

Dokumentacja implementacji

Point of Sale

Zuzanna Kulik

Spis treści

1. Wprowadzenie.....	3
2. Architektura aplikacji.....	3
3. Model danych.....	4
4. Procedura uruchomieniowa.....	5
4a. Dane testowe.....	5
4b. Przykładowe uruchomienie w trybie ConsoleScannerMock.....	6

1. Wprowadzenie

Niniejszy dokument opisuje implementację wymagań przekazanych w zadaniu rekrutacyjnym oraz dodatkowo wprowadzonych zmian, a także działanie stworzonej aplikacji.

2. Architektura aplikacji

Zadaniem implementacji było zapewnienie prostej funkcjonalności punktu sprzedaży – kasy sklepowej z urządzeniem wejściowym (skaner kodów kreskowych) oraz dwoma urządzeniami wyjściowymi (wyświetlacz LCD oraz drukarka).

Wymagana była obsługa 4 przypadków:

1. Zeskanowany kod pozwalał na odnalezienie produktu w bazie - nazwa i cena wyświetlają się na wyświetlaczu
2. Zeskanowany kod nie pozwalał na odnalezienie produktu w bazie – wyświetlenie informacji „Product not found”
3. Zeskanowany kod był pusty - wyświetlenie informacji „Invalid barcode”
4. Kasa otrzymała input „exit” – na wyświetlaczu pojawia się cena, którą klient musi zapłacić za wszystkie poprzednio zeskanowane produkty, drukowany jest paragon z listą produktów oraz finalnym rachunkiem

Dodatkowo zaimplementowane funkcjonalności:

1. Możliwość wybrania ile razy (w jakiej ilości) ma być dodany zeskanowany produkt – po zeskanowaniu produktu ręcznie wprowadza się za pomocą klawiatury ilość, która ma być dodana do rachunku. W przypadku nie wprowadzenia żadnej wartości, domyślną ilością jest jeden.
2. Wyłapywanie niepoprawnego formatu zeskanowanego kodu.
3. Zapisywanie historii paragonów wydanych w jednej sesji działania programu przez jedną kasę (historia wraz z godzinami dodania paragonów oraz łączną sumą wszystkich rachunków jest zapisywana w pliku, którego nazwa tworzona jest od identyfikatora kasy)
4. Możliwość uruchomienia programu w dwóch trybach imitacji procesu skanowania: kody mogą być losowane z wbudowanej tablicy lub wprowadzane ręcznie za pomocą klawiatury.

UWAGA: Poza klasami SDK, została użyta biblioteka umożliwiająca korzystanie z bazy danych SQLite.

