**Dokumentacja Projektowa** oprogramowania do obsługi automatu biletowego

Opracował: Michał Stężała

Wersja dokumentacji: 1.0

Wrocław, 2017

Spis treści

Wstęp 3

1. Scenariusze aplikacji 4

1.1. Typowy scenariusz 4

1.2. Scenariusze nietypowe 6

1.3. Scenariusze wyjątkowe 7

2. Opis architektury 8

3. Poprawki i funkcjonalności, które powinny zostać wykonane w następnej wersji projektu 10

# Wstęp

Niniejszy dokument jest częścią dokumentacji projektu informatycznego „Automat do biletów”. Projekt ten polega na zaprojektowaniu i zaimplementowaniu oprogramowania do obsługi automatu biletowego. Drugą część dokumentacji przeznaczona jest dla projektantów i programistów, ponieważ jest to dokumentacja kodu (Java Docs).

„Automat do biletów” powinien pozwalać:

* Drukować bilety czasowe jak i jednorazowe. Istnieje kilka rodzajów biletów:
* Czasowe
* 30-to minutowy w cenie 3,00 PLN
* 60-cio minutowy w cenie 4,50 PLN
* 90-cio minutowy w cenie 6,00 PLN
* 24-godzinny w cenie 11,00 PLN
* Jednorazowe
* Normalne w cenie 3,00 PLN
* Specjalne w cenie 3,20 PLN
* Dla każdego rodzaju biletu ustawiać (na życzenie kupującego) zniżkę w wysokości 35% ceny normalnej.
* Kupować bilet można jedynie za pomocą następujących rodzaji monet:
* 0,10 PLN
* 0,20 PLN
* 0,50 PLN
* 1,00 PLN
* 2,00 PLN
* 5,00 PLN
* Wydawać resztę, jeśli zapłacono więcej
* Wyświetlać rodzaje biletów i ich cenę
* Anulować zakup. Wówczas powinny zostać zwrócone monety, które zostały wrzucone.
* Wyświetlać odpowiedni komunikat w przypadku braku określonych biletów lub monet do wydania reszty

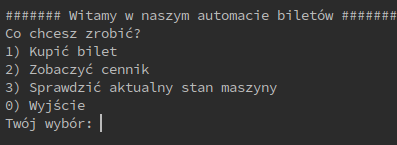
Aplikację zaimplementowano w języku Java w wersji 8. Do uruchomienia aplikacji wystarczy środowisko, pozwalające uruchomić aplikacje napisane w tym języku, dla tej właśnie wersji. Elementy projektowe aplikacji zostały wykonane w narzędziu „Astah Community 7.2.0.1”.

# Scenariusze aplikacji

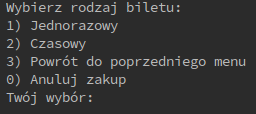
Aplikacja posiada jedynie interfejs konsolowy. Aplikacja pyta użytkownika co chce zrobić, a użytkownik wybierania interesującą go opcje.

## Typowy scenariusz

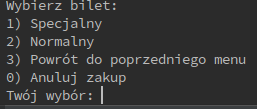
Typowy scenariusz jest następujący. Najpierw użytkownik uruchamia aplikację. Aplikacja wyświetla główne menu:



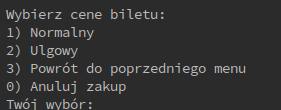
Następnie użytkownik chce kupić bilet, więc wybiera „1”. Wyświetla się następujący ekran:



Użytkownik jest proszony o wybranie rodzaju biletu. Założono, iż użytkownik chce kupić bilet jednorazowy normalny, więc użytkownik wybiera „1”. Wyświetla się następujący ekran:



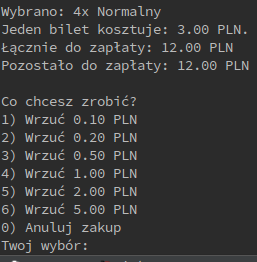
Zgodnie z założeniem, użytkownik chce kupić bilet jednorazowy normalny, więc wybiera „2”. Wyświetla się następujący ekran:



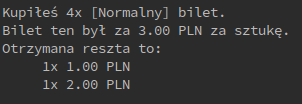
Na tym etapie aplikacja pyta użytkownika o cenę biletu, tzn. czy chce kupić bilet ulgowy czy normalny. Załóżmy, iż użytkownik chce kupić bilet po cenie normalnej, więc wybiera „1”. Wyświetla się następujący ekran:



