Testowanie w metodyce BDD

Zadanie 1

Na podstawie poniższego kodu źródłowego reprezentującego koszyk sklepowy:

```
public class Checkout {
    private final HashMap<String, Item> items = new HashMap<>();

    public void reset() {
        items.clear();
    }

    public void addItem(String name, int price) {
        if (name != null && !name.isEmpty() && !name.trim().isEmpty() &&
price > 0) {
            items.put(name, new Item(name, price));
        }
    }

    public void scanItems(String name, int count) {
        Item item = items.get(name);
        if (item != null) {
            item.count = item.count + count;
        }
    }

    public int totalCart() {
        int total = 0;
        for (Item item : items.values()) {
            total += item.price * item.count;
        }
        return total;
    }
}

class Item {
    String name;
    int price = 0;
    int count = 0;

    public Item(String name, int price) {
        this.name = name;
        this.price = price;
        this.count = 0;
    }
}
```

oraz poniższych definicji kroków:

```
Feature: Checkout

  Scenario: Add item with empty name
    Given the price of a "" is 25c
    When I checkout "" 1
    Then the total price should be 0c

  Scenario: Add item with empty name
    Given the price of a "banana" is -1c
    When I checkout "banana" 1
    Then the total price should be 0c

  Scenario: Checkout bananas twice
    Given the price of a "banana" is 40c
    When I checkout "banana" 1
    And I checkout "banana" 1
    Then the total price should be 80c

  Scenario: Checkout non existing items
    Given empty product list
    When I checkout "apple" 1
    And I checkout "banana" 2
    Then the total price should be 0c

  Scenario Outline: Checkout apples and bananas
    Given the price of a "banana" is 40c
    And the price of a "apple" is 25c
    When I checkout "apple" <applecount>
    And I checkout "banana" <bananacount>
    Then the total price should be <total>c

    Examples:
      | applecount | bananacount | total |
      |     1      |  1          | 65    |
      |     2      |  2          | 130   |
```

Przygotuj testy akceptacyjne z wykorzystaniem framework'a Cucumber.

Zadanie 2

Na podstawie poniższego kodu źródłowego:

```
public class BankAccount {
    private int accountBalance;

    public BankAccount() {
        accountBalance = 0;
    }

    public BankAccount(int accountBalance) {
        this.accountBalance = accountBalance;
    }

    public boolean withdraw(int amount) {
        if (amount > accountBalance) {
            return false;
        }
        accountBalance -= amount;
        return true;
    }

    public void transfer(int amount) {
        accountBalance += amount;
    }

    public int getAccountBalance() {
        return accountBalance;
    }
}
```

Zaprojektuj definicje kroków oraz testy akceptacyjne biorąc pod uwagę poniższe funkcjonalności:

- pobieranie środków z konta, które nie przekraczają kwoty na rachunku
- pobieranie środków z konta, które przekraczają kwotę na rachunku
- wielokrotne pobieranie środków z konta
- przelewanie pieniędzy na konto

Zadanie 3

Przygotuj testy automatyczne w oparciu o:

- Cucumber
- Selenium

Celem zadania jest przygotowanie testów automatycznych dla strony: *https://www.wikipedia.org* w oparciu o poniższe scenariusze:

- wyszukiwanie frazy BDD
- wyszukiwanie frazy BDD
- wyszukiwanie pustej frazy
- wyszukiwanie frazy składającej się z samych białych znaków

Zadanie 4

Przygotuj projekt oparty o framework:

- Cucumber
- Selenium

Celem zadania jest zaimplementowanie testów automatycznych w oparciu o BDD dla aplikacji webowej: *http://automationpractice.com*

W ramach struktury projektu stwórz co najmniej:

- pakiet steps: zawierający klasy reprezentujące definicje kroków
- klasę DriverFactor: odpowiedzialną za obsługę konkretnych WebDriver

Przeglądanie katalogu ubrań:

Przygotuj scenariusze oraz testy automatyzujące dla functional przeglądania ubrań w katalogu. Uwzględnij następujące scenariusze:

- przejście do kategorii T-Shirt
- przejście do kategorii Dresses
- przejście do kategorii Women

Wyszukiwanie:

Przygotuj scenariusze oraz testy automatyzujące dla funkcjonalności wyszukiwania w aplikacji. Uwzględnij następujące scenariusze:

- wyszukiwanie bez podawania frazy
- wyszukiwanie nieistniejących elementów
- wyszukiwanie elementów istniejących

Subskrypcja do newsletter:

Przygotuj scenariusze oraz testy automatyzujące dla funkcjonalności subskrypcji do newsletter. Uwzględnij następujące scenariusze:

- rejestracja z niepoprawnym adresem email
- rejestracja z poprawnym adresem email
- rejestracja z adresem email istniejącym w bazie