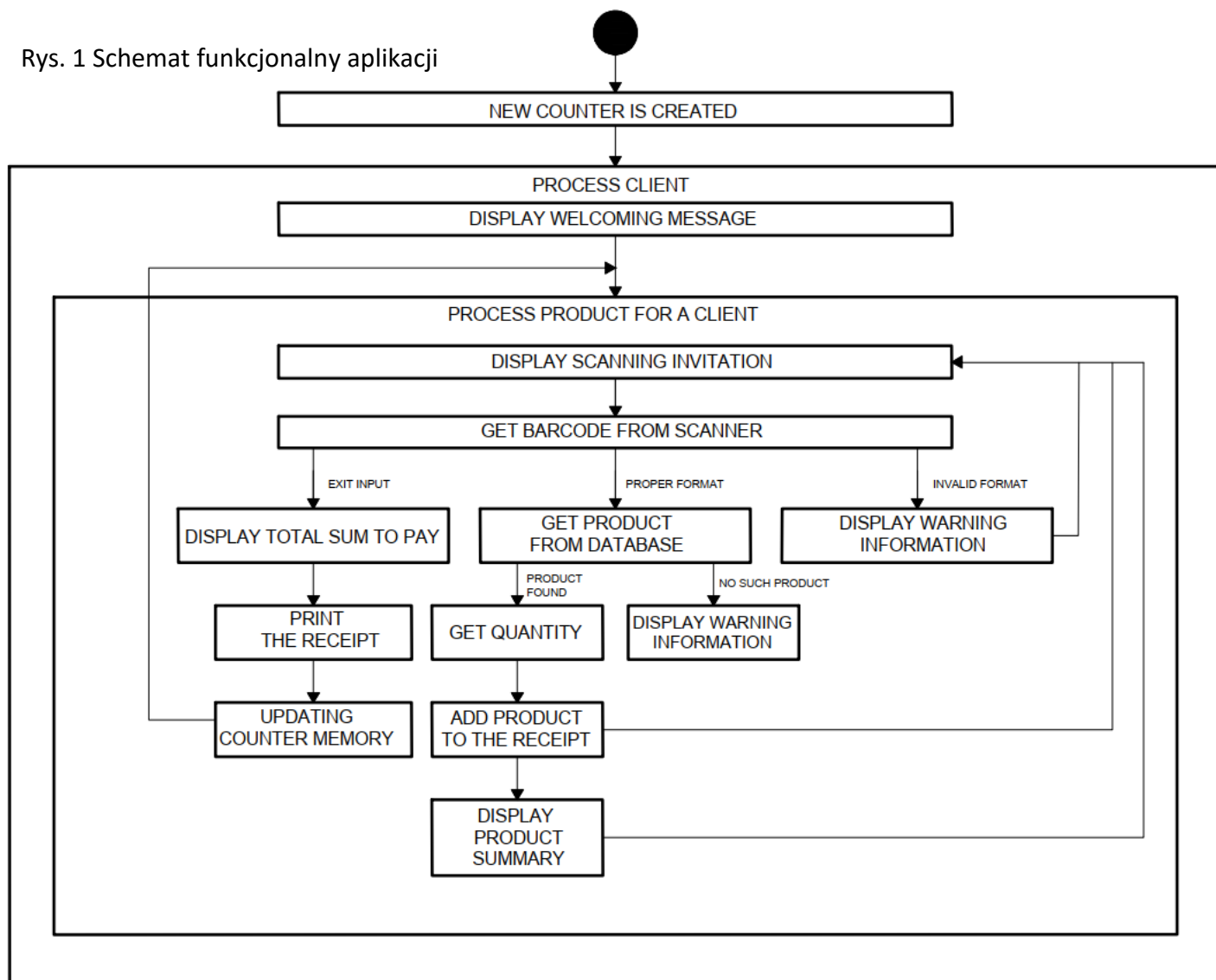
Z punktu widzenia architektury Point of Sale jest trójwarstwową aplikacją opartą o następujące składowe:

- warstwę prezentacji danych: wyświetlanie na ekranie, drukowanie (zapis do pliku jako zaślepka)
- warstwę logiki aplikacyjnej (rys. 1) : wywołanie skanowania, przyjęcie danych, odczyt produktu z bazy danych, obliczenie ceny za produkt po pojedynczym skanowaniu ($\text{cena} \times \text{wprowadzona ilość}$), dodanie pozycji do rachunku i podliczenie całkowitej kwoty do zapłaty, przesłanie paragonu do wydruku oraz zapisanie go w historii kasy (zapis do pliku o nazwie id drukarki jako zaślepka)
- warstwę modelu danych(rys.3) : baza danych SQLite przechowująca informacje o produktach(kod, nazwa, cena)