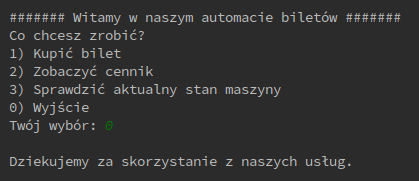
Na tym etapie użytkownik wpisuje ile biletów wybranego rodzaju o wybranej cenie chce kupić. Musi podać liczbę całkowitą z przedziału 1-10 włącznie. Załóżmy, iż użytkownik chce kupić 4 bilety normalne, więc wybiera „4”. Wyświetla się następujący ekran:



Na tym etapie symulowane jest wrzucanie monet o odpowiednim nominale przez użytkownika. Ekran ten (wraz z aktualizacją linii „pozostało do zapłaty”) będzie powtarzał się tak długo dopóki cała kwota nie zostanie uiszczona. Załóżmy, iż użytkownik wrzucił 3x 5.00 PLN. Wówczas wyświetli się następujący ekran:

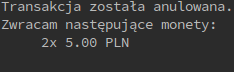


I następnie wyświetli się menu główne aplikacji. Kolejno użytkownik chce wyjść z aplikacji. W tym celu wybiera opcję „0”.



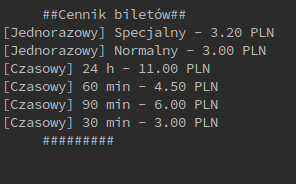
## Scenariusze nietypowe

Podczas realizacji typowego przypadku użycia, można na większości ekranach(oprócz menu głównego i ekranu wyświetlającego kupione bilety i resztę oraz podanie ilości biletów) „anulować zakup”. Opcja ta znajduje się pod przyciskiem „0”. Wybranie tej opcji powoduje wyświetlenie odpowiedniego komunikatu i zwraca monety, które użytkownik wrzucił.

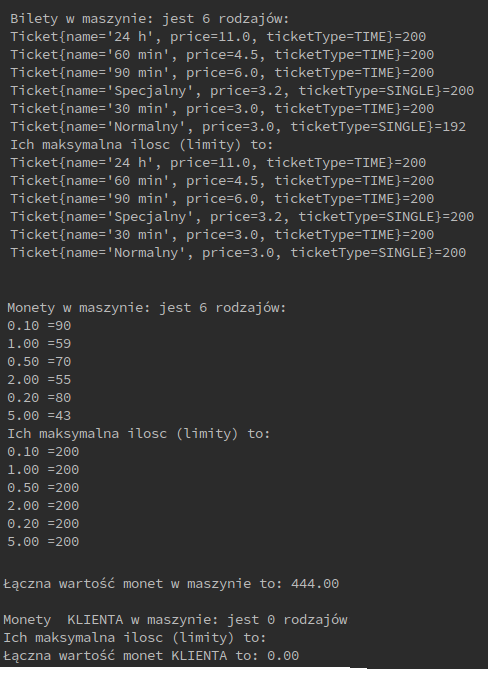


Wybranie opcji „Powrót do poprzedniego menu” powoduje, tak jak w nazwie, wyświetlenie poprzedniego ekranu. **Jedynym wyjątkiem** jest ekran, w którym wybieramy cenę biletu. Wybranie tej opcji spowoduje cofnięcie do ekranu z wyborem rodzaju biletu.

Użytkownik, będąc w menu głównym może wybrać opcję „Zobaczyć cennik”. Wówczas wyświetla się następujący ekran:



Jako dodatkową funkcjonalność , zaprojektowaną głównie dla np. serwisantów maszyny jest opcja „Sprawdzić aktualny stan maszyny”, która wyświetla ile aktualnie w maszynie znajduje się biletów danego rodzaju, ile aktualnie w maszynie znajduje się monet danego rodzaju oraz jakie są limity „slotów” w automacie zarówno dla biletów jak i dla monet danego rodzaju. Wyświetla również informację dot. monet wrzuconych przez użytkownika, do którego proces zakupu biletu nie doszedł końca. Podaje również wartość monet będących aktualnie w maszynie i monet wrzuconych przez klienta.



