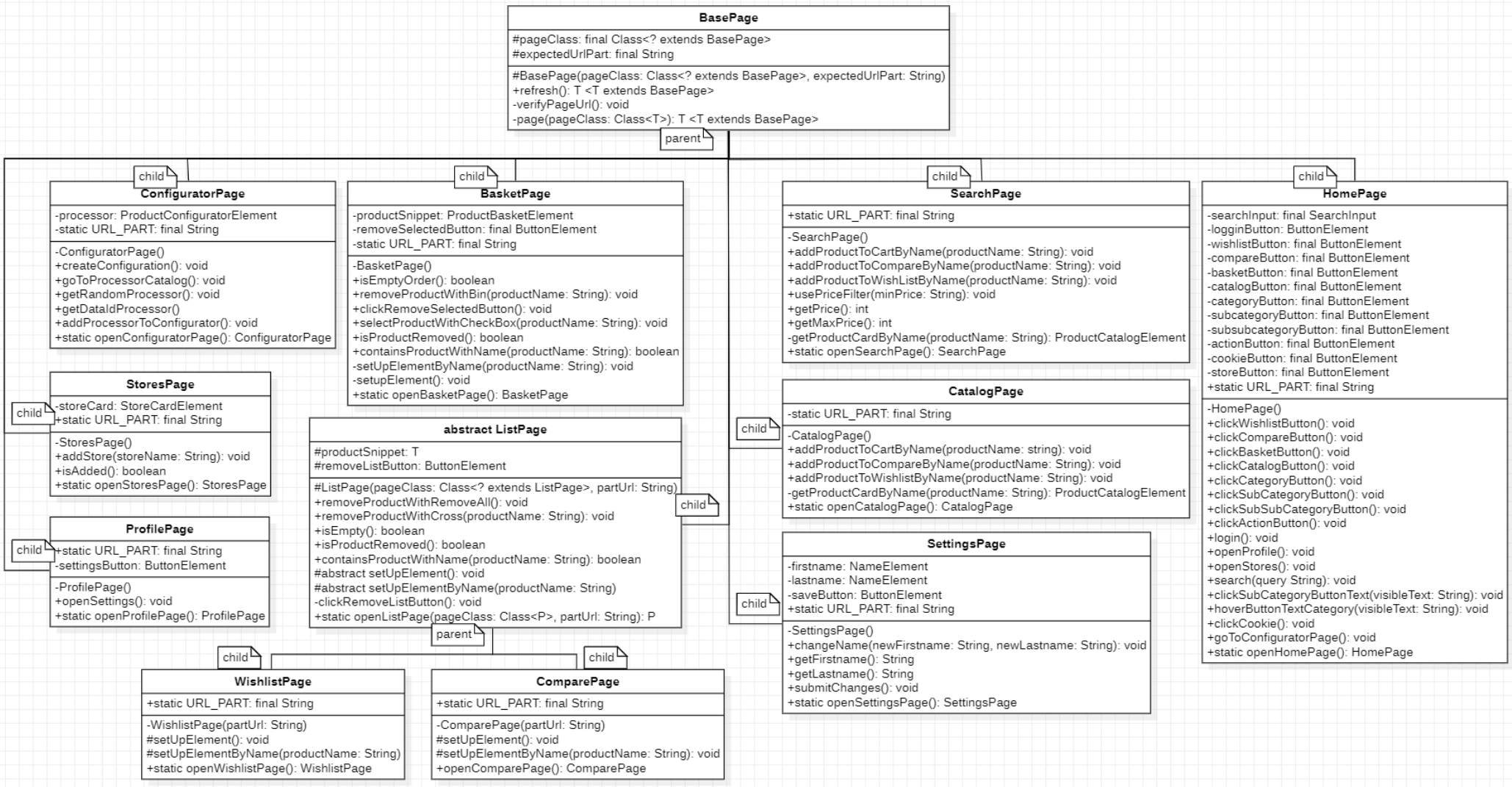

Рисунок А.4 - UML-диаграмма наследников BasePage

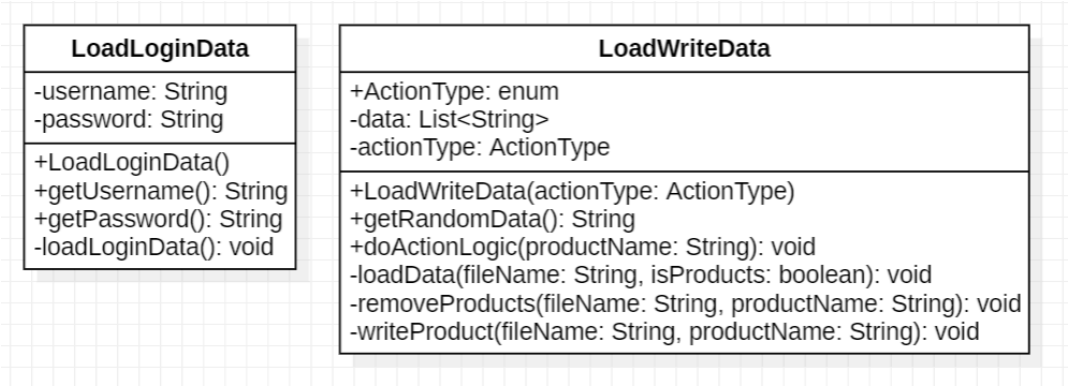