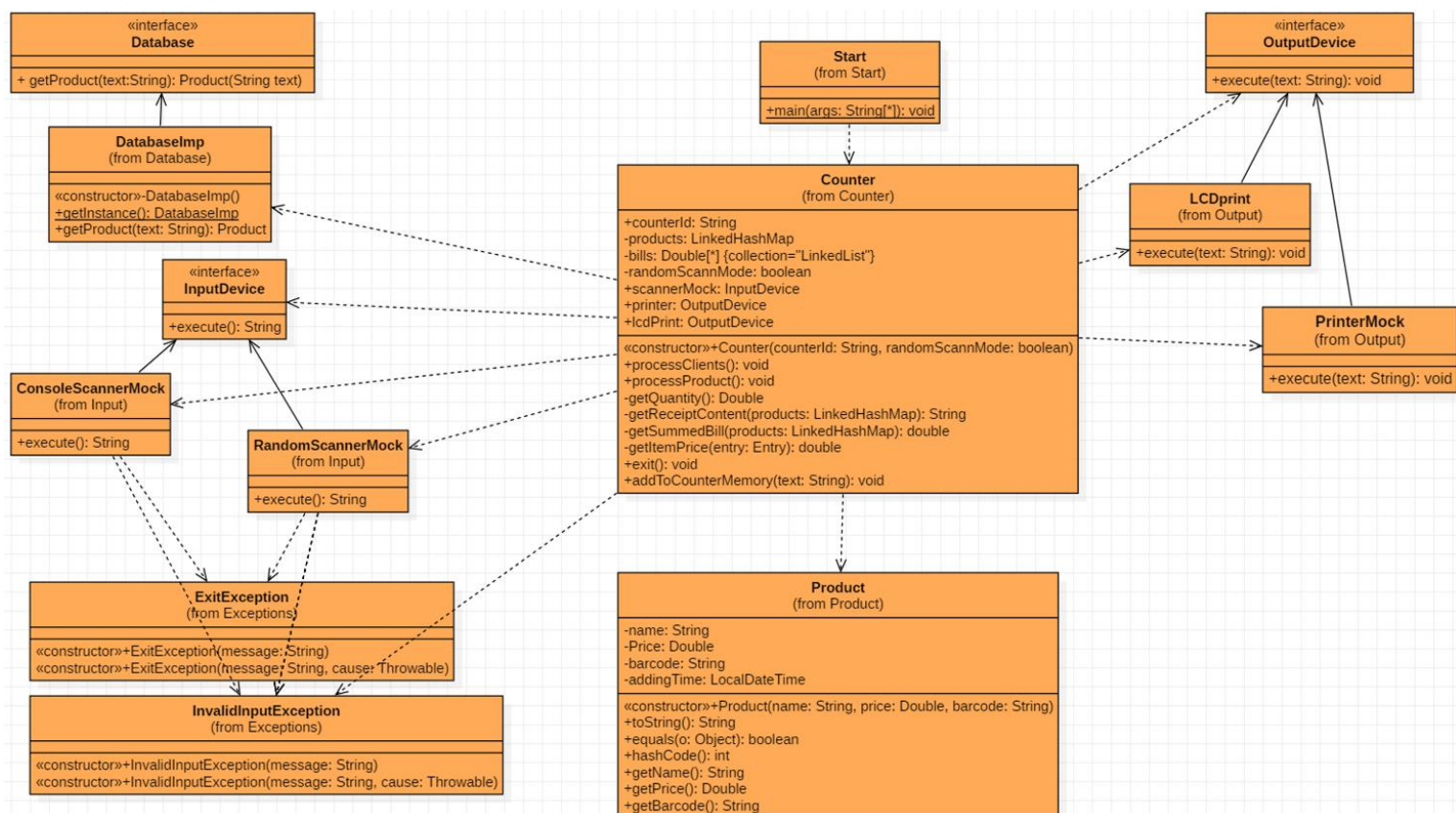
Implementacja składa się z następujących komponentów:

- Kasa – „Counter”
- Baza danych – „Database”
- Urządzenie wejściowe (skaner) – „Input”
- Urządzenie wyjściowe (drukarka, ekran LCD) – „Output”
- Pakiet wyjątków – „Exceptions”
- Aplikacyjny model danych – „Produkt”
- Komponent uruchomieniowy – „Start”

Rys. 1 Schemat funkcjonalny aplikacji




Rys. 2 Diagram klas



3. Model danych

W aplikacji zastosowano model danych: kody– primary key (BARCODE), nazwy (NAME), ceny (PRICE).