## Scenariusze wyjątkowe

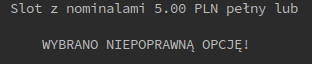
W dowolnym ekranie użytkownik może popełnić błąd i wpisać opcję, która nie istnieje, bądź litery. Wówczas pojawi się komunikat „Wybrano niepoprawną opcję” i dodatkowego wyświetli się ekran, w którym wystąpiła pomyłka.



Podczas wprowadzania ilości biletu może zdarzyć się sytuacja, iż maszyna nie posiada takiej ilości biletów, o czym poinformuje następującym komunikatem:



Podczas wrzucania monet do automatu może zdarzyć się sytuacja, w której dany „slot” na dany rodzaj monety został zapełniony. Wówczas zostanie wyświetlony następujący ekran:

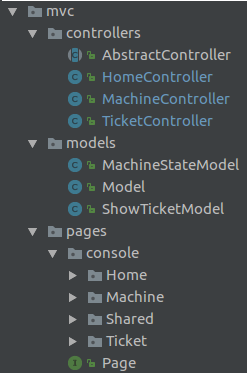
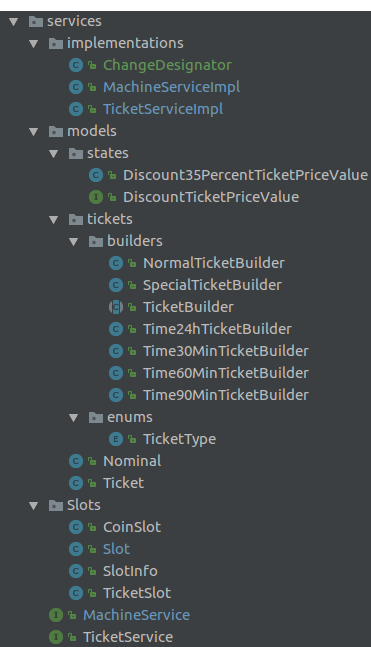


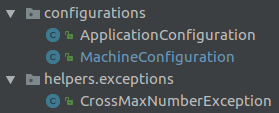
# Opis architektury

Aplikacja „Automat do biletów” została podzielona na dwie główne warstwy i pakiety pomocniczo-narzędziowe.

* Warstwa prezentacji znajduje się w pakiecie „mvc”, które przechowuje pakiety „models”, „pages”,”controllers”, które odpowiadają podziałowi architektury MVC.
* Drugą główną warstwę – logiki biznesowej – przechowuje pakiet „services”. Pakiet ten podzielony został na klasy odpowiadające funkcjonalności związanej z konkretną częścią serwisu (tzn. obsługa biletów i obsługa monet). Zawiera również pakiet „models”, który przechowuje podstawowe „klocki” aplikacji.
* Pomocniczo-narzędziowe pakiety to pakiety „configurations”, która ustawia wstępną konfigurację maszyny oraz „helpers”, która dostarcza przydatnych narzędzi programiście.

Podział pakietów od strony środowiska programistycznego został przedstawiony poniżej:





