ТЕСТИРОВАНИЕ ДОБАВЛЕНИЯ ЗАКАЗОВ В КОРЗИНУ НА САЙТЕ DOMINOS.BY V2

Поставленные задачи

1. Зайти на домашнюю страницу сайта.
2. В разделе «Пицца» добавить в корзину пиццу «Маргарита».
3. В разделе «Напитки» добавить в корзину любой напиток.
4. Перейти в корзину.
5. Проверить, что пицца и напиток есть в заказе.

Применяемые фреймворки и паттерны:

Junit	Page Object
Selenium	Page Factory

Реализация добавления товаров в корзину

Проектом предусматриваются унифицированные методы класса CatalogPage для добавления товаров в корзину и перехода по категориям.

Данный класс, в отличие от прошлого проекта, унифицирован для работы с любым типом товаров в любой категории, для чего выполнено объединение всех типов товаров в один Enum

Для формирования локатора используется Enum содержащий наименования типов продуктов и уникальные значения локатора, используемого в шаблоне, для добавления товара в корзину.

Аналогичные методы реализованы для переходов по категориям товаров.

```
public CatalogPage addToShoppingCart(ProductType type) {
    String xpathPattern = "//div[@data-code='%s']//descendant::button[@data-action='a2b']";
    String xpath = String.format(xpathPattern, type.getXpathValue());
    webDriver.findElement(By.xpath(xpath))
            .click();
    return this;
}
```

```
public enum ProductType {
    MARGARITA("MGRC", "Маргарита"),
    COCA_COLA_1L("KK1", "Кока-Кола 1 л");
    private final String xpathValue;
    private final String name;

    ProductType(String xpathValue, String name) {
        this.xpathValue = xpathValue;
        this.name = name;
    }

    public String getXpathValue() {
        return xpathValue;
    }

    public String getName() {
        return name;
    }
}
```

Получение позиций заказа

Для дальнейшей проверки правильности добавленных товаров в корзину, выполняемой в тесте, необходимо получить список наименований всех позиций корзины.

В отличие от прошлого проекта, в данном реализовано приведение полученного списка наименований товаров из корзины к Enum с типами товаров.

```
public List<ProductType> getOrderPositionFromShoppingCart() {
    List<ProductType> ordersPositionName = new ArrayList<>();
    for (WebElement position : ordersPositionFromShoppingCart) {
        String positionName = position.getText();
        ordersPositionName.add(getOrderPositionType(positionName));
    }
    return ordersPositionName;
}

private ProductType getOrderPositionType(String name) {
    for (ProductType value : ProductType.values()) {
        if (value.getName().equals(name)) {
            return value;
        }
    }
    return null;
}
```

Подготовка списков товаров к сравнению

Для сверки полученных наименований с ожидаемыми необходимо сформировать список типов товаров, добавленных в корзину:

```
List<ProductType> expectedPosition = Arrays.asList(ProductType.MARGARITA, ProductType.COCA_COLA_1L);
```

А так же передать в список перечень типов товаров полученных из корзины:

```
List<ProductType> actualPosition = catalogPage.openShoppingCart()
        .getOrderPositionFromShoppingCart();
```

Сверка списков, выполняемая в тесте чувствительна к порядку элементов в списках, в связи с чем необходимо выполнить сортировку обоих списков:

```
actualPosition.sort(Comparator.naturalOrder());
expectedPosition.sort(Comparator.naturalOrder());
```

Содержание тестового метода и структура проекта

```java
public class OrderVerificationTest extends AbstractTest {

    @Test
    public void OrderVerification() {
        CatalogPage catalogPage = new CatalogPage();
        catalogPage.closePopupBanner()
                .openCategory(Categories.PIZZA)
                .addToShoppingCart(ProductType.MARGARITA)
                .openCategory(Categories.DRINKS);
        Utilities.sleep(2);
        catalogPage.addToShoppingCart(ProductType.COCA_COLA_1L);
        List<ProductType> actualPosition = catalogPage.openShoppingCart()
                .getOrderPositionFromShoppingCart();
        List<ProductType> expectedPosition = Arrays.asList(ProductType.MARGARITA, ProductType.COCA_COLA_1L);
        actualPosition.sort(Comparator.naturalOrder());
        expectedPosition.sort(Comparator.naturalOrder());
        Assert.assertEquals(expectedPosition, actualPosition);
    }
}
```