Rys. 3 Schemat encji bazy danych

PRODUCTS_LIST	
 <u>BARCODE</u>	TEXT
NAME	TEXT
PRICE	REAL

4. Procedura uruchomieniowa

Aplikacja uruchamiana jest w trybie konsolowym. Na wejściu może przyjmować parametr „-c” – wtedy aplikacja uruchamiana zostaje w trybie „ConsoleScannerMock” - zaślepki skanowania jako ręcznego wprowadzania kodów za pomocą klawiatury. W każdym innym przypadku zostanie uruchomiona w trybie „RandomScannerMock”, gdzie kody losowane są z wbudowanej tablicy testowej.


a. Dane testowe


Do przetestowania aplikacji została użyta baza danych „ProductsDB” zaimplementowana przy pomocy narzędzia SQLite. Imituje ona bazę produktów danego sklepu. Zawiera ona w sobie informacje o produktach (kody– primary key, nazwy, ceny).

BARCODE - PRIMARY KEY (TEXT)	NAME (TEXT)	PRICE(REAL)
11111111	Smietana Super	3.99
33443344	Maslo Extra	6.59
44330908	Baton orzechowy	2.99
44330901	Bulka wiejska	00.59
44330902	Ziemniaki	3.00
44330903	Mydło waniliowe	7.99
44330904	Woda niegazowana	2.20
44330905	Sok pomarańczowy	4.99
44330906	jablka 1kg	3.50
44330907	Mleko 1l	4.50
14330907	Zurawina suszona	8.99
24330907	Płyn do prania XXL	49.99
22222222	Zel pod prysznic DUO	15.99
33333333	Makaron spaghetti	5.40

W trybie RandomScannerMock wykorzystywane jest losowanie kodu z tablicy. W implementacji Points of Sale została stworzona tablica „randomScanInputs”, w której umieszczono możliwe do zeskanowania kody (oprócz poprawnych kodów produktów, znajdują się w niej kody o niepoprawnym formacie lub puste, kody o poprawnym formacie, ale bez odpowiedników w bazie produktów oraz kody oznaczające „exit” czyli zakończenie skanowania produktów danego klienta). Losowana jest liczba z zakresu od 0 do ilość elementów tablicy pomniejszona o 1, a następnie odczytywany jest element tablicy o indeksie tej liczby i przekazywany jako kolejny zeskanowany kod.

```
String[] randomScanInputs = {"11111111", "33443344", "44330908", "44330901", "44330902",  
"44330903", "44330904", "44330905", "44330906", "44330907", "14330907", "24330907", "22222222",  
"33333333", "44444444", "55555555", "66666666", "77777777", "88888888", "exit", "EXIT", "ExiT", "eXiT",  
"", "123", "123456789", "abcd"};
```