Рисунок А.5 - UML-диаграмма классов загрузки данных

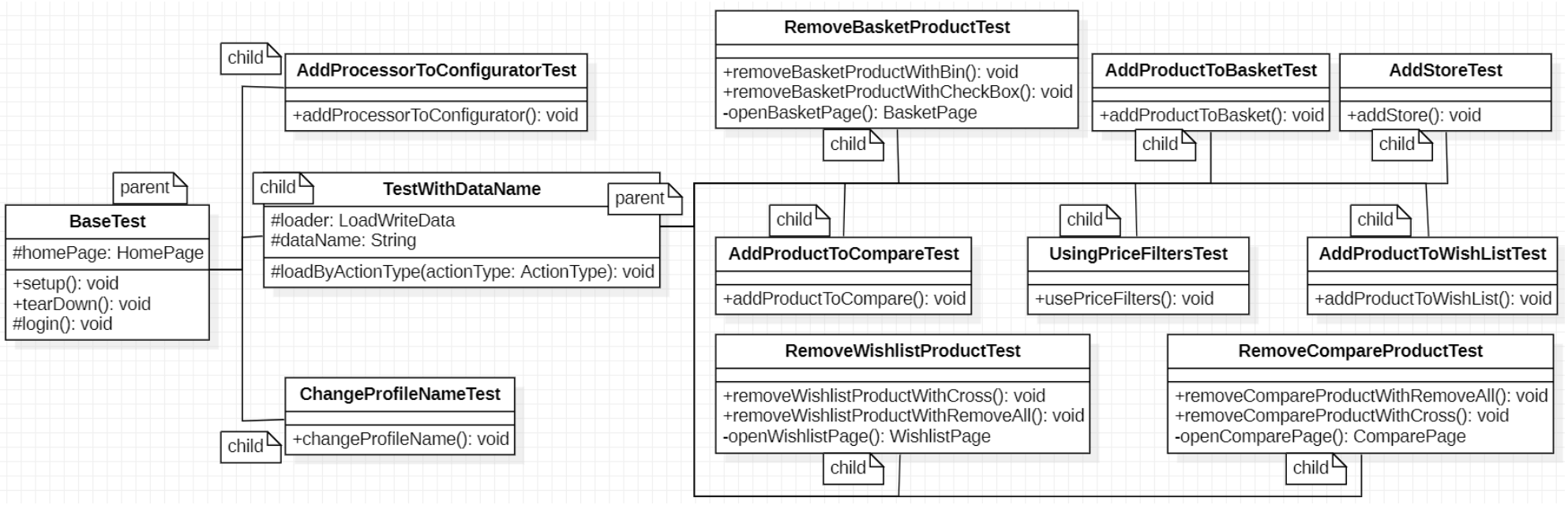

Рисунок А.6 - UML-диаграмма наследников BaseTest