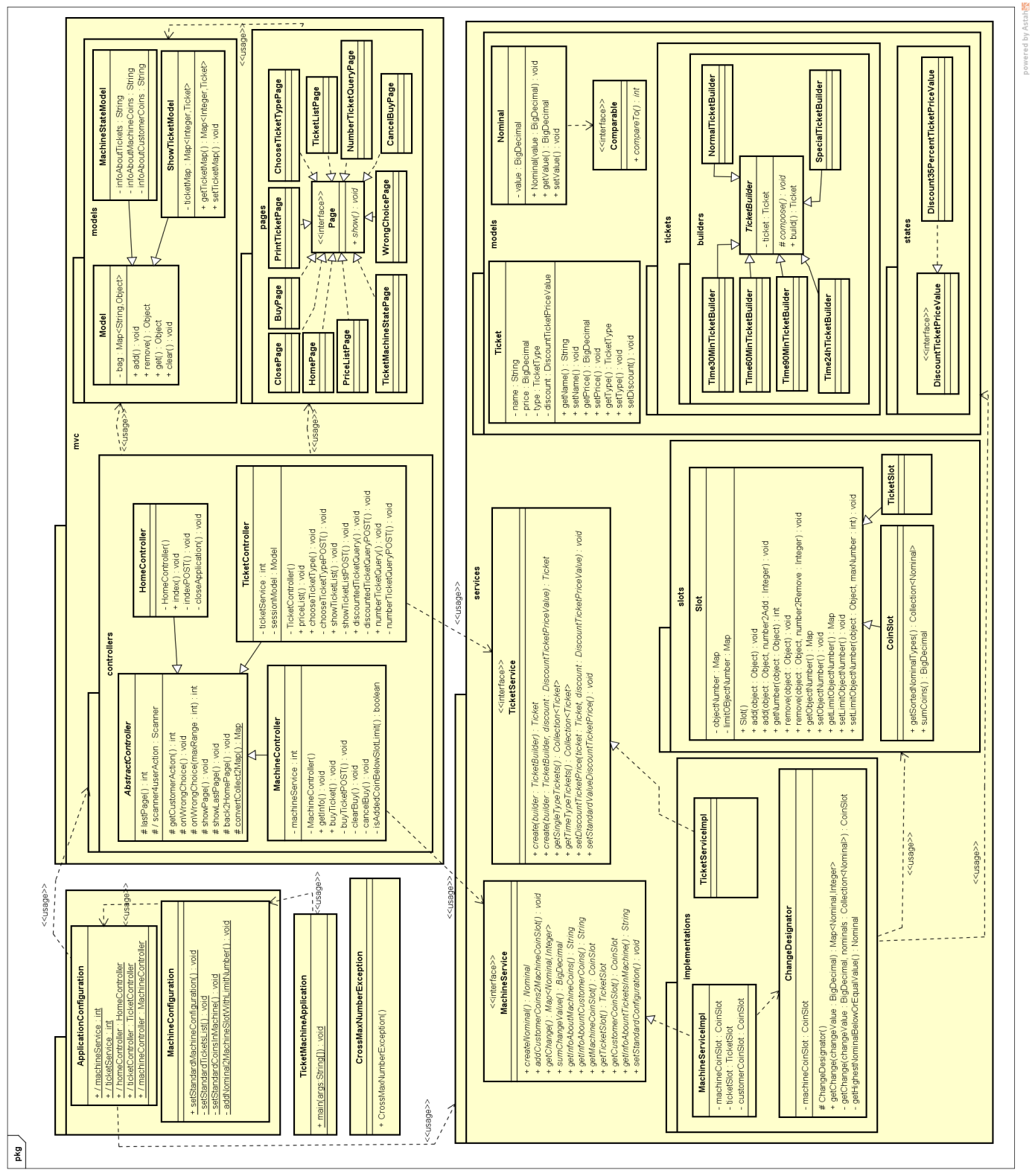


Diagram pakietów, który zawiera również większość wewnętrznych pakietów i klas został przedstawiony poniżej:

**UWAGA:** Diagram ten jest nie zgodny z językiem UML. Jednak celem autora było naszkicowanie większego fragmentu systemu z podziałem na pakiety, podpakiety i klasy oraz oznaczenie „powiązań komunikacyjnych”, między pakietami i ew. klasami, w taki sposób aby było to czytelne.

**UWAGA 2:** Opis poszczególnych metod znajduje się w kodzie źródłowym oraz w wygenerowanym „java docs” dokumencie. Opis ten dostępny jest jedynie w języku angielskim.

# Poprawki i funkcjonalności, które powinny zostać wykonane w następnej wersji projektu

* Oddzielenie podpakietu „models” od pakietu „services”,
* Refaktoryzacja metody buyPOST w MachineController,
* Ustandaryzowanie nazw (ustawić wszędzie albo coin albo nominal, zmienić nazwę serwisu MachineService),
* Usunięcie pól („slotów”) z machineServiceImpl i ticketServiceImpl poprzez przeniesienie ich w inne miejsce,
* Usuniecie getrów dla pól („slotów) z machineServiceImpl i ticketServiceImpl,
* Przerobienie tak wywoływania kolejnych metod w kontrolerze, w taki sposób, aby kolejne wywołania metod zakańczały się, a nie były odkładane na stos. (może mechanizmy refleksji ?),
* Uproszczenie niektórych Page’s,
* Poprawienie SlotInfo, bądź wchłonięcie tej klasy do toString() klasy Slot(),
* Umożliwienie konfiguracji maszyny z zewnątrz (np. pliku XML) – w tym ustalanie wartości ulgi dla biletów,
* Wykonanie „menu” dla serwisanta maszyny,
* Wykonanie większej liczby testów jednostkowych