 - kody poprawne, z odpowiednikiem w bazie produktów

 - kody poprawne, ale bez odpowiednika w bazie

 - kody oznaczające zakończenie zakupów danego klienta

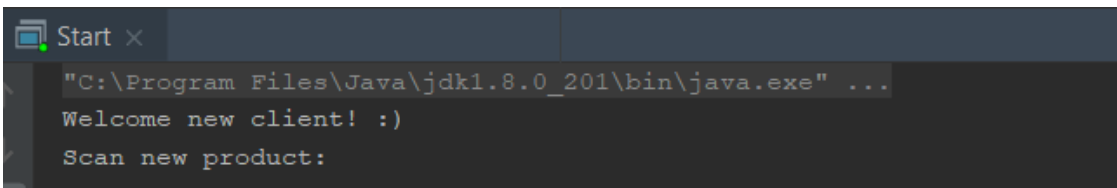
 - kody niepoprawne lub puste

b. Przykładowe uruchomienie w trybie ConsoleScannerMock

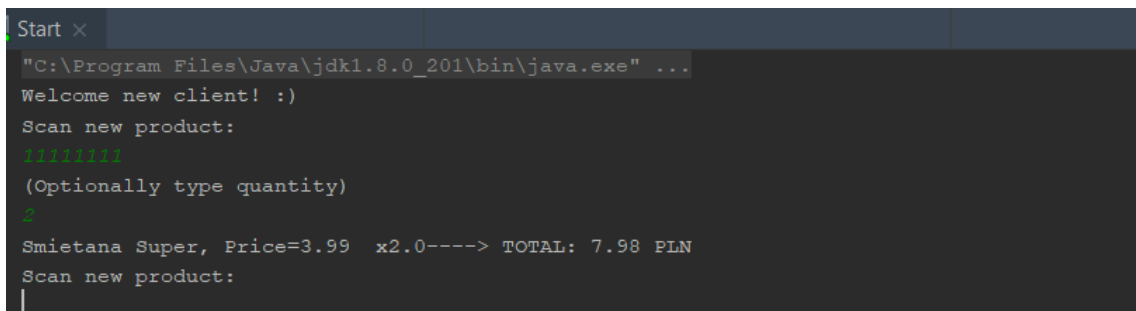
1. Start aplikacji na kasie nazwanej „Kasa_fiskalna_tryb_console1” z parametrem -c, dzięki któremu parametr randomScannMode w konstruktorze kasy przyjmie wartość **false**.

```
Counter Kasa_fiskalna_tryb_console1 = new Counter( counterid: "Kasa_fiskalna_tryb_console1", randomScannMode);  
while (true) {  
    Kasa_fiskalna_tryb_console1.processClient();  
}
```

2. Po uruchomieniu na ekranie konsoli pojawia się powitanie oraz prośba o zeskanowanie produktu.



3. Wprowadzamy ręcznie kod produktu: „11111111”, odpowiadający śmietanie za 3.99 PLN. Po zatwierdzeniu, wyświetlona zostaje prośba o podanie ilości. Wpisujemy „2” i zatwierdzamy. Wtedy do naszego rachunku dopisywana jest śmietana x2 oraz ponownie wyświetlana jest prośba o zeskanowanie produktu.



4. Bez wprowadzania kodu, zatwierdzamy. Na konsoli pojawia się informacja o niepoprawnym kodzie. Ponownie wprowadzamy kod – tym razem również błędny: „12a45678” i zatwierdzamy- na konsoli ponownie pojawia się informacja o niepoprawnym kodzie. Wprowadzamy poprawny kod (8 cyfr), ale dla nieistniejącego produktu w bazie: „12345678” i po zatwierdzeniu wyświetla się informacja „Product not found”.

```
Smietana Super, Price=3.99  x2.0----> TOTAL: 7.98 PLN
Scan new product:

Invalid bar-code
Scan new product:
12a45678
Invalid bar-code
Scan new product:
12345678
Product not found
Scan new product:
```

5. Wprowadzamy kod żelu pod prysznic: „22222222”, za 15,99 PLN, zatwierdzamy, ilości nie podajemy (zatwierdzamy puste pole – domyślna ilość jest równa 1). Wyświetlane są szczegóły produktu i prośba o zeskanowanie kolejnego produktu. Następnie ponownie dodajemy śmietanę (kod: „11111111”) z domyślną ilością. Ponownie wyświetlane są szczegóły produktu i prośba o zeskanowanie kolejnego produktu. Podajemy kod jabłek: „44330906” i ilość „0.7” (kod przypisany jest do ceny kilograma, a nie sztuki). Wyświetlona zostaje nazwa oraz cena za 0.7kg jabłek i zachęta do dalszego skanowania.

```
Scan new product:
22222222
(Optional type quantity)

Zel pod prysznic DUO, Price=15.99  x1.0----> TOTAL: 15.99 PLN
Scan new product:
11111111
(Optional type quantity)

Smietana Super, Price=3.99  x1.0----> TOTAL: 3.99 PLN
Scan new product:
44330906
(Optional type quantity)
0.7
jablka 1kg, Price=3.5  x0.7----> TOTAL: 2.45 PLN
Scan new product:
|
```

6. Wprowadzamy kod „exit” (wielkość liter nie ma znaczenia) i zatwierdzamy. Na ekranie pojawia się informacja o całkowitej sumie rachunku oraz przywitanie nowego klienta i zachęta do ponownego skanowania. Paragon drukowany jest do pliku receipt.txt.

```
Scan new product:
exit
TOTAL BILL: 30.41PLN

Welcome new client! :)
Scan new product:
|
```

7. Jako następny klient wprowadzamy kolejne produkty: „33443344”, Masło Extra, 6.59 x „2”, „24330907”, Płyn do prania XXL, 49.99, „44330901”, Bulka wiejska, 00.59 x „4”.

```
PrinterDeviceMock
Start x

Welcome new client! :)
Scan new product:
33443344
(Optionally type quantity)
2
Masło Extra, Price=6.59  x2.0----> TOTAL: 13.18 PLN
Scan new product:
24330907
(Optionally type quantity)

Płyn do prania XXL, Price=49.99  x1.0----> TOTAL: 49.99 PLN
Scan new product:
44330901
(Optionally type quantity)
4
Bulka wiejska, Price=0.59  x4.0----> TOTAL: 2.36 PLN
Scan new product:
exit
TOTAL BILL: 65.53PLN

Welcome new client! :)
Scan new product:
```

8. Jako trzeci klient wprowadzamy kolejne produkty: „44330908”, Baton orzechowy, 2.99, „44330905”, Sok pomarańczowy, 4.99 x „2”

```
Welcome new client! :)
Scan new product:
44330908
(Optionally type quantity)
1
Baton orzechowy, Price=2.99  x1.0----> TOTAL: 2.99 PLN
Scan new product:
44330905
(Optionally type quantity)
2
Sok pomarańczowy, Price=4.99  x2.0----> TOTAL: 9.98 PLN
Scan new product:
exit
TOTAL BILL: 12.97PLN

Welcome new client! :)
Scan new product:
|
```

9. Sprawdzamy zawartość pliku drukowania resources>>PrinterDeviceMock>>receipt.txt. W pliku zapisany jest paragon ostatniego klienta (ostatnio „drukowany”).

1	Baton orzechowy, Price=2.99 x1.0	= 2.99 PLN
2	Sok pomarańczowy, Price=4.99 x2.0	= 9.98 PLN
3	TOTAL BILL: 12.97PLN	

10. Sprawdzamy zawartość pliku pamięci drukarki

resources>>CounterMemoryMock>>Kasa_fiskalna_tryb_console1.txt. W pliku zapisane są wszystkie paragony wydane przez daną kasę wraz z podanym stanem kasy w konkretnym czasie(po każdym zaktualizowaniu- dodaniu paragonu do pamięci). Ostatnia wartość jest sumą kwot wszystkich paragonów – najaktualniejszym stanem kasy.

```
1 Smietana Super, Price=3.99 x2.0      = 7.98 PLN
2 Zel pod prysznic DUO, Price=15.99 x1.0    = 15.99 PLN
3 Smietana Super, Price=3.99 x1.0      = 3.99 PLN
4 jablka 1kg, Price=3.5 x0.7          = 2.45 PLN
5 TOTAL BILL: 30.41PLN
6 -----FOR: 2019-03-20T23:50:20.114 OVERALL IS: 30.41 -----
7 Maslo Extra, Price=6.59 x2.0          = 13.18 PLN
8 Plyn do prania XXL, Price=49.99 x1.0    = 49.99 PLN
9 Bulka wiejska, Price=0.59 x4.0        = 2.36 PLN
10 TOTAL BILL: 65.53PLN
11 -----FOR: 2019-03-21T00:04:14.463 OVERALL IS: 95.94 -----
12 Baton orzechowy, Price=2.99 x1.0      = 2.99 PLN
13 Sok pomaranczowy, Price=4.99 x2.0     = 9.98 PLN
14 TOTAL BILL: 12.97PLN
15 -----FOR: 2019-03-21T00:08:20.652 OVERALL IS: 108.91 -----
16
```

W przypadku uruchomienia aplikacji w trybie RandomScannerMock aplikacja działa w identyczny sposób, z tą różnicą, że kody są generowane losowo (ilość produktu nadal jest wprowadzana ręcznie).